PROCEEDINGS OF SCIENCE

HyperFORM – a FORM package for parametric integration with hyperlogarithms

Adam Kardos,[a,b,*] **Sven-Olaf Moch**[c] **and Oliver Schnetz**[c]

[a] *Department of Experimental Physics, Institute of Physics, Faculty of Science and Technology, University of Debrecen,*
4010 Debrecen, PO Box 105, Hungary

[b] *Institute for Theoretical Physics, ELTE Eötvös Loránd University,*
Pázmány Péter 1/A, H–1117 Budapest, Hungary

[c] *II. Institut für Theoretische Physik, Universität Hamburg,*
Luruper Chaussee 149, D–22761 Hamburg, Germany

E-mail: kardos.adam@science.unideb.hu, sven-olaf.moch@desy.de, schnetz@mi.uni-erlangen.de

HyperFORM brings the parametric integration of hyperlogarithms, weighted by rational prefactors, into the symbolic-manipulation system FORM. It ports the capabilities of Erik Panzer's Maple package HyperInt, capitalizing on FORM's speed with bulky algebraic input and on its ability to spread a single calculation across many processor cores. We keep the description of the method brief and concentrate instead on how the package is organized and driven: a fully self-contained program for the three-loop zigzag period serves as a worked illustration, and timing measurements for zigzags through six loops gauge its present reach. HyperFORM is released openly and applies to a broad class of problems, the evaluation of Feynman integrals prominently among them.

Loops and Legs in Quantum Field Theory (LL2026)
12–17 April, 2026
Bayreuth, Germany

*Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0) All rights for text and data mining, AI training, and similar technologies for commercial purposes, are reserved. ISSN 1824-8039 . Published by SISSA Medialab.

PoS(LL2026)069

1. Introduction

Among the many techniques developed for high-order perturbative calculations [1], those that return analytic answers hold a special place: beyond a number, they lay bare the algebraic and number-theoretic structures that organize the perturbative series. Parametric integration of hyperlogarithms is one such technique. Embedded in the theory of graphical functions, it has pushed the renormalization of scalar field theories to unprecedented orders: six loops in ϕ^3 [2] and seven loops in ϕ^4 [3].

The approach is due to Francis Brown, who introduced it as a general-purpose route to the analytic evaluation of a wide class of integrals [4]. Its applicability rests on a single requirement: each successive integration must land back in the space of hyperlogarithms. Integrals that satisfy this condition are termed *linearly reducible*. Two independent realizations of Brown's algorithm were written in `Maple`, one by Bogner [5] and one by Panzer [6]; the latter, distributed as `HyperInt`, became the de-facto standard for the linearly reducible Feynman integrals within its reach, up to intermediate loop orders.

The method is, however, computationally heavy: intermediate expressions swell quickly, an intrinsic feature of the algorithm, and in `HyperInt` this is compounded by the host system, where `Maple` manages large expressions but is not the most efficient choice, and by the program's limited development after release. Our remedy is a faster engine built on `FORM` [7, 8], chosen because it processes large algebraic expressions rapidly, is freely available and open, and parallelizes efficiently across the many cores of present-day servers. As a first milestone we have reimplemented most of `HyperInt`'s functionality in `FORM`, released as `HyperFORM` [9]; since no new mathematics enters here, the present contribution focuses on what the package offers and how it is used, with the underlying theory left to Refs. [4, 10] and a detailed account in the full paper [9].

2. Hyperlogarithms and parametric integration

At the heart of the method sit the hyperlogarithms, also known as Goncharov polylogarithms [11]. A convenient way to characterize them is through the recursion

$$\partial_z L(a_n, \ldots, a_1; z) = \frac{L(a_{n-1}, \ldots, a_1; z)}{z - a_n}, \qquad L(a_n, \ldots, a_1; 0) = 0, \tag{1}$$

with $a_1, \ldots, a_n, z \in \mathbb{C}$ and $\log(0) = 0$ taken as the convention for their evaluation. The two conditions above fix L uniquely except for the words built solely from zeros, for which the recursion gives $L(0, \ldots, 0; z) = \log^n(z)/n!$ with n the length of the word, so that the value at $z = 0$ does not exist as a limit. It is only in this sense that $\log(0) = 0$ is to be read: the divergent constant $\log 0$, kept as a formal symbol by the regularized limits discussed below, is set to zero, which extends the boundary condition $L(a_n, \ldots, a_1; 0) = 0$ to these words as well. Equivalently they are Chen iterated integrals [12], and they close under the shuffle product $L(u; z)\, L(v; z) = L(u \sqcup v; z)$, the sum $u \sqcup v$ running over every shuffle of the words u and v. When each letter a_i is 0 or -1, the regularized value at ∞ is a multiple zeta value (MZV) $\zeta(n_r, \ldots, n_1)$. At the modest weights relevant here, these MZVs can simply be looked up: `HyperFORM` carries a reduction to a $\mathbb{Q}$ basis taken from the *Multiple Zeta Value Data Mine* [13] bundled with `FORM`.

