

## Computer Programs in Physics

## Tropical Feynman integration in the Minkowski regime ☆☆☆

Michael Borinsky<sup>a,\*</sup>, Henrik J. Munch<sup>b</sup>, Felix Tellander<sup>c</sup><sup>a</sup> Institute for Theoretical Studies, ETH Zürich, 8092 Zürich, Switzerland<sup>b</sup> Dipartimento di Fisica e Astronomia, Università degli Studi di Padova, 35131 Padova, Italy<sup>c</sup> Deutsches Elektronen-Synchrotron DESY, Notkestr. 85, 22607 Hamburg, Germany

## ARTICLE INFO

## Article history:

Received 1 March 2023

Received in revised form 5 June 2023

Accepted 27 July 2023

Available online 9 August 2023

Dataset link: <https://github.com/michibo/feyntrop>

## Keywords:

Feynman integrals

Monte Carlo integration

Tropical geometry

Epsilon-expansion

Contour deformation

Causal i-epsilon prescription

## ABSTRACT

We present a new computer program, *feyntrop*, which uses the tropical geometric approach to evaluate Feynman integrals numerically. In order to apply this approach in the physical regime, we introduce a new parametric representation of Feynman integrals that implements the causal  $i\epsilon$  prescription concretely while retaining projective invariance. *feyntrop* can efficiently evaluate dimensionally regulated, quasi-finite Feynman integrals, with not too exceptional kinematics in the physical regime, with a relatively large number of propagators and with arbitrarily many kinematic scales. We give a systematic classification of all relevant kinematic regimes, review the necessary mathematical details of the tropical Monte Carlo approach, give fast algorithms to evaluate (deformed) Feynman integrands, describe the usage of *feyntrop* and discuss many explicit examples of evaluated Feynman integrals.

## Program summary

Program title: *feyntrop*.CPC Library link to program files: <https://doi.org/10.17632/k6r62hdgvd.1>Developer's repository link: <https://github.com/michibo/feyntrop>.

Licensing provisions: MIT License.

Programming language: The tropical Monte Carlo code is written in C++. The high-level interface is written in python.

Supplementary material: The repository includes installation and usage instructions (README.md), a jupyter notebook tutorial (tutorial\_2L\_3pt.ipynb), the collection of examples presented in section 6 (see the folder /examples), and a test suite to ensure a successful installation (see the folder /tests).

Nature of problem: Sufficiently fast numerical integration of (dimensionally regularized) Feynman integrals (also in the Minkowski regime of phase space).

Solution method: Tropical Monte Carlo integration of a manifestly  $i\epsilon$ -free parametric representation of Feynman integrals.Additional comments: The program *feyntrop* is based on previous code available at <https://github.com/michibo/tropical-feynman-quadrature>, which was published as a proof-of-concept with, Michael Borinsky, 'Tropical Monte Carlo quadrature for Feynman integrals', *Ann. Inst. Henri Poincaré Comb. Phys. Interact.* (in press) [1]. This previous code did not have features which are required for phenomenological studies in high-energy physics. In particular, it only allowed for phase space points in the Euclidean regime, and only computed the leading term in the  $\epsilon$  expansion.

Restrictions: The Feynman integral must be quasi-finite and the momentum configuration must be sufficiently generic. Numerators of Feynman integrals are not implemented.

☆ The review of this paper was arranged by Prof. Z. Was.

☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (<http://www.sciencedirect.com/science/journal/00104655>).

\* Corresponding author.

E-mail address: [michael.borinsky@eth-its.ethz.ch](mailto:michael.borinsky@eth-its.ethz.ch) (M. Borinsky).

## References

Eigen3 [2]. The xoshiro256+ pseudo-random-number generator [3]. python [4]. pybind11 [5]. sympy [6].  
 © 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

## Contents

1. Introduction . . . . .	2
2. Feynman integrals . . . . .	4
2.1. Momentum and parametric representations . . . . .	4
2.2. Kinematic regimes . . . . .	4
2.3. Contour deformation . . . . .	5
2.4. Dimensional regularization and $\epsilon$ expansions . . . . .	7
3. Tropical geometry . . . . .	7
3.1. Tropical approximation . . . . .	7
3.2. Tropical sampling . . . . .	8
3.3. Base polytopes and generalized permutahedra . . . . .	8
3.4. Generalized permutahedral tropical sampling . . . . .	10
4. Numerical integration . . . . .	12
4.1. Monte Carlo integration . . . . .	12
4.2. Fast evaluation of (deformed) Feynman integrands . . . . .	12
5. The program <code>feynthrop</code> . . . . .	13
5.1. Installation . . . . .	13
5.2. Basic usage of <code>feynthrop</code> . . . . .	14
5.3. Deformation parameter . . . . .	16
6. Examples of Feynman integral evaluations . . . . .	16
6.1. A 5-loop 2-point zigzag diagram . . . . .	17
6.2. A 3-loop 4-point envelope diagram . . . . .	17
6.3. A 2-loop 4-point $\mu e$ -scattering diagram . . . . .	18
6.4. A QCD-like, 2-loop 5-point diagram . . . . .	19
6.5. Diagram contributing to triple Higgs production via gluon fusion . . . . .	19
6.6. A QED-like, 4-loop vacuum diagram . . . . .	20
6.7. An elliptic, conformal, 4-point integral . . . . .	21
7. Conclusions and outlook . . . . .	21
Declaration of competing interest . . . . .	22
Data availability . . . . .	22
Acknowledgements . . . . .	22
References . . . . .	22

## 1. Introduction

Feynman integrals are a key tool in quantum field theory. They are necessary to produce accurate predictions from given theoretical input such as a Lagrangian. Applications are, for instance, the computations of virtual contributions to scattering cross-sections for particle physics phenomenology [7], corrections to the magnetic moment of the muon or the half-life of positronium [8], critical exponents in statistical field theory [9] and corrections to the Newton potential due to general relativity [10]. An entirely mathematical application of Feynman integrals is the certification of cohomology classes in moduli spaces of curves or of graphs [11].

In this paper, we introduce `feynthrop`,<sup>1</sup> a new tool to evaluate Feynman integrals numerically. In contrast to existing tools, `feynthrop` can efficiently evaluate Feynman integrals with a relatively large number of propagators and with an arbitrary number of scales. Moreover, `feynthrop` can deal with Feynman integrals in the *physical* Minkowski regime and automatically takes care of the usually intricate contour deformation procedure. The spacetime dimension is completely arbitrary and integrals that are expanded in a dimensional regulator can be evaluated. The main restriction of `feynthrop` is that it cannot deal with Feynman integrals having subdivergences, that means the input Feynman integrals are required to be *quasi-finite*. Moreover, `feynthrop` is not designed to integrate Feynman integrals at certain highly exceptional kinematic points. Outside the Euclidean regime, the external kinematics are required to be sufficiently *generic*. It is worthwhile mentioning though that such highly exceptional kinematic points seem quite rare and `feynthrop` performs surprisingly well in these circumstances—in spite of the lack of mathematical guarantees for functioning. In fact, we were not able to find a quasi-finite integral with exceptional kinematics for which the integration with `feynthrop` fails. We only observed significantly decreased rates of convergence in such cases.

The mathematical theory of Feynman integrals has advanced rapidly in the last decades. Corner stone mathematical developments for Feynman integrals were, for instance, the systematic exploitation of their unitarity constraints (see, e.g., [12,13]), the systematic solution of their integration-by-parts identities (see, e.g., [14,15]), the application of modern algebraic geometric and number theoretic tools for

<sup>1</sup> `feynthrop` can be downloaded from <https://github.com/michibo/feynthrop>.

the benefit of their evaluation (see, e.g., [16–18]) and the systematic understanding of the differential equations which they fulfill (see, e.g., [19,20]).

Primarily, these theoretical developments were aimed at facilitating the *analytic* evaluation of Feynman integrals. All known analytic evaluation methods are inherently limited to a specific class of sufficiently simple diagrams. Especially for high-accuracy collider physics phenomenology, such analytic methods are often not sufficient to satisfy the demand for Feynman integral computations at higher loop order, which frequently involve complicated kinematics with many scales. Even if an analytic expression for a given Feynman integral is available, it is usually a highly non-trivial task to perform the necessary analytic continuation into the physical kinematic regime. On a different tack, computations of corrections to the Newton potential in the post-Newtonian expansion of general relativity [10] require the evaluation of large amounts of Feynman diagrams in three dimensional Euclidean space. As analytic evaluation is often more difficult in odd-dimensional spacetime, tropical Feynman integration is a promising candidate to fulfill the high demand for large loop order Feynman integrals in this field.

