



# Version 3.0 Release Notes

tentative, as of May 4, 2022

*Package-X* 3.0 is the second major update since its original release. *Package-X* 3.0 features a completely overhauled Dirac and tensor algebraic system, with **FermionLineExpand** and **LoopIntegrate** having been rewritten from scratch. These functions now perform over an order of magnitude faster than in earlier versions. *Package-X* now reduces **FermionLine** and **FermionLineProduct** objects to a minimal set of standard matrix elements for up to 4 external particles with any mass configuration. Finally, *Package-X* features a suite of functions to manipulate tensor expressions, and for carrying out spin/polarizations sums to calculate differential scattering cross sections.

## Bug Fixes since version 2.1.1

- Fixed an issue that would occasionally cause glyphs to turn red in the front end (thanks, Douglas Tuckler).
- Addressed a graphical issue with *Mathematica* 12.0 or earlier running on Windows with 4K displays causing the Feynman slash notation to be imperceptible in **TraditionalForm** (thanks, Ezra Lesser).
- Addressed an issue that caused machine precision evaluation of **ScalarD0** functions to return incorrect results in *Mathematica* 12.0 and above (thanks, Hai-Qing).
- Fixed an issue that caused a slowdown of numerical evaluation of special functions **ScalarC0**, **ScalarD0**, **ScalarC0IR6**, etc in *Mathematica* 12.0 and above.
- Fixed a bug in **LoopRefine** that would cause it to return an IR divergent **ScalarD0** that immediately evaluates to **ComplexInfinity** producing an error message rather than given an expression with a  $1/\epsilon$  pole (thanks, Hai-Qing and Yannick Ulrich).
- Fixed a bug in **LoopRefine** that caused the reduction of single-scale IR divergent tensor **PVD** functions to return an incorrect result (thanks, Darren Scott).
- Fixed a bug that caused **LoopRefine** with setting **Part->Discontinuity** to return an incorrect result (thanks, Hai-Qing and Si-Wei Hu).
- Fixed a bug that caused **ContinuedDiLog** to simplify down to an incorrect expression for certain symbolic arguments (thanks, Hai-Qing and Si-Wei Hu).
- Fixed a bug that caused **C0Expand** to return an analytic expression for **ScalarC0[0,s,0,0,m,m]** that was inconsistent with the  $+i\epsilon$  prescription (thanks, Hai-Qing).
- Fixed a bug that caused **D0Expand** to return an incorrect analytic expression for **ScalarD0IR16** for the case  $m_2 = 0$  (thanks, Yannick Ulrich).

## Additional Functionality

- Added a palette for point-and-click entry of *Package-X* glyphs into a *Mathematica* notebook.
- **LTensor** now accepts integer-valued Lorentz indices **0**, **1**, **2**, or **3**, and functions that manipulate these objects (**Spur**, **LoopIntegrate**, **Contract** etc.) now support them.
- Added new tensor manipulation functions **TensorCollect**, **Uncontract**, and new query functions **TensorList**, **DummyIndices**, and **FreeIndices**.

*Developer comment: The functionality of these routines were previously found in the X`Utilities` context under different names (CollectByTensorStructures, etc.) and were not meant for regular use. But now, as the scope of Package-X broadens, it is appropriate for them to become available as part of the collection of core tensor manipulation functions.*

- Added **LoopReduce** to algebraically reduce Passarino-Veltman functions **PVA**, **PVB**, **PVC**, and **PVD** to scalar functions valid for any  $d(\text{Dim})$ . Provides the same functionality as **PVReduce** expansion for *Package-X* 2.0.

## Amplitudes and Cross Sections

- Added new symbols **PolVecE** $[p, m]$  and **PolVecEC** $[p, m]$  (or  $e[p, m]$  and  $e^*[p, m]$ ) to represent polarization four-vectors.
- Added new function **FermionLineTranspose** to transpose **FermionLine** objects or to perform Fierz transformations of **FermionLineProduct** objects.
- Added new function **AmplitudeConjugate** to calculate the complex conjugate of a scattering amplitude.
- Added new function **AmplitudeInnerProduct** to calculate the square magnitude of an amplitude, and perform sums over spin and polarization states.

## LoopIntegrate

-  Syntax of **LoopIntegrate** is changed so that denominator specifiers are now enclosed in a **List** (version 2.0 syntax is still accepted).