PoS(LL2026)069

The prime application is to Feynman integrals themselves. Introducing Feynman parameters, an L-loop integral with N propagators can be written as

$$I = \frac{\mathrm{e}^{L\epsilon\gamma_{\mathrm{E}}}\,\Gamma\!\left(\nu - \frac{Ld}{2}\right)}{\prod_{i=1}^{N}\Gamma(\nu_i)} \left(\prod_{i=1}^{N}\int_0^\infty \mathrm{d}x_i\right) \delta\!\left(1 - \sum_{i=1}^{N} x_i\right) \left(\prod_{i=1}^{N} x_i^{\nu_i - 1}\right) \frac{\mathcal{U}^{\nu - \frac{(L+1)d}{2}}}{\mathcal{F}^{\nu - \frac{Ld}{2}}}\,, \tag{2}$$

with $\mathcal{U}$ and $\mathcal{F}$ the first and second Symanzik polynomials, L the number of loops and $d = 4 - 2\epsilon$ the spacetime dimension. Hyperlogarithmic integration exploits two structural properties: each x_i enters $\mathcal{U}$ only to the first power, and the same holds for $\mathcal{F}$ in the absence of internal masses. The projectivity encoded in the δ function is eliminated by the Chen–Wu gauge choice, by which any single parameter may be frozen to 1 while the rest are integrated over $[0, \infty)$. Splitting the (linear) polynomials in the first parameter into partial fractions yields logarithms; on repeating the step the arguments of those logarithms come to depend on the following parameter, and this is precisely how hyperlogarithms arise.

One subtlety deserves emphasis. The original integral is finite, yet linearity makes the algorithm break it into separate pieces, and an individual piece may blow up like $\log(0)^k$ or $\log(\infty)^k$. Such terms are tamed by *regularized limits* [10]: $\log 0$ and $\log\infty$ are carried along as formal constants which, for a convergent integral, cancel out of the answer. `HyperFORM` tracks them throughout and verifies, by default, that the cancellation indeed takes place.

In practice, then, the integrals `HyperFORM` targets are those whose ϵ-expansion is to be obtained order by order, of the generic shape

$$I(\epsilon) = \int_0^\infty \mathrm{d}x_1 \cdots \int_0^\infty \mathrm{d}x_N \prod_{j=1}^{M} f_j^{\,n_j + m_j\epsilon}(x_1, \ldots, x_N)\,, \qquad n_j \in \mathbb{Z},\ m_j \in \mathbb{Q}\,, \tag{3}$$

the f_j being rational in the integration variables. Whether or not such an integral carries poles in ϵ, the divergent case must first be cast into an expandable form by the regularization of Ref. [10], performed ahead of the Laurent expansion.

3. The HyperFORM package

The distribution [9] comprises three `FORM` headers. The procedures themselves reside in `hyperform.h`; the companion `declare-hyperform.h` gathers every symbol, function and preprocessor-variable declaration; and `mzvlow.h` stores the MZV reduction table through weight ten [13], which suffices to carry a calculation to the sixth loop order. Every identifier the package introduces is namespaced, with `HYP` for symbols and functions and `Hyp` for procedures, so that user-chosen names can never collide with internal ones. Greatest-common-divisor and related polynomial manipulations are delegated to `FLINT` [14, 15] through `FORM`'s built-in interface.

The package handles both regular and singular integrals at $\epsilon = 0$: it expands non-integer powers, integrates rational functions as well as hyperlogarithms, evaluates regularized limits, performs series expansions, rewrites limits in a fibration basis, and reduces the outcome to MZVs.