For this reason, numerical methods for the evaluation of Feynman integrals seem unavoidable once a certain threshold in precision has to be overcome. In this paper, we will use *tropical sampling* that was introduced in [1] to evaluate Feynman integrals numerically. This numerical integration technique is faster than traditional methods because the known (tropical) geometric structures of Feynman integrals are employed for the benefit of their numerical evaluation. For instance, general Euclidean Feynman integrals with up to 17 loops and 34 propagators can be evaluated using basic hardware with the proof-of-concept implementation that was distributed by the first author with [1]. The code of `feynthrop` is based on this implementation. The relevant mathematical structure is the *tropical geometry* of Feynman integrals in the parametric representation [21,1]. This tropical geometry itself is a simplification of the intricate *algebraic geometry* Feynman integrals display (see, e.g., [22]). Tropical Feynman integration was already used, for instance, in [23] to estimate the  $\phi^4$  theory  $\beta$  function up to loop order 11. Some ideas from [1] were already implemented in the FIESTA package [24]. Tropical sampling was extended to toric varieties with applications to Bayesian statistics [25]. Moreover, the tropical approach was recently applied to study infrared divergences of Feynman integrals in the Minkowski regime [26].

The tropical approach to Feynman integrals falls in line with the increasing number of fruitful applications of tools from convex geometry in the context of quantum field theory. These include, for example, the discovery of polytopes in amplitudes (see, e.g., [27,28]). Further, Feynman integrals can be seen as generalized Mellin-transformations [29–31]. As such they are solutions to GKZ-type differential equation systems [32]. Tropical and convex geometric tools are central to this analytic approach towards Feynman integrals (see, e.g., [33–37]).

Tropical Feynman integration is closely related to the *sector decomposition* approach [38–40], which applies to completely general algebraic integrals. State of the art implementations of sector decompositions are, for instance, `pySecDec` [41] and FIESTA [24]. Other numerical methods that are tailored specifically to Feynman integrals are, for instance, difference equations [15], unitarity methods [42], the Mellin-Barnes representation [43] and loop-tree duality [44,45]. With respect to potential applications to collider phenomenology, the latter three have the advantage of being inherently adapted to Minkowski spacetime kinematics. A newer technique is the systematic semi-numerical evaluation of Feynman integrals using differential equations [46,47], which is implemented, for instance, in `AMFlow` [48], `DiffExp` [49] and `SeaSyde` [50]. A similar semi-numerical approach was put forward in [51]. This technique can evaluate Feynman integrals quickly in the physical regime with high accuracy. A caveat is that it relies on the algebraic solution of the usually intricate integration-by-parts system associated to the respective Feynman integral and (usually) on analytic boundary values for the differential equations (see [48,52] for an exception where the boundary values are computed exclusively from algebraic input). We expect `feynthrop`, which does not rely on any analytic or algebraic input, to be useful for computing boundary values as input for such methods.

`feynthrop` uses the *parametric representation* of Feynman integrals for the numerical evaluation, which we briefly review in Section 2.1. This numerical evaluation has quite different characters in separate *kinematic regimes*. We propose a new classification of such kinematic regimes in Section 2.2 which, in addition to the usual Euclidean and Minkowski regimes, includes the intermediate *pseudo-Euclidean* regime. The original tropical Feynman integration implementation from [1] was limited to the Euclidean regime. Here, we achieve the extension of this approach to non-Euclidean regimes.

In the Minkowski regime, parametric Feynman integrands can have a complicated pole structure inside the integration domain. For the numerical integration an explicit *deformation* of the integration contour, which respects the desired causality properties, is needed. The use of explicit contour deformation prescriptions for numerics was pioneered in [42] and was later applied in the sector decomposition framework [53]. (Recently, a momentum space based approach for the solution of the deformation problem was put forward [54].) In Section 2.3, we propose an explicit deformation prescription which, in its basic form, was employed in [55] in the context of cohomological properties of Feynman integrals. This deformation prescription has the inherent advantage of retaining the projective symmetry of the parametric Feynman integrand. We provide explicit formulas for the Jacobian and thereby propose a new *deformed parametric representation* of the Feynman integral.

It is often desirable to evaluate a Feynman integral using dimensional regularization by adding a formal expansion parameter to the spacetime dimension, e.g.  $D = D_0 - 2\epsilon$ , where  $D_0$  is a fixed number and we wish to evaluate the Laurent or Taylor expansion of the integral in  $\epsilon$ . We will explain how `feynthrop` deals with such dimensionally regularized Feynman integrals in Section 2.4. Moreover, we will discuss one of the major limitations of `feynthrop` in this section: In its present form `feynthrop` can only integrate Feynman integrals that are *quasi-finite*. That means, input Feynman integrals are allowed to have an overall divergence, but no subdivergences. Further analytic continuation prescriptions (along the lines of [29,30,56]) would be needed to deal with such subdivergences and we postpone the implementation of such prescriptions into `feynthrop` to a future publication. For now, the user of the program is responsible to render all input integrals quasi-finite; for instance by projecting them to a quasi-finite basis [56]. Note, however, that within our approach, the base dimension  $D_0$  is completely arbitrary and can even be a non-integer value if desired. The applicability in the case  $D_0 = 3$  makes `feynthrop` a promising tool for the computation of post-Newtonian corrections to the gravitational potential [57].

In Sections 3.1 and 3.2, we will review the necessary ingredients for the tropical Monte Carlo approach from [1]: The concepts of the *tropical approximation* and *tropical sampling*. In Section 3.3, we review the (tropical) geometry of parametric Feynman integrands and the particular shape that the Symanzik polynomials' Newton polytopes exhibit. We will put special focus on the *generalized permutahedron* property of the second Symanzik  $\mathcal{F}$  polynomial. At particularly exceptional kinematic points, this property of the  $\mathcal{F}$  polynomial can be

lost. In these cases the integration with `feyntrop` might fail. We discuss this limitation in detail in Section 3.3. The overall tropical sampling algorithm is summarized in Section 3.4.

In Section 4.2, we summarize the necessary steps for the efficient evaluation of (deformed) parametric Feynman integrands. The key step is to express the entire integrand in terms of explicit matrix expressions. Our method is more efficient than the naive expansion of the Symanzik polynomials, as fast linear algebra routines can be used for the evaluation of such matrix expressions.

The structure, installation and usage of the program `feyntrop` is described in Section 5. To illustrate its capabilities we give multiple detailed examples of evaluated Feynman integrals in Section 6. In Section 7, we conclude and give pointers for further developments of the general tropical Feynman integration method and the program `feyntrop`.

## 2. Feynman integrals

### 2.1. Momentum and parametric representations

Let  $G$  be a one-particle irreducible Feynman graph with edge set  $E$  and vertex set  $V$ . Each edge  $e \in E$  comes with a mass  $m_e$  and an edge weight  $v_e$ . Each vertex  $v \in V$  comes with an incoming spacetime momentum  $p_v$ . Vertices without incoming momentum, i.e. where  $p_v = 0$ , are internal. Let  $\mathcal{E}$  be the incidence matrix of  $G$  which is formed by choosing an arbitrary orientation for the edges and setting  $\mathcal{E}_{v,e} = \pm 1$  if  $e$  points to/from  $v$  and  $\mathcal{E}_{v,e} = 0$  if  $e$  is not incident to  $v$ . The Feynman integral associated to  $G$  reads

$$\mathcal{I} = \int \prod_{e \in E} \frac{d^D q_e}{i\pi^{D/2}} \left( \frac{-1}{q_e^2 - m_e^2 + i\varepsilon} \right)^{v_e} \prod_{v \in V \setminus \{v_0\}} i\pi^{D/2} \delta^{(D)} \left( p_v + \sum_{e \in E} \mathcal{E}_{v,e} q_e \right), \quad (1)$$

where we integrate over all  $D$ -dimensional spacetime momenta  $q_e$  and we extracted the  $\delta$  function that accounts for overall momentum conservation by removing the vertex  $v_0 \in V$ . We compute the squared length  $q_e^2 = (q_e^0)^2 - (q_e^1)^2 - (q_e^2)^2 - \dots$  using the mostly-minus signature Minkowski metric.

To evaluate  $\mathcal{I}$  numerically, we will use the equivalent parametric representation (see, e.g., [58])

$$\mathcal{I} = \Gamma(\omega) \int_{\mathbb{P}_+^E} \phi \quad \text{with} \quad \phi = \left( \prod_{e \in E} \frac{x_e^{v_e}}{\Gamma(v_e)} \right) \frac{1}{\mathcal{U}(\mathbf{x})^{D/2}} \left( \frac{1}{\mathcal{V}(\mathbf{x}) - i\varepsilon \sum_{e \in E} x_e} \right)^\omega \Omega. \quad (2)$$

We integrate over the *positive projective simplex*  $\mathbb{P}_+^E = \{\mathbf{x} = [x_0, \dots, x_{|E|-1}] \in \mathbb{R} \mathbb{P}^{E-1} : x_e > 0\}$  with respect to its canonical volume form

$$\Omega = \sum_{e=0}^{|E|-1} (-1)^{|E|-e-1} \frac{dx_0}{x_0} \wedge \dots \wedge \frac{dx_e}{x_e} \wedge \dots \wedge \frac{dx_{|E|-1}}{x_{|E|-1}}. \quad (3)$$

Note that in the scope of this article we make the unusual choice to start the indexing with 0 for the benefit of a seamless notational transition to our computer implementation. So, the edge and vertex sets are always assumed to be given by  $E = \{0, 1, \dots, |E| - 1\}$  and  $V = \{0, 1, \dots, |V| - 1\}$ .

The *superficial degree of divergence* of the graph  $G$  is given by  $\omega = \sum_{e \in E} v_e - DL/2$ , where  $L = |E| - |V| + 1$  is the number of loops of  $G$ .