**LoopIntegrate** $[q_\mu, q_\nu, q, \{q, m\}, \{q+p, M\}] \rightarrow \text{LoopIntegrate}[q_\mu, q_\nu, q, \{\{q, m\}, \{q+p, M\}\}]$

*Developer comment: This is the second time the syntax for **LoopIntegrate** has changed since original release. My apologies. The reason for this is two-fold: (1) in preparation for its extension to multiloop integrals, and (2) in anticipation of introducing new related functions like **AsymptoticLoopIntegrate** which requires additional arguments. Also, users who manipulate denominator factors before passing them to **LoopIntegrate** will find this change welcome.*

- Default value of option **Apart** to **LoopIntegrate** is changed to **Apart→Automatic**.

*Developer comment: Since **LoopRefine** does not work with **PVX** functions (5- and higher-point Passarino-Veltman functions), a growing misconception is that Package-X is incapable of dealing with higher point functions at all. On the contrary, for low energy EFT applications, these integrals are easily expressible in terms of lower point functions (**PVD**, **PVC**, ...) by expanding the integrand into partial fractions. Previously, this was turned off by default to avoid kinematic variables appearing in the denominator, and most users were unaware this behavior was changeable. Because of its importance, **LoopIntegrate** will now by default perform the expansion into partial fractions for 5- and higher-point functions.*

- Removed option **Dimensions** from **LoopIntegrate** and from Passarino-Veltman functions **PVA**, **PVB**, **PVC** and **PVD**.

*Developer comment: Since **LoopIntegrate** gives results valid for any  $d$  (provided no Dirac gamma-5 matrix or Levi-Civita symbols in the numerator), and that the Passarino-Veltman functions are already defined for arbitrary  $d$ , this option was completely superfluous. Setting option **Dimensions** to **LoopRefine** and **LoopRefineSeries** will appropriately expand the input expression containing Passarino-Veltman functions around the desired integer number of spacetime dimensions (thanks, Logan Morrison).*

- Option **Organization** to **LoopIntegrate** no longer takes **Function** as a possible setting. The default option setting **LTensor** is renamed to **Automatic**, and collects the result in terms of tensor structures/standard matrix elements, as before.

- Added options **ChiralBasis→Automatic**, **ChisholmExpand→Automatic**, **GordonIdentity→True** to **LoopIntegrate** to control how **DiracMatrix** and **FermionLine** objects are manipulated.

*Developer comment: Before, **LoopIntegrate** would use options set to **FermionLineExpand** to decide how to manipulate **DiracMatrix** and **FermionLine** objects. To improve upon this awkward design choice, **LoopIntegrate** now gets dedicated options for these.*

- Added option **ConservationConstraint→None** to **LoopIntegrate**, and if appropriately set, facilitates the reduction of **FermionLine** and **FermionLineProduct** objects to a minimal set of standard matrix elements for up to four external particles. It is also used in the expansion of the integrand into partial fractions.

## LoopRefine and LoopRefineSeries

- With the setting **Part→UVDivergent**, **LoopRefine** can now return the UV divergent pole part of 5- and higher point-Passarino-Veltman functions **PVX**.
- **Not yet implemented.** **LoopRefineSeries** no longer uses **Series** internally to construct Taylor series expansions of Passarino-Veltman functions, and instead, now uses lightweight codes rewritten from scratch.

*Developer comment: In recent versions of Mathematica (11 and 12), the performance of the built-in function **Series** has deteriorated to the extent that when applied to expressions of sizes typical to high energy physics, **LoopRefineSeries** would fail to complete in a reasonable amount of time. With this change, all routines of Package-X are finally decoupled from Mathematica's top-level megafunctions.*

## FermionLineReduce (was FermionLineExpand)

-  **FermionLineExpand** is renamed to **FermionLineReduce**.

*Developer comment: With the overhaul of the Dirac algebraic system, it became increasingly apparent that **FermionLineExpand** does much more than to just expand out the **FermionLine(Product)** objects. It reduces them to some minimal set of standard matrix elements — hence the name change. Setting the option **DiracAlgebra→False** still limits it to distributing products of gamma matrices over addition without using the Dirac algebra.*

- **FermionLineReduce** no longer expands out **Inactive[Spur]** objects.
- Added option **ConservationConstraint→None** and if appropriately set, reduces **FermionLine** and **FermionLineProduct** objects representing up to four external lines to a minimal set of standard matrix elements.
- Options **ChiralBasis**, **ChisholmExpand**, and **GordonIdentity** to **FermionLineReduce** no longer control how the reduction is carried out, but instead control the form of the standard matrix elements it returns. The method of reduction is now automatically chosen for best performance and quality of result.
- Option **ChiralBasis** now accepts a list of **True** or **False** and toggles the basis in terms of which each corresponding **FermionLine** inside **FermionLineProduct** objects is expressed.