Workflow. A calculation is a short `FORM` program. The user includes the package, defines the integrand as a local expression in terms of numerator/denominator polynomials with ϵ-dependent exponents, and then calls a fixed sequence of procedures:

PoS(LL2026)069

- `HypParseInputExpr` translates the user integrand to the internal representation;
- `HypAutoRegularize` detects and regularizes ϵ-divergences (recursive and automatic; leaves ϵ-finite integrands unchanged);
- `HypEpExpand` expands in ϵ up to the order set by `HYPMAXEP`;
- `HypApplyChenWu` fixes one integration variable to 1 for projective (Feynman) integrals;
- `HypIntegrationStep` integrates over one variable, called in a loop over a chosen integration sequence;
- `HypFinalizeResult` produces a human-readable result (MZV shorthand, ϵ-expansion to the requested order).

The choice of integration sequence matters: for more complex integrals some sequences are linearly reducible while others are not, and different reducible sequences can have very different runtimes.

Test-driven development. `HyperFORM` follows a strictly modular design: if a routine needs more than one sentence to describe, it is broken up. Each of the roughly 90 procedures comes with its own unit test (built on the `check.rb` script shipped with `FORM`), exercising several cases. Development follows the red-green-refactor cycle, so that the algorithms, which are not final, can be updated safely, with regressions caught immediately.

The overall workflow of a `HyperFORM` calculation is summarized in Fig. 1. Starting from a user `FORM` program, the integrand is parsed into the internal representation, optionally regularized (recursively, until ϵ-finite), expanded in ϵ and reduced with the Chen–Wu gauge choice. The heart of the package is the integration step `HypIntegrationStep`, applied once per variable in the chosen integration sequence: it converts the integrand to hyperlogarithms, partial-fractions in the current variable, integrates the hyperlogarithmic and rational parts, and takes the limits at infinity and zero; the limit at infinity triggers the fibration-basis "bootstrap" (differentiation, shuffle regularization and a zero-limit constant). A final normalization and finalization produce the result in terms of MZVs. The same bootstrap is also exposed as a standalone entry point, `HypFibrationBasis`.