We use  $\mathcal{V}(\mathbf{x}) = \mathcal{F}(\mathbf{x})/\mathcal{U}(\mathbf{x})$  as a shorthand for the quotient of the two *Symanzik polynomials* that can be defined using the *reduced graph Laplacian*  $\mathcal{L}(\mathbf{x})$ , a  $(|V| - 1) \times (|V| - 1)$  matrix given element-wise by  $\mathcal{L}(\mathbf{x})_{u,v} = \sum_{e \in E} \mathcal{E}_{u,e} \mathcal{E}_{v,e} x_e$  for all  $u, v \in V \setminus \{v_0\}$ . We have the identities

$$\mathcal{U}(\mathbf{x}) = \det \mathcal{L}(\mathbf{x}) \left( \prod_{e \in E} x_e \right), \quad \mathcal{F}(\mathbf{x}) = \mathcal{U}(\mathbf{x}) \left( - \sum_{u,v \in V \setminus \{v_0\}} \mathcal{P}^{u,v} \mathcal{L}^{-1}(\mathbf{x})_{u,v} + \sum_{e \in E} m_e^2 x_e \right), \quad (4)$$

where  $\mathcal{P}^{u,v} = p_u \cdot p_v$  with the scalar product being computed using the Minkowski metric.

**Combinatorial Symanzik polynomials** We also have the combinatorial formulas for  $\mathcal{U}$  and  $\mathcal{F}$

$$\mathcal{U}(\mathbf{x}) = \sum_T \prod_{e \notin T} x_e, \quad \mathcal{F}(\mathbf{x}) = - \sum_F p(F)^2 \prod_{e \notin F} x_e + \mathcal{U}(\mathbf{x}) \sum_{e \in E} m_e^2 x_e, \quad (5)$$

where we sum over all spanning trees  $T$  and all spanning two-forests  $F$  of  $G$ , and  $p(F)^2$  is the Minkowski squared momentum running between the two-forest components. From this formulation it can be seen that  $\mathcal{U}$  and  $\mathcal{F}$  are homogeneous polynomials of degree  $L$  and  $L + 1$  respectively. Hence,  $\mathcal{V}$  is a homogeneous rational function of degree 1.

We will give fast algorithms to evaluate  $\mathcal{U}(\mathbf{x})$  and  $\mathcal{F}(\mathbf{x})$  in Section 4.2.

### 2.2. Kinematic regimes

By Poincaré invariance, the value of the Feynman integral (1) only depends on the  $|V| \times |V|$  Gram matrix  $\mathcal{P}^{u,v} = p_u \cdot p_v$  and not on the explicit form of the vectors  $p_v$ . In fact, it is even irrelevant in which ambient dimension the vectors  $p_v$  are defined. The following characterization of the different *kinematic regimes* that we propose will therefore only take the input of a symmetric  $|V| \times |V|$  matrix  $\mathcal{P}$  with vanishing row and column sums (i.e. the momentum conservation conditions  $\sum_{v \in V} p_u \cdot p_v = \sum_{v \in V} \mathcal{P}^{u,v} = 0$  for all  $u \in V$ ), without requiring any explicit knowledge of the  $p_v$  vectors. In fact, we will not even require that there are any vectors  $p_v$  for which  $\mathcal{P}^{u,v} = p_u \cdot p_v$ .

**Euclidean regime** We say a given Feynman integral computation problem is in the *Euclidean regime* if the matrix  $\mathcal{P}$  is negative semi-definite. In this regime,  $\mathcal{F}(\mathbf{x}) \geq 0$  for all  $\mathbf{x} \in \mathbb{P}_+^E$ . We call this the Euclidean regime, because the integral (1) is equivalent to an analogous Feynman integral where scalar products are computed with the Euclidean all-minus metric. To see this, note that as  $-\mathcal{P}$  is positive semi-definite, there is a  $|V| \times |V|$  matrix  $\mathcal{Q}$  such that  $\mathcal{P} = -\mathcal{Q}^T \mathcal{Q}$ . We can think of the column vectors  $\tilde{p}_1, \dots, \tilde{p}_{|V|}$  of  $\mathcal{Q}$  as an auxiliary set of incoming momentum vectors. Elements of  $\mathcal{P}$  can be interpreted as Euclidean, all-minus metric, scalar products of the  $\tilde{p}_v$ -vectors:  $\mathcal{P}^{u,v} = -\tilde{p}_u^T \tilde{p}_v = -\sum_{w \in V} \mathcal{Q}^{w,u} \mathcal{Q}^{w,v}$ . Translating this back to (1) means that we can change the signature of the scalar products to the all-minus metric if we replace the external momenta with the  $\tilde{p}_v$  vectors which are defined in an auxiliary space  $\mathbb{R}^{|V|}$ . We emphasize that this way of relating Euclidean and Minkowski space integrals is inherently different from the typical *Wick rotation* procedure and that the  $\tilde{p}_v$ -vectors will in general be different from the original  $p_v$  vectors.

**Pseudo-Euclidean regime** In fact,  $\mathcal{F}(\mathbf{x}) \geq 0$  for all  $\mathbf{x} \in \mathbb{P}_+^E$  in a larger kinematic regime, where  $\mathcal{P}$  is not necessarily negative semi-definite. If for each subset  $V' \subset V$  of the vertices the inequality

$$\left( \sum_{v \in V'} p_v \right)^2 = \sum_{u,v \in V'} p_u \cdot p_v = \sum_{u,v \in V'} \mathcal{P}^{u,v} \leq 0 \quad (6)$$

is respected, then we are in the *pseudo-Euclidean regime*. The first two equalities in (6) are only included as mnemonic devices; knowledge of  $\mathcal{P}$  is sufficient to check the inequalities. Equivalently, we can require the element sums of all *principle minor matrices* of the  $\mathcal{P}$  matrix to be  $\leq 0$ .

By (5) and (6), the coefficients of  $\mathcal{F}$  are non-negative in the pseudo-Euclidean regime. Our choice of normalization factors ensures that (1) and (2) are real positive in this case.

We remark that there is a commonly used alternative definition of a kinematic regime which, on first sight, is similar to the condition above. This alternative definition requires the inequalities  $p_u \cdot p_v \leq 0$  to be fulfilled for all  $u, v \in V$  (see, e.g., [59, Sec. 2.5]). This is more restrictive than our condition in (6). In fact, it is too restrictive for our purposes, as not even entirely *Euclidean Feynman integrals* can generally be described in this regime. The reason for this is that not all negative semi-definite matrices  $\mathcal{P}$  fulfill this more restrictive condition.

In our case, the Euclidean regime is contained in the pseudo-Euclidean regime. To verify this, we have to make sure that a negative semi-definite  $\mathcal{P}$  fulfills the conditions in (6). Such a  $\mathcal{P}$  can be represented with an appropriate set of  $\tilde{p}_v$  vectors as above:  $\mathcal{P}^{u,v} = -\tilde{p}_u^T \tilde{p}_v$ . For each  $V' \subset V$  we get the principle minor element sum

$$\sum_{u,v \in V'} \mathcal{P}^{u,v} = - \sum_{u,v \in V'} \tilde{p}_u^T \tilde{p}_v = - \left( \sum_{v \in V'} \tilde{p}_v \right)^T \left( \sum_{v \in V'} \tilde{p}_v \right) \leq 0. \quad (7)$$

**Minkowski regime** If we are not in the pseudo-Euclidean regime (and thereby also not in the Euclidean regime), then we are in the *Minkowski regime*.

**Generic and exceptional kinematics** Without any resort to the explicit incoming momentum vectors  $p_v$ , we call a vertex  $v$  *internal* if  $\mathcal{P}^{u,v} = 0$  for all  $u \in V$  and *external* otherwise. Let  $V^{\text{ext}} \subset V$  be the set of external vertices. Complementary to the classification above, we say that our kinematics are *generic* if for each *proper* subset  $V' \subsetneq V^{\text{ext}}$  of the external vertices of  $G$  and for each non-empty subset  $E' \subset E$  of the edges of  $G$  we have

$$\left( \sum_{v \in V'} p_v \right)^2 = \sum_{u,v \in V'} p_u \cdot p_v = \sum_{u,v \in V'} \mathcal{P}^{u,v} \neq \sum_{e \in E'} m_e^2. \quad (8)$$

For example, the kinematics are always generic in the pseudo-Euclidean regime if  $m_e > 0$  for all  $e \in E$  or if  $\sum_{u,v \in V'} \mathcal{P}^{u,v} < 0$  for all  $V' \subsetneq V^{\text{ext}}$ . Note that generic kinematics also exclude *on-shell external momenta*, i.e. cases where  $p_v^2 = \mathcal{P}^{v,v} = 0$  for some  $v \in V^{\text{ext}}$  as long as not all  $m_e > 0$ , for then there exists at least one edge  $e \in E$  such that  $p_v^2 = 0 = m_e^2$ , thus violating (8). Genericity, for instance, guarantees that there will be no cancellation between the momentum and the mass part of the  $\mathcal{F}$ -polynomial as defined in (5).

Kinematic configurations that are not generic are called *exceptional*.