## Special Functions

- **ScalarD0IR16**, can now be evaluated numerically with machine and arbitrary precision arithmetic.


*Developer comment: Long overdue. Before, **D0Expand** had to be applied to express **ScalarD0IR16** in terms of analytic functions which Mathematica could numerically evaluate.*

- Machine precision evaluation of special functions **ScalarC0**, **ScalarC0IR6**, **ScalarD0**, **ScalarD0IR12**, **ScalarD0IR13**, and **ScalarD0IR16** only use the Wolfram Virtual Machine (WVM) to evaluate, and can no longer be compiled to C. These functions are now automatically compiled and saved to disk when *Package-X* is loaded for the first time.

*Developer comment: In earlier versions, these functions were recompiled every time they were first called in a Package-X session. This was to make it possible to compile to C rather than WVM. But due to lengthy compilation times to C with little gain in performance relative to WVM code, this option was not really sensible. Therefore, this option was removed, and the short wait to compile code to WVM in each Package-X session was eliminated.*

- Added option **Assumptions:>\$Assumptions** to **C0Expand** and **D0Expand**, which when set is used to simplify and resolve the  $\pm i \epsilon$  parts of **DiLog** (thanks, Chris Flett).

## Other Incompatible Changes

-  *Package-X* now requires *Mathematica* 10.1 or later.
- Removed global symbol **\$DiracAlgebra**.
- **LoopRefine** and **LoopRefineSeries** no longer has the option **ExplicitC0**.

## Standard Matrix Elements involving FermionLines

Below is a list of standard matrix elements in terms of which the output of **LoopIntegrate** and **FermionLineReduce** is generated.

1. The basis set is determined by properties of the input expression such as the number of free indices, number of independent external vectors, polarization vectors and the configuration of **FermionLines**.
2. The basis sets are displayed for only a representative configuration of external spinors and polarization vectors, although similar sets are generated for other configurations reachable by crossing symmetry.
3. The precise form of the basis set is influenced by toggling the options **ChiralBasis**, **ChisholmExpand**, and **GordonIdentity**.

### ***F\*F\* (Off shell fermion self energy)***

Number of Standard Matrix elements: 4

Represented by: **DiracMatrix**[...]

Input expression: 0 free indices,  
1 external vector, **p**

Basis set: (**ChiralBasis**→**True**):

$$\not{p}\hat{P}_L, \quad \not{p}\hat{P}_R, \quad \hat{P}_L, \quad \hat{P}_R$$

### ***FFS\* (Off shell scalar)***

Number of Standard Matrix elements: 2

Represented by: **FermionLine**[...]

Input expression: 0 free indices,  
2 external vectors **p1**, **p2**

Basis set: (**ChiralBasis**→**True**):

$$\langle \hat{P}_L \rangle, \quad \langle \hat{P}_R \rangle$$

### ***FFV\* (Off shell vector)***

Number of Standard Matrix elements: 6

Represented by: **FermionLine**[...]

Input expression: 1 free index,  
2 external vectors **p1**, **p2**

Basis set: (**ChiralBasis**→**True**, **ChisholmExpand**→**True**, **GordonIdentity**→**True**):

$$\begin{aligned} \langle \gamma^\mu \hat{P}_L \rangle \epsilon_\mu^*, \quad \langle \sigma^{\mu\{p_1-p_2\}} \hat{P}_L \rangle \epsilon_\mu^*, \quad \langle \hat{P}_L \rangle (p_1 - p_2)^\mu \epsilon_\mu^*, \\ \langle \gamma^\mu \hat{P}_R \rangle \epsilon_\mu^*, \quad \langle \sigma^{\mu\{p_1-p_2\}} \hat{P}_R \rangle \epsilon_\mu^*, \quad \langle \hat{P}_R \rangle (p_1 - p_2)^\mu \epsilon_\mu^* \end{aligned}$$

### ***FFSS***

Number of Standard Matrix elements: 4

Represented by: **FermionLine**[...]

Input expression: 0 free indices,  
4 external vectors **p1** ... **p4**, and option **ConservationConstraint** set.

Basis set: (**ChiralBasis**→**True**, **ChisholmExpand**→**True**):

$$\langle \hat{P}_L \rangle, \quad \langle \not{p}_3 \hat{P}_L \rangle, \quad \langle \hat{P}_R \rangle, \quad \langle \not{p}_3 \hat{P}_R \rangle$$

## FFSV

Number of Standard Matrix elements: 12

Represented by: `FermionLine[...]`

Input expression: 0 free indices,  
4 external vectors  $\mathbf{p}_1 \dots \mathbf{p}_4$ , and option `ConservationConstraint` set.