PoS(LL2026)069

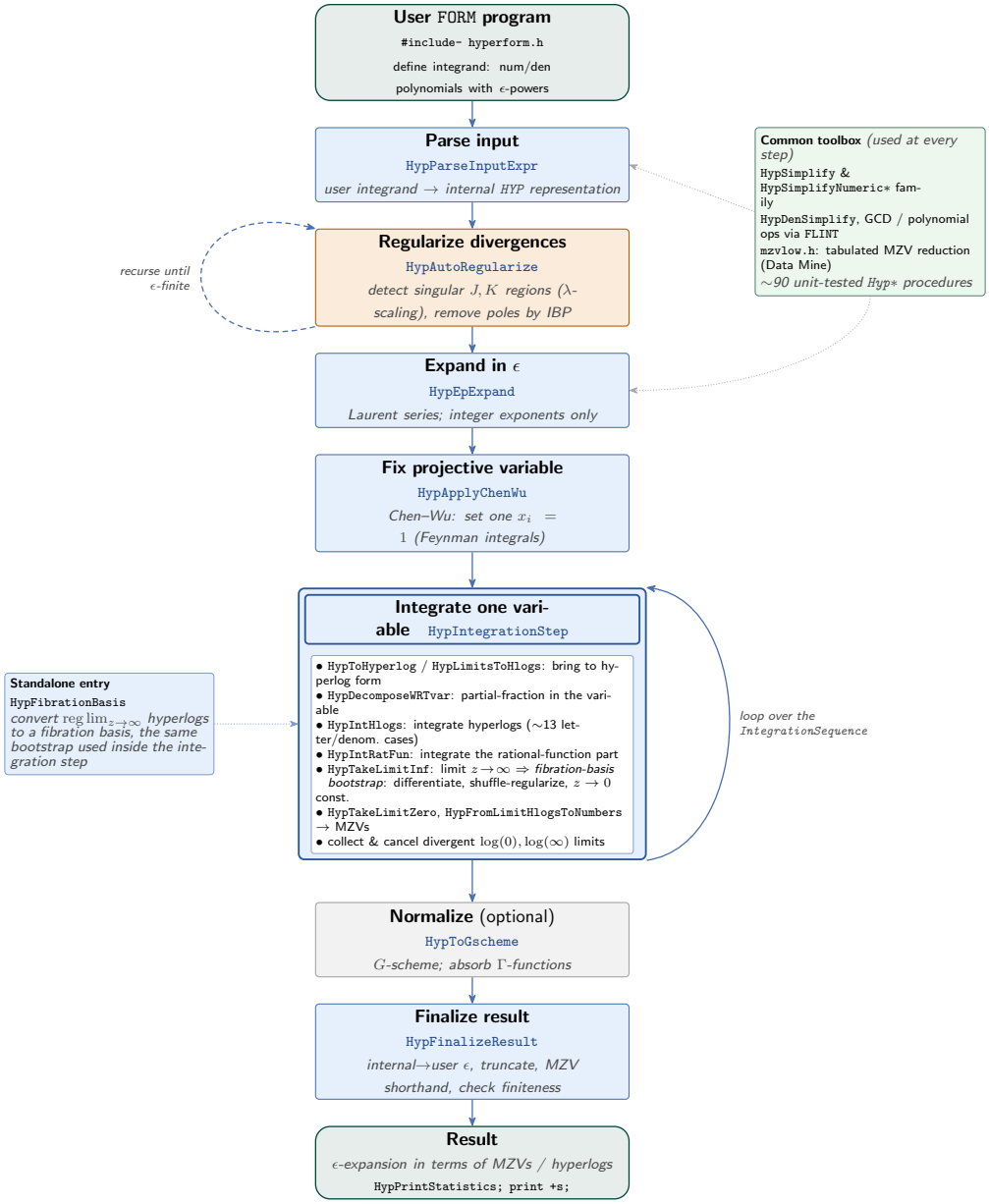

Figure 1: High-level workflow of the HyperFORM package, showing the top-level procedures (typewriter font) and the internal sub-steps of the per-variable integration step. The common toolbox (simplification, polynomial and GCD operations via FLINT, the tabulated MZV reduction) underlies every step; each of the ∼90 procedures is covered by dedicated unit tests.

4. A complete example: the three-loop zigzag period

To show how compact a full calculation is, we give the complete, self-contained FORM program for the three-loop zigzag period Z_3. It is an ϵ-finite, projective integral over six Feynman parameters, whose integrand is $\delta(x_i - 1)/\mathcal{U}_{Z_3}^2$ with the first Symanzik polynomial $\mathcal{U}_{Z_3}$ (16 monomials). We fix x_6 via Chen–Wu and integrate x_1 through x_5:

PoS(LL2026)069

```
#include- hyperform.h
#define IntegralExpr "Z3"
#define IntegrationSequence "1,2,3,4,5"
#define ChenWuVar "6"
symbols x1,...,x6;
symbol ep;
cfunctions num,den;
local 'IntegralExpr' =
  den(  x4*x5*x6 + x2*x5*x6 + x1*x5*x6 + x3*x4*x6 + x1*x4*x6 + x2*x3*x6
      + x1*x3*x6 + x1*x2*x6 + x3*x4*x5 + x2*x4*x5 + x2*x3*x5 + x1*x3*x5
      + x1*x2*x5 + x2*x3*x4 + x1*x3*x4 + x1*x2*x4 )^2;
.sort
#call HypParseInputExpr(ep,num,den,x1,...,x6)
.sort
#call HypEpExpand
#call HypApplyChenWu('IntegralExpr',ChenWuVar)
#do IntVar={'IntegrationSequence'}
  #call HypIntegrationStep('IntegralExpr','IntVar')
#enddo
#call HypFinalizeResult(ep,'HYPMAXEP')
print +s;
.end
```

Note that `num` never appears on the right-hand side of `IntegralExpr`; `HyperFORM` takes an absent numerator to mean a numerator of 1 by default. The program runs in about 0.05 s and returns the known result

$$Z_3 = 6\,\zeta_3\,. \tag{4}$$

No additional code is required; the listing above performs the entire calculation: parsing, Chen–Wu gauge fixing, five integrations and finalization.

5. Benchmarks: the zigzag family

The zigzag graphs (Fig. 2) make an especially convenient testing ground. Their periods are finite in ϵ and linearly reducible [17]; carrying no external scales, their second Symanzik polynomial drops out entirely, and closed forms are available at every loop order. First conjectured by Broadhurst and Kreimer [18], the values were later established by Brown and Schnetz [19], with a separate proof given in Ref. [20],

$$Z_n = 4\,\frac{(2n-2)!}{n!\,(n-1)!}\left(1-\frac{1-(-1)^n}{2^{2n-3}}\right)\zeta(2n-3)\,. \tag{5}$$

In parametric form the period reads $I_{Z_n} = \int_0^\infty \mathrm{d}x_1\cdots\mathrm{d}x_{2n}\,\delta(x_i-1)/\mathcal{U}_{Z_n}^2$, with one variable fixed by the projective constraint.