As above, only the statements on  $\mathcal{P}^{u,v}$  are sufficient for the classification. The other equalities are added to enable a seamless comparison to the literature.

The discussed kinematic regimes and their respective overlaps are illustrated in Fig. 1. In contrast to what the figure might suggest, the exceptional kinematics only cover a space that is of lower dimension than the one of the generic regime. The Minkowski regime is not explicitly shown as it covers the whole area that is not pseudo-Euclidean. Note that Minkowski, pseudo-Euclidean and Euclidean kinematics can be exceptional.

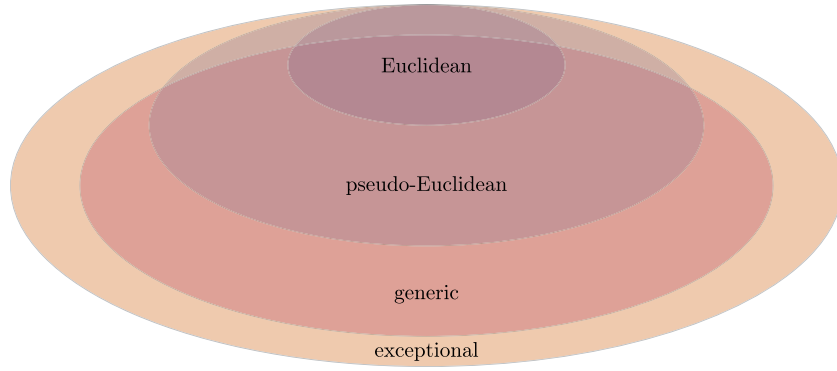
`feyntrap` detects the relevant kinematic regime using the conditions discussed above.

### 2.3. Contour deformation

In the pseudo-Euclidean (and thereby also in the Euclidean) regime,  $\mathcal{F}(\mathbf{x})$  stays positive and the integral (2) cannot have any simple poles inside the integration domain.

In the Minkowski regime however, simple propagator poles of the integrand (1) and simple poles associated to zeros of  $\mathcal{F}$  in (2) are avoided using the causal  $i\epsilon$  prescription (see, e.g., [60]). This prescription tells us to which side of the pole the integration contour needs to be deformed. When evaluating integrals such as (1) numerically, we have to find an *explicit* choice for such an integration





**Fig. 1.** Partition of kinematics into different regimes.

contour. Finding such an explicit contour deformation, which also has decent numerical stability properties, is a surprisingly complicated task. Explicit contour deformations for numerical evaluation were pioneered by Soper [42] and later refined [53,61]. This original type of contour deformation has the caveat that the projective symmetry of the integral (2) is lost as these deformations are inherently non-projective and usually formulated in affine charts, i.e. ‘gauge fixed’ formulations of (2). Experience, e.g. from [1], shows that the projective symmetry of (2) is a treasured good that should not be given up lightly.

To retain projective symmetry we will hence use a different deformation than established numerical integration tools. We will use the embedding  $\iota_\lambda : \mathbb{P}_+^E \hookrightarrow \mathbb{C}\mathbb{P}^{|E|-1}$  (recall that  $\mathbb{P}_+^E$  is a subset of  $\mathbb{R}\mathbb{P}^{|E|-1}$ ) of the projective simplex into  $|E| - 1$  complex dimensional projective space given by

$$\iota_\lambda : x_e \mapsto x_e \exp\left(-i\lambda \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x})\right). \quad (9)$$

This deformation prescription was proposed in [55, eq. (43)] in the context of the cohomological viewpoint on Feynman integrals (see also [62, Sec. 4.3]). As  $\mathcal{U}$  and  $\mathcal{F}$  are homogeneous polynomials of degree  $L$  and  $L + 1$  respectively and  $\mathcal{V}(\mathbf{x}) = \mathcal{F}(\mathbf{x})/\mathcal{U}(\mathbf{x})$ , the partial derivative  $\frac{\partial \mathcal{V}}{\partial x_e}$  is a rational function in  $\mathbf{x}$  of homogeneous degree 0, so  $\iota_\lambda$  indeed respects projective equivalence.

We want to deform the integration contour  $\mathbb{P}_+^E$  of (2) into  $\iota_\lambda(\mathbb{P}_+^E) \subset \mathbb{C}\mathbb{P}^{|E|-1}$ . The deformation  $\iota_\lambda$  does not change the boundary of  $\mathbb{P}_+^E$  as each boundary face of  $\mathbb{P}_+^E$  is characterized by at least one vanishing homogeneous coordinate  $x_e = 0$ . So,  $\iota_\lambda(\partial \mathbb{P}_+^E) = \partial \mathbb{P}_+^E$ . By Cauchy’s theorem, we can deform the contour as long as we do not hit any poles of the integrand  $\phi$ . Supposing that  $\lambda$  is small enough such that no poles of  $\phi$  are hit by the deformation, we have

$$\mathcal{I} = \Gamma(\omega) \int_{\iota_\lambda(\mathbb{P}_+^E)} \phi = \Gamma(\omega) \int_{\mathbb{P}_+^E} \iota_\lambda^* \phi, \quad (10)$$

where  $\iota_\lambda^* \phi$  denotes the *pullback* of the differential form  $\phi$ . A computation on forms reveals that  $\iota_\lambda^* \Omega = \det(\mathcal{J}_\lambda(\mathbf{x})) \Omega$ , where the Jacobian  $\mathcal{J}_\lambda(\mathbf{x})$  is the  $|E| \times |E|$  matrix given element-wise by

$$\mathcal{J}_\lambda(\mathbf{x})^{e,h} = \delta_{e,h} - i\lambda x_e \frac{\partial^2 \mathcal{V}}{\partial x_e \partial x_h}(\mathbf{x}) \text{ for all } e, h \in E. \quad (11)$$

Thus, we arrive at the desired *deformed parametric Feynman integral* by making (10) explicit,

$$\mathcal{I} = \Gamma(\omega) \int_{\mathbb{P}_+^E} \iota_\lambda^* \phi = \Gamma(\omega) \int_{\mathbb{P}_+^E} \left( \prod_{e \in E} \frac{X_e^{v_e}}{\Gamma(v_e)} \right) \frac{\det \mathcal{J}_\lambda(\mathbf{x})}{\mathcal{U}(\mathbf{X})^{D/2} \cdot \mathcal{V}(\mathbf{X})^\omega} \Omega, \quad (12)$$

where  $\mathbf{X} = \iota_\lambda(\mathbf{x})$ , that means  $\mathbf{X} = (X_0, \dots, X_{|E|-1})$  and  $X_e = x_e \exp(-i\lambda \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}))$  for all  $e \in E$ .

Although the prescription (9) was proposed before in a more formal context, the deformed formulation of the parametric Feynman integral (12) with the explicit Jacobian factor given by (11) appears not to have been considered previously in the literature.

In Section 4.2, we provide fast algorithms and formulas to evaluate  $\frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x})$  and  $X_e$ .

**Landau singularities** In the formulation (12), the  $i\epsilon$  prescription is taken care of by the deformation of the rational function  $\mathcal{V}$ . To see this, consider the Taylor expansion of  $\mathcal{V}(\mathbf{X})$  in  $\lambda$ ,

$$\mathcal{V}(\mathbf{X}) = \mathcal{V}(\mathbf{x}) - i\lambda \sum_{e \in E} x_e \left( \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}) \right)^2 + \mathcal{O}(\lambda^2). \quad (13)$$

The  $i\epsilon$  prescription in (2) is ensured if the imaginary part of  $\mathcal{V}(\mathbf{X})$  is strictly negative for *sufficiently small*  $\lambda$ . This is the case for all  $\mathbf{x} \in \mathbb{P}_+^E$  as long as there are no solutions of the *Landau equations*

$$0 = x_e \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}) \quad \text{for each } e \in E, \quad \text{for any } \mathbf{x} \in \mathbb{P}_+^E, \quad (14)$$

whose solutions are the *Landau singularities*. We will assume that our Feynman integral is always free of Landau singularities.

Even though we require that  $\lambda$  is *small enough*, we can give it, in contrast to the  $\varepsilon$  in (2), an explicit *finite* value. Hence, eq. (12) is finally an explicit form of the original Feynman integral (1) that is going to serve as input for the tropical numerical integration algorithm.

## 2.4. Dimensional regularization and $\epsilon$ expansions

So far, we did not make any restrictions on the finiteness properties of the integrals (1), (2) and (12). We say a Feynman integral is *quasi-finite* if the integral in the parametric representation (2) (or equivalently (12)) is finite. Only the integral needs to be finite. The  $\Gamma$  function prefactor is allowed to give divergent contributions. Note that this is more permissive than requiring that (1) is finite, which is already divergent, e.g., for the 1-loop bubble in  $D = 4$  with unit edge weights.

In this paper, we will restrict our attention to such quasi-finite Feynman integrals. If an integral is not quasi-finite, it can be expanded as a linear combination of quasi-finite integrals [29,30,56].