Basis set: (`ChiralBasis`→`True`, `ChisholmExpand`→`True`, `GordonIdentity`→`False`):

$$\begin{aligned} &\langle \hat{P}_L \rangle p_1 \cdot \epsilon^*, \quad \langle \hat{P}_L \rangle p_2 \cdot \epsilon^*, \quad \langle \gamma^\mu \hat{P}_L \rangle \epsilon_\mu^*, \quad \langle \gamma^\mu \not{p}_3 \hat{P}_L \rangle \epsilon_\mu^*, \quad \langle \not{p}_3 \hat{P}_L \rangle p_1 \cdot \epsilon^*, \quad \langle \not{p}_3 \hat{P}_L \rangle p_2 \cdot \epsilon^*, \\ &\langle \hat{P}_R \rangle p_1 \cdot \epsilon^*, \quad \langle \hat{P}_R \rangle p_2 \cdot \epsilon^*, \quad \langle \gamma^\mu \hat{P}_R \rangle \epsilon_\mu^*, \quad \langle \gamma^\mu \not{p}_3 \hat{P}_R \rangle \epsilon_\mu^*, \quad \langle \not{p}_3 \hat{P}_R \rangle p_1 \cdot \epsilon^*, \quad \langle \not{p}_3 \hat{P}_R \rangle p_2 \cdot \epsilon^*, \end{aligned}$$

## FFVW (not yet implemented)

## FFFF

Number of Standard Matrix elements: 16 + 2

Represented by: `FermionLineProduct[...]`

Input expression: 0 free indices,  
4 external vectors  $\mathbf{p}_1 \dots \mathbf{p}_4$ , and option `ConservationConstraint` set.

Basis set: (`ChiralBasis`→`True`, `ChisholmExpand`→`True`, `GordonIdentity`→`True`):

$$\begin{aligned} \text{S} \times \text{S}: & \langle \hat{P}_L \rangle \langle \hat{P}_L \rangle, \quad \langle \hat{P}_L \rangle \langle \hat{P}_R \rangle, \quad \langle \hat{P}_R \rangle \langle \hat{P}_L \rangle, \quad \langle \hat{P}_R \rangle \langle \hat{P}_R \rangle, \\ \text{D} \times \text{D}: & \langle \gamma_\mu \hat{P}_L \rangle \langle \gamma^\mu \hat{P}_L \rangle, \quad \langle \gamma_\mu \hat{P}_L \rangle \langle \gamma^\mu \hat{P}_R \rangle, \quad \langle \gamma_\mu \hat{P}_R \rangle \langle \gamma^\mu \hat{P}_L \rangle, \quad \langle \gamma_\mu \hat{P}_R \rangle \langle \gamma^\mu \hat{P}_R \rangle, \\ \text{P} \times \text{D}: & \langle \sigma^{\mu\{p_1-p_2\}} \hat{P}_L \rangle \langle \gamma_\mu \hat{P}_L \rangle, \quad \langle \sigma^{\mu\{p_1-p_2\}} \hat{P}_L \rangle \langle \gamma_\mu \hat{P}_R \rangle, \quad \langle \sigma^{\mu\{p_1-p_2\}} \hat{P}_R \rangle \langle \gamma_\mu \hat{P}_L \rangle, \quad \langle \sigma^{\mu\{p_1-p_2\}} \hat{P}_R \rangle \langle \gamma_\mu \hat{P}_R \rangle, \\ \text{D} \times \text{P}: & \langle \gamma_\mu \hat{P}_L \rangle \langle \sigma^{\mu\{p_3-p_4\}} \hat{P}_L \rangle, \quad \langle \gamma_\mu \hat{P}_L \rangle \langle \sigma^{\mu\{p_3-p_4\}} \hat{P}_R \rangle, \quad \langle \gamma_\mu \hat{P}_R \rangle \langle \sigma^{\mu\{p_3-p_4\}} \hat{P}_L \rangle, \quad \langle \gamma_\mu \hat{P}_R \rangle \langle \sigma^{\mu\{p_3-p_4\}} \hat{P}_R \rangle, \\ \text{T} \times \text{T}: & \langle \sigma_{\mu\nu} \hat{P}_L \rangle \langle \sigma^{\mu\nu} \hat{P}_L \rangle, \quad \langle \sigma_{\mu\nu} \hat{P}_R \rangle \langle \sigma^{\mu\nu} \hat{P}_R \rangle \quad (\text{these SMEs can be expressed in terms of the other 16}) \end{aligned}$$