Although finite and analytically known, the zigzags carry substantial polynomial complexity, which makes them ideal for stress-testing the code. Table 1 lists, per order, the Feynman-parameter count and the number of monomials in $\mathcal{U}_{Z_n}$ (from the `FORM` package `uf` [16]), with runtimes for `HyperInt`, `HyperFORM` and `HyperlogProcedures`; `HyperFORM` reaches Z_6, about 3.5 times faster than `HyperInt` there. The `HyperlogProcedures` column should be read with care, however: it

PoS(LL2026)069

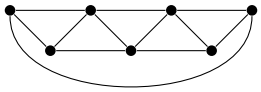

Figure 2: The six-loop zigzag diagram Z_6. The long diagonal is fixed by the Chen–Wu gauge choice.

rests on the theory of graphical functions, an altogether different and far quicker route that only applies in narrow settings (massless, at most three external legs). What parametric integration offers in return is generality.

Diagram	Variables	Monomials in $\mathcal{U}$	HyperInt [s]	HyperFORM [s]	HyperlogProc.* [s]
Z_3	6	16	0.8	0.05	0.04
Z_4	8	45	1.5	0.3	0.04
Z_5	10	130	54	17	0.04
Z_6	12	368	1.0×10^5	2.9×10^4	0.05
Z_7	14	1040	–	–	0.08
Z_8	16	2919	–	–	0.13

Table 1: Size of the zigzag problems, given by the number of Feynman parameters they carry and the monomials in their first Symanzik polynomial, with single-core runtimes from three programs. *HyperlogProcedures follows the graphical-functions approach, which is methodologically distinct.

Good integration sequences. Only some integration orders make the zigzags linearly reducible. Following Brown [17], a good order completes as many cycles as possible in the already-integrated subgraph while sharing as few vertices as possible with the remainder; the largest such count is the *vertex width*. A graph of vertex width ≤ 3 is guaranteed linearly reducible, and the zigzags admit an explicit width-three ordering, which is the one used to obtain the HyperFORM timings in Table 1.

Multicore scaling. FORM runs on multicore, multi-CPU machines, yet parallelizing symbolic manipulation is hard and a speed-up linear in the core count is essentially never reached. Running Z_6 on an AMD Ryzen 9 5900XT (16 physical cores, 3.3 GHz, 128 GB RAM) with one to sixteen FORM workers (Fig. 3), the CPU time falls substantially but sub-linearly and saturates, reflecting the overhead of distributing and collecting terms. Overclocking the same machine (Fig. 4), the CPU clock yields a nearly linear speed-up while the RAM frequency matters far less; the multilayered cache keeps the cores well fed.

6. Conclusions and outlook

To summarize, HyperFORM realizes hyperlogarithmic parametric integration, with rational prefactors allowed, inside FORM. It ports the HyperInt feature set to a faster host and supplies, as a modular, unit-tested library of some ninety procedures, the complete algorithmic chain: regularization in ϵ, passage to a fibration basis, the integration itself, regularized limits and MZV reduction. On the zigzag suite it runs about 3.5 times faster than HyperInt on a single core, with FORM's multicore execution adding further gains.

PoS(LL2026)069

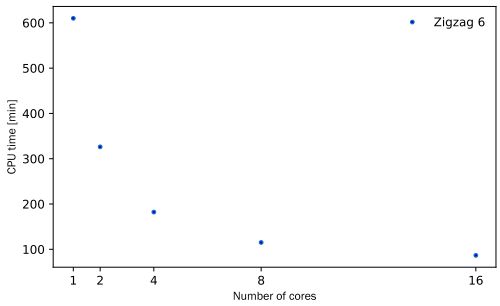

Figure 3: CPU time for Z_6 as a function of the number of FORM workers.

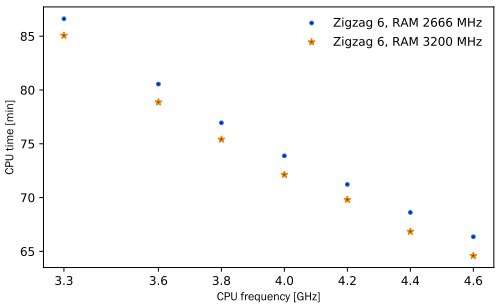

Figure 4: Computation time for Z_6 versus CPU clock frequency, for two RAM frequencies, using 16 cores.