Quasi-finiteness allows *overall* divergences due to the  $\Gamma(\omega)$  factor that becomes singular if  $\omega$  is an integer  $\leq 0$ . Such divergences are easily taken care of by using dimensional regularization. As usual we will perturb the dimension by  $\epsilon$  in the sense that

$$D = D_0 - 2\epsilon, \quad (15)$$

where  $D_0$  is a fixed number and  $\epsilon$  is an expansion parameter.<sup>2</sup> Analogously, we define  $\omega_0 = \sum_{e \in E} v_e - D_0 L/2$ . Using this notation, we may make the  $\epsilon$  dependence in (12) explicit and expand,

$$\mathcal{I} = \Gamma(\omega_0 + \epsilon L) \sum_{k=0}^{\infty} \frac{\epsilon^k}{k!} \int_{\mathbb{P}_+^E} \left( \prod_{e \in E} \frac{X_e^{v_e}}{\Gamma(v_e)} \right) \frac{\det \mathcal{J}_\lambda(\mathbf{x})}{\mathcal{U}(\mathbf{x})^{D_0/2} \cdot \mathcal{V}(\mathbf{x})^{\omega_0}} \log^k \left( \frac{\mathcal{U}(\mathbf{x})}{\mathcal{V}(\mathbf{x})^L} \right) \Omega. \quad (16)$$

If the  $k=0$  integral is finite, all higher orders in  $\epsilon$  are also finite as the  $\log^k$  factors cannot spoil the integrability. The  $\Gamma$  factor can be expanded in  $\epsilon$  using  $\Gamma(z+1) = z\Gamma(z)$  and the expansion

$$\log \Gamma(1 - \epsilon) = \gamma_E \epsilon + \sum_{n=2}^{\infty} \frac{\zeta(n)}{n} \epsilon^n, \quad (17)$$

with Euler's  $\gamma_E$  and Riemann's  $\zeta$  function.

Together, eqs. (16) and (17) give us an explicit formulation of the  $\epsilon$  expansion of the Feynman integral (1) in the quasi-finite case. In the remainder of this article we will explain how to evaluate the expansion coefficients in (16) using the tropical sampling approach.

## 3. Tropical geometry

### 3.1. Tropical approximation

We will use the tropical sampling approach which was put forward in [1] to evaluate the deformed parametric Feynman integrals in (12) and (16). Here we briefly review the basic concepts.

For any homogeneous polynomial in  $|E|$  variables  $p(\mathbf{x}) = \sum_{k \in \text{supp}(p)} a_k \prod_{e=0}^{|E|-1} x_e^{k_e}$ , the support  $\text{supp}(p)$  is the set of multi-indices for which  $p$  has a non-zero coefficient  $a_k$ . For any such polynomial  $p$ , we define the *tropical approximation*  $p^{\text{tr}}$  as

$$p^{\text{tr}}(\mathbf{x}) = \max_{k \in \text{supp}(p)} \prod_{e=0}^{|E|-1} x_e^{k_e}. \quad (18)$$

If, for example,  $p(\mathbf{x}) = x_0^2 x_1 - 2x_0 x_1 x_2 + 5ix_2^3$ , then  $p^{\text{tr}}(\mathbf{x}) = \max\{x_0^2 x_1, x_0 x_1 x_2, x_2^3\}$ . Note that the tropical approximation forgets about the explicit value of the coefficients; it only depends on the fact that a specific coefficient is zero or non-zero. This way, the tropical approximation only depends on the set  $\text{supp}(p) \subset \mathbb{Z}_{\geq 0}^{|E|}$ . In fact, it only depends on the shape of the *convex hull* of  $\text{supp}(p)$ , which is the *Newton polytope* of  $p$ . For this reason,  $p^{\text{tr}}$  is nothing but a function avatar of this polytope. Indeed, we can write  $p^{\text{tr}}(\mathbf{x})$  as follows,

$$p^{\text{tr}}(\mathbf{x}) = \exp \left( \max_{\mathbf{v} \in \mathbf{N}[p]} \mathbf{v}^T \mathbf{y} \right), \quad (19)$$

where  $\mathbf{y} = (y_0, \dots, y_{|E|-1})$  with  $y_e = \log x_e$ ,  $\mathbf{v}^T \mathbf{y} = \sum_{e \in E} v_e y_e$  and we maximize over the Newton polytope  $\mathbf{N}[p]$  of  $p$ . The exponent above is the *tropicalization*  $\text{Trop}[p]$  of  $p$  over  $\mathbb{C}$  with trivial valuation. It plays a central role in tropical geometry (see, e.g., [63]). For us, the key property of the tropical approximation is that it may be used to put upper and lower bounds on a polynomial:

**Theorem 3.1** ([1, Theorem 8]). *For a homogeneous  $p \in \mathbb{C}[x_0, \dots, x_{|E|-1}]$  that is completely non-vanishing on  $\mathbb{P}_+^E$  there exist constants  $C_1, C_2 > 0$  such that*

$$C_1 \leq \frac{|p(\mathbf{x})|}{p^{\text{tr}}(\mathbf{x})} \leq C_2 \quad \text{for all } \mathbf{x} \in \mathbb{P}_+^E. \quad (20)$$

<sup>2</sup> Note that the causal  $i\varepsilon$  and the regularization/expansion parameter  $\epsilon$  are (unfortunately) usually referred to with the same Greek letter. We will follow this tradition, but use different versions of the letter for the respective meanings consistently.

A polynomial  $p$  is *completely non-vanishing* on  $\mathbb{P}_+^E$  if it does not vanish in the interior of  $\mathbb{P}_+^E$  and if another technical condition is fulfilled (see [29, Definition 1] for a precise definition).

The  $\mathcal{U}$  polynomial is always completely non-vanishing on  $\mathbb{P}_+^E$  and in the pseudo-Euclidean regime also  $\mathcal{F}$  is completely non-vanishing on  $\mathbb{P}_+^E$ . We define the associated tropical approximations  $\mathcal{U}^{\text{tr}}$ ,  $\mathcal{F}^{\text{tr}}$  and  $\mathcal{V}^{\text{tr}} = \mathcal{F}^{\text{tr}}/\mathcal{U}^{\text{tr}}$ .

Our key assumption for the integration of Feynman integrals in the Minkowski regime is that the approximation property can also be applied to the deformed Symanzik polynomials.

**Assumption 3.2.** There are  $\lambda$  dependent constants  $C_1(\lambda), C_2(\lambda) > 0$  such that for small  $\lambda > 0$ ,

$$C_1(\lambda) \leq \left| \left( \frac{\mathcal{U}^{\text{tr}}(\mathbf{x})}{\mathcal{U}(\mathbf{X})} \right)^{D_0/2} \left( \frac{\mathcal{V}^{\text{tr}}(\mathbf{x})}{\mathcal{V}(\mathbf{X})} \right)^{\omega_0} \right| \leq C_2(\lambda) \quad \text{for all } \mathbf{x} \in \mathbb{P}_+^E, \quad (21)$$

where we recall that  $\mathbf{X} = (X_1, \dots, X_{|E|})$  and  $X_e = x_e \exp(-i\lambda \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}))$ .

In the pseudo-Euclidean regime the assumption is fulfilled, as we are allowed to set  $\lambda = 0$  and use the established approximation property from [1] on  $\mathcal{U}$  and  $\mathcal{F}$ . In the Minkowski regime, Assumption 3.2 can only be fulfilled if there are no Landau singularities, i.e. solutions to (14). After extensive numerical testing we conjecture that Assumption 3.2 is fulfilled if there are no Landau singularities. It would be very interesting to give a concise set of conditions for the validity of Assumption 3.2 and how it interplays with such singularities. We leave this to future research.

Another highly promising research question is to find a value for  $\lambda$  such that the constants  $C_1(\lambda)$  and  $C_2(\lambda)$  tighten the bounds as much as possible. Finding such an optimal value for  $\lambda$  would result in the first entirely *canonical* deformation prescription which does not depend on free parameters.

### 3.2. Tropical sampling

Intuitively, Assumption 3.2 tells us that the integrands in (12) and (16) are, except for phase factors, reasonably approximated by the tropical approximation of the undeformed integrand. To evaluate the integrals (16) with *tropical sampling*, as in [1, Sec. 7.2], we define the probability distribution

$$\mu^{\text{tr}} = \frac{1}{I^{\text{tr}}} \frac{\prod_{e \in E} x_e^{\nu_e}}{\mathcal{U}^{\text{tr}}(\mathbf{x})^{D_0/2} \mathcal{V}^{\text{tr}}(\mathbf{x})^{\omega_0}} \Omega, \quad (22)$$

where  $I^{\text{tr}}$  is a normalization factor, which is chosen such that  $\int_{\mathbb{P}_+^E} \mu^{\text{tr}} = 1$ . As of Assumption 3.2 and the requirement that the integrals in (16) shall be finite, the factor  $I^{\text{tr}}$  must also be finite. If  $\omega_0 = 0$ , this normalization factor is equal to the associated *Hepp bound* of the graph  $G$  [21]. Because  $\mu^{\text{tr}} > 0$  for all  $\mathbf{x} \in \mathbb{P}_+^E$ ,  $\mu^{\text{tr}}$  gives rise to a proper probability distribution on this domain.