This is a first release, with several improvements in view. Because polynomials are at present stored explicitly, term sizes are bounded, which caps the highest loop order reached at Z_6 for now; reworking this representation will remove the ceiling. Beyond that we plan to suppress spurious polynomials via the Fubini restriction and, more ambitiously on the mathematical side, to admit integrands that go past strict linear reducibility. On the regularization side, the third author has devised an alternative, more natural procedure for singular integrals that avoids the term proliferation inherent to the integration-by-parts method of Ref. [10]; incorporating it into HyperFORM is planned for a future release. The code is openly available [9], with the algorithms and a wider selection of examples documented in the full paper.

Acknowledgments

It is a pleasure to thank Francis Brown and Erik Panzer for numerous stimulating exchanges, and Jos Vermaseren together with the entire FORM development team for the tools that made this work possible. The research was supported in part by the ERC Advanced Grant 101095857 *Conformal-EIC* (A.K. and S.M.) and by DFG grant SCHN 1240/3-1 (O.S.). A.K. further thanks the EIC Theory Institute at Brookhaven National Laboratory, where part of this work was carried out, and acknowledges support from the University of Debrecen Program for Scientific Publication.

References

[1] S. Weinzierl, *Feynman Integrals. A Comprehensive Treatment for Students and Researchers*, UNITEXT for Physics, Springer (2022) [arXiv:2201.03593].
[2] O. Schnetz, ϕ^3 *theory at six loops*, *Phys. Rev. D* **112** (2025) 016028 [arXiv:2505.15485].
[3] O. Schnetz, ϕ^4 *theory at seven loops*, *Phys. Rev. D* **107** (2023) 036002 [arXiv:2212.03663].
[4] F. Brown, *The massless higher-loop two-point function*, *Commun. Math. Phys.* **287** (2009) 925 [arXiv:0804.1660].
[5] C. Bogner, *MPL – a program for computations with iterated integrals on moduli spaces of curves of genus zero*, *Comput. Phys. Commun.* **203** (2016) 339 [arXiv:1510.04562].

PoS(LL2026)069

[6] E. Panzer, *Algorithms for the symbolic integration of hyperlogarithms with applications to Feynman integrals*, *Comput. Phys. Commun.* **188** (2015) 148 [arXiv:1403.3385].
[7] M. Tentyukov and J. A. M. Vermaseren, *The multithreaded version of FORM*, *Comput. Phys. Commun.* **181** (2010) 1419 [arXiv:hep-ph/0702279].
[8] J. Davies, T. Kaneko, C. Marinissen, T. Ueda and J. A. M. Vermaseren, *FORM Version 5.0*, [arXiv:2601.19982].
[9] A. Kardos, S.-O. Moch and O. Schnetz, *HyperFORM v1.0*, Zenodo (2025), doi:10.5281/zenodo.17706909; *HyperFORM – a FORM package for parametric integration with hyperlogarithms* [arXiv:2511.19992].
[10] E. Panzer, *On hyperlogarithms and Feynman integrals with divergences and many scales*, *JHEP* **03** (2014) 071 [arXiv:1401.4361].
[11] A. B. Goncharov, *Multiple polylogarithms, cyclotomy and modular complexes*, *Math. Res. Lett.* **5** (1998) 497 [arXiv:1105.2076].
[12] K.-T. Chen, *Iterated path integrals*, *Bull. Am. Math. Soc.* **83** (1977) 831.
[13] J. Blümlein, D. J. Broadhurst and J. A. M. Vermaseren, *The multiple zeta value data mine*, *Comput. Phys. Commun.* **181** (2010) 582 [arXiv:0907.2557].
[14] The FLINT team, *FLINT: Fast Library for Number Theory*, version 3.3.1, https://flintlib.org (2025).
[15] F. Johansson, *Arb: efficient arbitrary-precision midpoint-radius interval arithmetic*, *IEEE Trans. Comput.* **66** (2017) 1281.
[16] A. Kardos and S.-O. Moch, *uf v1.0*, Zenodo (2025), doi:10.5281/zenodo.17646663.
[17] F. C. S. Brown, *On the periods of some Feynman integrals*, [arXiv:0910.0114].
[18] D. J. Broadhurst and D. Kreimer, *Knots and numbers in ϕ^4 theory to 7 loops and beyond*, *Int. J. Mod. Phys. C* **6** (1995) 519 [arXiv:hep-ph/9504352].
[19] F. Brown and O. Schnetz, *Single-valued multiple polylogarithms and a proof of the zig-zag conjecture*, *J. Number Theor.* **148** (2015) 478.
[20] S. E. Derkachev, A. I. Isaev and L. A. Shumilov, *Zig-zag diagrams and conformal triangles*, *Phys. Part. Nucl. Lett.* **20** (2023) 240.

PoS(LL2026)069