Using the definition of  $\mu^{\text{tr}}$  to rewrite (16) results in

$$\begin{aligned} \mathcal{I} &= \frac{\Gamma(\omega_0 + \epsilon L)}{\prod_{e \in E} \Gamma(\nu_e)} \sum_{k=0}^{\infty} \frac{\epsilon^k}{k!} \mathcal{I}_k, \quad \text{with} \\ \mathcal{I}_k &= I^{\text{tr}} \int_{\mathbb{P}_+^E} \frac{(\prod_{e \in E} (X_e/x_e)^{\nu_e}) \det \mathcal{J}_\lambda(\mathbf{x})}{(\mathcal{U}(\mathbf{X})/\mathcal{U}^{\text{tr}}(\mathbf{x}))^{D_0/2} \cdot (\mathcal{V}(\mathbf{X})/\mathcal{V}^{\text{tr}}(\mathbf{x}))^{\omega_0}} \log^k \left( \frac{\mathcal{U}(\mathbf{X})}{\mathcal{V}(\mathbf{X})^L} \right) \mu^{\text{tr}}. \end{aligned} \quad (23)$$

We will evaluate the integrals above by sampling from the probability distribution  $\mu^{\text{tr}}$ .

In [1], two different methods to generate samples from  $\mu^{\text{tr}}$  were introduced. The first method [1, Sec. 5], which does not take the explicit structure of  $\mathcal{U}$  and  $\mathcal{F}$  into account, requires the computation of a triangulation of the refined normal fans of the Newton polytopes of  $\mathcal{U}$  and  $\mathcal{F}$ . Once such a triangulation is computed, arbitrarily many samples from  $\mu^{\text{tr}}$  can be generated with little computational effort. Unfortunately, obtaining such a triangulation is a highly computationally demanding process.

The second method [1, Sec. 6] to generate samples from the probability distribution  $\mu^{\text{tr}}$  makes use of a particular property of the Newton polytopes of  $\mathcal{U}$  and  $\mathcal{F}$  which allows to bypass the costly triangulation step. This second method additionally has the advantage that it is relatively straightforward to implement. This faster method of sampling from  $\mu^{\text{tr}}$  relies on the Newton polytopes of  $\mathcal{U}$  and  $\mathcal{F}$  being *generalized permutahedra*.

For the program `feyntrop` we will make use of this second method. Our tropical sampling algorithm to produce samples from  $\mu^{\text{tr}}$  is essentially equivalent to the one published with [1].

### 3.3. Base polytopes and generalized permutahedra

A fantastic property of generalized permutahedra is that they come with a *canonical normal fan* which greatly facilitates the sampling of  $\mu^{\text{tr}}$ , see [1, Theorem 27 and Algorithm 4]. Here, we briefly explain the necessary notions. As a start, we define a more general class of polytopes first and discuss restrictions later.

**Base polytopes** Consider a function  $z: 2^E \rightarrow \mathbb{R}$  that assigns a number to each subset of  $E$ , the edge set of our Feynman graph  $G$ . In the following we often identify a subset of  $E$  with a subgraph of  $G$  and use the respective terms interchangeably. So,  $z$  assigns a number to each subgraph of  $G$ . We define  $\mathbf{P}[z]$  to be the subset of  $\mathbb{R}^{|E|}$  that consists of all points  $(a_0, \dots, a_{|E|-1}) \in \mathbb{R}^{|E|}$  which fulfill  $\sum_{e \in E} a_e = z(E)$  and the  $2^{|E|} - 1$  inequalities



$$\sum_{e \in \gamma} a_e \geq z(\gamma) \quad \text{for all } \gamma \subsetneq E. \quad (24)$$

Clearly, these inequalities describe a convex bounded domain, i.e. a *polytope*. This polytope  $\mathbf{P}[z]$  associated to an arbitrary function  $z : 2^E \rightarrow \mathbb{R}$  is called the *base polytope*.

**Generalized permutahedra** The following is a special case of a theorem by Aguiar and Ardila who realized that numerous seemingly different structures from combinatorics can be understood using the same object: The *generalized permutahedron* which was initially defined by Postnikov [64].

**Theorem 3.3** ([65, Theorem 12.3] and the references therein). *The polytope  $\mathbf{P}[z]$  is a generalized permutahedron if and only if the function  $z$  is supermodular. That means,  $z$  fulfills the inequalities*

$$z(\gamma) + z(\delta) \leq z(\gamma \cup \delta) + z(\gamma \cap \delta) \text{ for all pairs of subgraphs } \gamma, \delta \subset E. \quad (25)$$

Because other properties of generalized permutahedra are not of central interest in this paper, we will take Theorem 3.3 as our definition of these special polytopes. Important for us is that for many kinematic situations the Newton polytopes of the Symanzik polynomials are of this type.

Let  $L_\gamma$  denote the number of loops of the subgraph  $\gamma$ , then we have the following theorem due to Schultka [31]:

**Theorem 3.4.** *The Newton polytope  $\mathbf{N}[\mathcal{U}]$  of  $\mathcal{U}$  is equal to the base polytope  $\mathbf{P}[z_{\mathcal{U}}]$  with  $z_{\mathcal{U}}$  being the function  $z_{\mathcal{U}}(\gamma) = L_\gamma$ . Moreover,  $z_{\mathcal{U}}$  is supermodular. Hence, by Theorem 3.3,  $\mathbf{N}[\mathcal{U}]$  is a generalized permutahedron.*

**Proof.** See [31, Sec. 4] and the references therein. In [21], it was observed that  $\mathbf{N}[\mathcal{U}]$  is a *matroid polytope*, which by [65, Sec. 14] also proves the statement.  $\square$

Because  $\mathbf{N}[\mathcal{U}]$  is a generalized permutahedron, we also say that  $\mathcal{U}$  has the generalized permutahedron property.

**Generalized permutahedron property of the  $\mathcal{F}$  polynomial** For the second Symanzik  $\mathcal{F}$  polynomial the situation is more tricky. We need the notion of *mass-momentum spanning* subgraphs which was defined by Brown [22] (see also [31, Sec. 4] for an interesting relationship to the concept of *s-irreducibility* [66] or [67] where related results were obtained or [68] for relations to the  $R^*$  operation). We use the following slightly generalized version of Brown's definition (see also [1, Sec. 7.2]): We call a subgraph  $\gamma \subset E$  mass-momentum spanning if the second Symanzik polynomial of the cograph  $G/\gamma$  vanishes identically  $\mathcal{F}_{G/\gamma} = 0$ .

**Theorem 3.5.** *In the Euclidean regime with generic kinematics, the Newton polytope  $\mathbf{N}[\mathcal{F}]$  is a generalized permutahedron. It is equal to the base polytope  $\mathbf{P}[z_{\mathcal{F}}]$  with the function  $z_{\mathcal{F}}$  defined for all subgraphs  $\gamma$  by  $z_{\mathcal{F}}(\gamma) = L_\gamma + 1$  if  $\gamma$  is mass-momentum spanning and  $z_{\mathcal{F}}(\gamma) = L_\gamma$  otherwise. Consequently, this function  $z_{\mathcal{F}} : 2^E \rightarrow \mathbb{R}$  is supermodular, i.e. it fulfills (25).*

**Proof.** This has also been proven in [31, Sec. 4]. The proof relies on a special infrared factorization property of  $\mathcal{F}$  that was discovered by Brown [22, Theorem 2.7].  $\square$

We explicitly state the following generalization of Theorem 3.5:

**Theorem 3.6.** *Theorem 3.5 holds in all regimes if the kinematics are generic.*

**Proof.** The  $\mathcal{F}$  polynomial has the same monomials (with different coefficients) as in the Euclidean regime with generic kinematics. To verify this, note that the conditions for generic kinematics prevent cancellations between the mass and momentum part of the  $\mathcal{F}$  polynomial as given in eq. (5). So, the respective Newton polytopes coincide.  $\square$

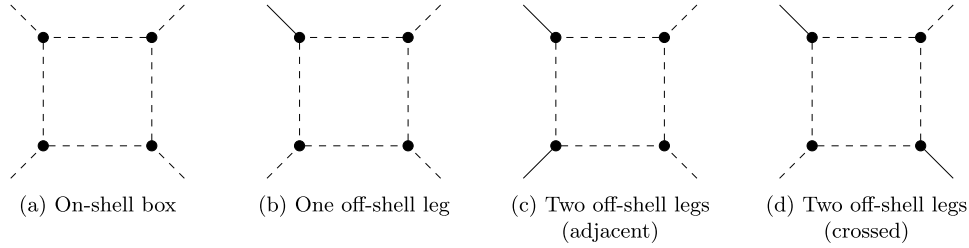
There is also the following further generalization of Theorem 3.5 to Euclidean but exceptional kinematics. This generalization is very plausible (see [22, Example 2.5]), but it is a technical challenge to prove it. We will not attempt to include a proof here for the sake of brevity. So, we state this generalization as a conjecture:

**Conjecture 3.7.** *Theorem 3.5 holds in the Euclidean regime for all (also exceptional) kinematics.*

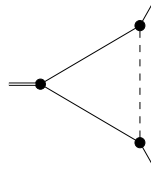
We emphasize that  $\mathbf{N}[\mathcal{F}]$  is generally *not* a generalized permutahedron outside of the Euclidean regime. This was observed in [31, Remark 4.16] (see also [69, Sec. 4.2], [70, Sec. 2.2.3] or [1, Remark 35]). Explicit counter examples are encountered while computing the massless on-shell boxes depicted in Fig. 2. The  $\mathcal{F}$  polynomials of the completely massless box with only on-shell external momenta, the massless box with one off-shell momentum and the massless box with two adjacent off-shell momenta (depicted in Figs. 2a, 2b and 2c) do not fulfill the generalized permutahedron property. On the other hand, the  $\mathcal{F}$  polynomial does fulfill the generalized permutahedron property for the massless box with two or more off-shell legs such that two off-shell legs are on opposite sides (as depicted in Fig. 2d).

Therefore, we have to make concessions in the Minkowski regime with exceptional kinematics.

An observation of Arkani-Hamed, Hillman, Mizera is helpful (see [26, eq. (8)] and the discussion around it): the facet presentation of  $\mathbf{N}[\mathcal{F}]$  given in Theorem 3.5 turns out to hold in a quite broad range of kinematic regimes, even if  $\mathbf{N}[\mathcal{F}]$  is not a generalized permutahedron.



**Fig. 2.** Massless box with different external legs on- or off-shell. On-shell ( $p^2 = 0$ ) legs are drawn as dashed lines and off-shell ( $p^2 \neq 0$ ) legs with solid lines. Internal propagators are massless.



**Fig. 3.** Triangle Feynman graph relevant in QED. The two solid propagators have mass  $m$  and the solid legs have incoming squared momentum  $m^2$ . The dashed propagator is massless and the doubled leg has incoming squared momentum  $Q^2$ .

**Observation 3.8.** The Newton polytope of  $\mathcal{F}$  is often equal to the base polytope  $\mathbf{P}[z_{\mathcal{F}}]$  with the function  $z_{\mathcal{F}}$  defined as in Theorem 3.5.

This is significant since `feyntrap` uses the polytope  $\mathbf{P}[z_{\mathcal{F}}]$  internally as a substitute for  $\mathbf{N}[\mathcal{F}]$  as the former is easier to handle and faster to compute than the latter.

For instance, all massless boxes depicted in Fig. 2 have the property that the Newton polytopes of their  $\mathcal{F}$  polynomial are base polytopes described by the respective  $z_{\mathcal{F}}$  functions, i.e.  $\mathbf{N}[\mathcal{F}] = \mathbf{P}[z_{\mathcal{F}}]$ . In the first three cases (Figs. 2a, 2b, 2c) the  $z_{\mathcal{F}}$  function does not fulfill the inequalities (25). For the graph in Fig. 2d these inequalities are fulfilled and the associated Newton polytope  $\mathbf{N}[\mathcal{F}] = \mathbf{P}[z_{\mathcal{F}}]$  is a generalized permutahedron.

It would be very beneficial to have precise conditions for when  $\mathbf{P}[z_{\mathcal{F}}]$  indeed is equal to  $\mathbf{N}[\mathcal{F}]$ , we leave this for a future project. Empirically, we have observed that it is valid for quite a wide range of exceptional kinematics. We know, however, that this condition is not fulfilled for arbitrary exceptional kinematics [71]. An explicit counter example<sup>3</sup> is depicted in Fig. 3. For this triangle graph with the indicated exceptional kinematic configuration, the polytope  $\mathbf{N}[\mathcal{F}]$  is different from  $\mathbf{P}[z_{\mathcal{F}}]$ . We find that  $\mathcal{F}(\mathbf{x}) = m^2(x_1^2 + x_2^2) + (2m^2 - Q^2)x_1x_2$  which implies that  $\mathbf{N}[\mathcal{F}]$  is a one-dimensional polytope. On the other hand,  $\mathbf{P}[z_{\mathcal{F}}]$  can be shown to be a two-dimensional polytope. In  $D = 4$ , the Feynman integral associated to Fig. 3 is infrared divergent and therefore not quasi-finite. In  $D = 6$ , `feyntrap` can evaluate the integral without problems. Nonetheless, we expect there to be more complicated Feynman graphs with similarly exceptional external kinematics, that are quasi-finite, but which cannot be evaluated using `feyntrap`. We did not, however, manage to find such a graph.

Even if  $\mathbf{N}[\mathcal{F}] \neq \mathbf{P}[z_{\mathcal{F}}]$ , the Newton polytope  $\mathbf{N}[\mathcal{F}]$  is *bounded* by the base polytope  $\mathbf{P}[z_{\mathcal{F}}]$ . The reason for this is that  $\mathcal{F}$  can only lose monomials if we make the kinematics less generic.

**Theorem 3.9.** We have  $\mathbf{N}[\mathcal{F}] \subset \mathbf{P}[z_{\mathcal{F}}]$ .

*Efficient check of the generalized permutahedron property of a base polytope* Naively, it is quite hard to check if the base polytope  $\mathbf{P}[z]$  associated to a given function  $z : 2^E \rightarrow \mathbb{R}$  is a generalized permutahedron. There are of the order  $2^{|E|}$  many inequalities to be checked for (25). A more efficient way is to only check the following inequalities

$$z(\gamma \cup \{e\}) + z(\gamma \cup \{h\}) \leq z(\gamma) + z(\gamma \cup \{e, h\}) \quad (26)$$

for all subgraphs  $\gamma \subset E$  and edges  $e, h \in E \setminus \gamma$ . The inequalities (26) imply the ones in (25). For (26) less than  $|E|^2 2^{|E|}$  inequalities need to be checked. So, (26) is a more efficient version of (25).

### 3.4. Generalized permutahedral tropical sampling

`feyntrap` uses a slightly adapted version of the generalized permutahedron tropical sampling algorithm from [1, Sec. 6.1 and Sec. 7.2] to sample from the distribution given by  $\mu^{\text{tr}}$  in eq. (22).

The algorithm involves a preprocessing and a sampling step.

*Preprocessing* The first algorithmic task to prepare for the sampling from  $\mu^{\text{tr}}$  is to check in which regime the kinematic data are located. The kinematic data are provided via the matrix  $\mathcal{P}^{u,v}$  as it was defined in Section 2.1 and via a list of masses  $m_e$  for each edge  $e \in E$ . If the symmetric  $|V| \times |V|$  matrix  $\mathcal{P}^{u,v}$  is negative semi-definite (which is easy to check using matrix diagonalization), then we are in

<sup>3</sup> We thank Erik Panzer for sharing this (counter) example with us.

**Table 1**Table of the necessity of a deformation (def.) and the fulfillment of the generalized permutahedron property of  $\mathcal{F}$  (GP) in each kinematic regime.

	Euclidean	Pseudo-Euclidean	Minkowski
Generic	no def. / always GP	no def. / always GP	def. / always GP
Exceptional	no def. / always GP	no def. / not always GP	def. / not always GP

the Euclidean regime. Similarly we check if the defining (in)equalities for the other kinematic regimes given in Section 2.2 are fulfilled or not. Depending on the kinematic regime, we need to use a contour deformation for the integration or not. Further, if the kinematics are Euclidean or generic, we know that the generalized permutahedron property of  $\mathcal{F}$  is fulfilled (also thanks to the unproven Conjecture 3.7). Table 1 summarizes this dependence of the algorithm on the kinematic regime.

If we find that we are at an exceptional and non-Euclidean kinematic point,  $\mathbf{N}[\mathcal{F}]$  might not be a generalized permutahedron and it might not even be equal to  $\mathbf{P}[\mathcal{Z}_{\mathcal{F}}]$ . In this case, the program prints a message warning the user that the integration might not work. The program then continues under the assumption that  $\mathbf{N}[\mathcal{F}] = \mathbf{P}[\mathcal{Z}_{\mathcal{F}}]$ . In any other case,  $\mathbf{N}[\mathcal{F}]$  is a generalized permutahedron and equal to  $\mathbf{P}[\mathcal{Z}_{\mathcal{F}}]$ . Hence, the tropical sampling algorithm is guaranteed to give a convergent Monte Carlo integration method by [1, Sec. 6.1].

The next task is to compute the loop number  $L_{\gamma}$  and check if  $\gamma$  is mass-momentum spanning (by asking if  $\mathcal{F}_{G/\gamma} = 0$ ) for each subgraph  $\gamma \subset E$ . Using these data, we can compute the values of  $z_{\mathcal{U}}(\gamma)$  and  $z_{\mathcal{F}}(\gamma)$  for all subgraphs  $\gamma \subset E$  using the respective formulas from Theorems 3.4 and 3.5.

If we are at an exceptional and non-Euclidean kinematic point, we check the inequalities (26) for the  $z_{\mathcal{F}}$  function. If they are all fulfilled, then  $\mathbf{P}[\mathcal{Z}_{\mathcal{F}}]$  is a generalized permutahedron and we get further indication that the tropical integration step will be successful. The program prints a corresponding message in this case. Also assuming that Assumption 3.2 is fulfilled, we can compute all integrals in (23) efficiently.

Note that even in the pseudo-Euclidean and the Minkowski regimes with exceptional kinematics, the integration is often successful. For instance, we can integrate all Feynman graphs depicted in Fig. 2 regardless of the fulfillment of the generalized permutahedron property. In fact, we did not find a quasi-finite example where the algorithm fails (even though the convergence rate is quite bad for examples in highly exceptional kinematic regimes). We emphasize, however, that the user should check the convergence of the result separately when integrating at a manifestly exceptional and non-Euclidean kinematic point. For instance, by running the program repeatedly with different numbers of sample points or by slightly perturbing the kinematic point. Recall that for generic kinematics  $\mathbf{N}[\mathcal{F}]$  is always a generalized permutahedron by Theorem 3.6 and the integration is guaranteed to work if the finiteness assumptions are fulfilled.

The next computational step is to compute the *generalized degree of divergence* (see [1, Sec. 7.2]) for each subgraph  $\gamma \subset E$ . It is defined by

$$\omega(\gamma) = \sum_{e \in \gamma} v_e - DL_{\gamma}/2 - \omega \delta_{\gamma}^{\text{m.m.}}, \quad (27)$$

where  $L_{\gamma}$  is the loop number of the subgraph  $\gamma$  and  $\delta_{\gamma}^{\text{m.m.}} = 1$  if  $\gamma$  is mass-momentum spanning and 0 otherwise. The prefactor  $\omega$  of  $\delta_{\gamma}^{\text{m.m.}}$  is the usual superficial degree of divergence of the overall graph  $G$  as it was defined in Section 2.1,  $\omega = \sum_{e \in E} v_e - DL/2$ .

If  $\omega(\gamma) \leq 0$  for any proper subgraph  $\gamma$ , then we discovered a subdivergence. This means that all integrals (16) are divergent. Tropical sampling is not possible in this case and the program prints an error message and terminates. An additional analytic continuation step from (16) to a set of quasi-finite integrals (see Section 2.4) would resolve this problem. Translating a divergent integral into a linear combination of quasi-finite integrals is always possible, but we will leave the implementation of this step into `feyntrap` to a future research project.

If we have  $\omega(\gamma) > 0$  for all  $\gamma \subset E$ , we can proceed to the key preparatory step for generalized permutahedral tropical sampling: We use  $\omega(\gamma)$  to compute the following auxiliary subgraph function  $J(\gamma)$ , which is *recursively* defined by setting  $J(\emptyset) = 1$ , agreeing that  $\omega(\emptyset) = 1$  and

$$J(\gamma) = \sum_{e \in \gamma} \frac{J(\gamma \setminus e)}{\omega(\gamma \setminus e)} \text{ for all } \gamma \subset E, \quad (28)$$

where  $\gamma \setminus e$  is the subgraph  $\gamma$  with the edge  $e$  removed. The terminal element of this recursion is the subgraph that contains all edges  $E$  of  $G$ . We find that  $J(E) = I^{\text{tr}}$ , where  $I^{\text{tr}}$  is the normalization factor in (22) and (23) (see [1, Proposition 29] for a proof and details).

In the end of the preprocessing step we compile a table with the information  $L_{\gamma}$ ,  $\delta_{\gamma}^{\text{m.m.}}$ ,  $\omega(\gamma)$  and  $J(\gamma)$  for each subgraph  $\gamma \subset E$  and store it in the memory of the computer.

**Sampling step** The sampling step of the algorithm repeats the following simple algorithm to generate samples  $\mathbf{x} \in \mathbb{P}_{+}^E$  that are distributed according to the probability density (22). It is completely described in Algorithm 1. The runtime of our implementation of the algorithm grows roughly quadratically with  $|E|$ , but a linear runtime is achievable. The validity of the algorithm was proven in a more general setup in [1, Proposition 31]. The additional computation of the values of  $\mathcal{U}^{\text{tr}}(\mathbf{x})$  and  $\mathcal{V}^{\text{tr}}(\mathbf{x})$  is an application of an optimization algorithm by Fujishige and Tomizawa [72] (see also [1, Lemma 26]).

The key step of the sampling algorithm is to interpret the recursion (28) as a probability distribution for a given subgraph over its edges. That means, for a given  $\gamma \subset E$  we define  $p_e^{\gamma} = \frac{1}{J(\gamma)} \frac{J(\gamma \setminus e)}{\omega(\gamma \setminus e)}$ . Obviously,  $p_e^{\gamma} \geq 0$  and by (28) we have  $\sum_{e \in \gamma} p_e^{\gamma} = 1$ . So, for each  $\gamma \subset E$ ,  $p_e^{\gamma}$  gives a proper probability distribution on the edges of the subgraph  $\gamma$ .

**Algorithm 1** Generating a sample distributed as  $\mu^{\text{tr}}$  from (22).

---

```

Initialize the variables  $\gamma = E$  and  $\kappa, U = 1$ .
while  $\gamma \neq \emptyset$  do
    Pick a random edge  $e \in \gamma$  with probability  $p_e^\gamma = \frac{1}{J(\gamma)} \frac{J(\gamma \setminus e)}{\omega(\gamma \setminus e)}$ .
    Set  $x_e = \kappa$ .
    If  $\gamma$  is mass-momentum spanning but  $\gamma \setminus e$  is not, set  $V = x_e$ .
    If  $L_{\gamma \setminus e} < L_\gamma$ , multiply  $U$  with  $x_e$  and store the result in  $U$ , i.e. set  $U \leftarrow x_e \cdot U$ .
    Remove the edge  $e$  from  $\gamma$ , i.e. set  $\gamma \leftarrow \gamma \setminus e$ .
    Pick a uniformly distributed random number  $\xi \in [0, 1]$ .
    Multiply  $\kappa$  with  $\xi^{1/\omega(\gamma)}$  and store the result in  $\kappa$ , i.e. set  $\kappa \leftarrow \kappa \xi^{1/\omega(\gamma)}$ .
end while
Return  $\mathbf{x} = [x_0, \dots, x_{|E|-1}] \in \mathbb{P}_+^E$ ,  $\mathcal{U}^{\text{tr}}(\mathbf{x}) = U$  and  $\mathcal{V}^{\text{tr}}(\mathbf{x}) = V$ .

```

---

The algorithm can also be interpreted as iteratively *cutting* edges of the graph  $G$ : We start with  $\gamma = E$  and pick a random edge with probability  $p_e^\gamma$ . This edge is *cut* and removed from  $\gamma$ . We continue with the newly obtained graph and repeat this cutting process until all edges are removed. In the course of this, Algorithm 1 computes appropriate random values for the coordinates  $\mathbf{x} \in \mathbb{P}_+^E$ .

## 4. Numerical integration

### 4.1. Monte Carlo integration

We now have all the necessary tools at hand to evaluate the integrals in (23) using Monte Carlo integration. In this section, we briefly review this procedure. The integrals in (23) are of the form

$$I_f = \int_{\mathbb{P}_+^E} f(\mathbf{x}) \mu^{\text{tr}}, \quad (29)$$

where, thanks to the tropical approximation property,  $f(\mathbf{x})$  is a function that is at most log-singular inside, or on the boundary of,  $\mathbb{P}_+^E$ . To evaluate such an integral, we first use the tropical sampling Algorithm 1 to randomly sample  $N$  points  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)} \in \mathbb{P}_+^E$  that are distributed according to the tropical probability measure  $\mu^{\text{tr}}$ . By the central limit theorem and as  $f(\mathbf{x})$  is square-integrable,

$$I_f \approx I_f^{(N)} \quad \text{where} \quad I_f^{(N)} = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}^{(i)}). \quad (30)$$

For sufficiently large  $N$ , the expected error of this approximation of the integral  $I_f$  is

$$\sigma_f = \sqrt{\frac{I_{f^2} - I_f^2}{N}} \quad \text{where} \quad I_{f^2} = \int_{\mathbb{P}_+^E} f(\mathbf{x})^2 \mu^{\text{tr}}, \quad (31)$$

which itself can be estimated (as long as  $f(\mathbf{x})^2$  is square-integrable) by

$$\sigma_f \approx \sigma_f^{(N)} \quad \text{where} \quad \sigma_f^{(N)} = \sqrt{\frac{1}{N-1} (I_{f^2}^{(N)} - (I_f^{(N)})^2)} \quad \text{and} \quad I_{f^2}^{(N)} = \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}^{(i)})^2. \quad (32)$$

To evaluate the estimator  $I_f^{(N)}$  and the expected error  $\sigma_f^{(N)}$  it is necessary to evaluate  $f(\mathbf{x})$  for  $N$  different values of  $\mathbf{x}$ . As the random points  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)} \in \mathbb{P}_+^E$  can be obtained quite quickly using Algorithm 1, this evaluation becomes a bottleneck. In the next section, we describe a fast method to perform this evaluation, which is implemented in `feyntrop` to efficiently obtain Monte Carlo estimates and error terms for the integrals in (23).

### 4.2. Fast evaluation of (deformed) Feynman integrands

To evaluate the integrals in (23) using a Monte Carlo approach we do not only have to be able to sample from the distribution  $\mu^{\text{tr}}$ , but we also need to rapidly evaluate the remaining integrand (denoted as  $f(\mathbf{x})$  in the last section). Explicitly for the numerical evaluation of (23), we have to be able to compute  $X_e = x_e \exp(-i\lambda \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}))$  as well as  $\mathcal{U}(\mathbf{X})$ ,  $\mathcal{V}(\mathbf{X})$  and  $\det \mathcal{J}_\lambda(\mathbf{x})$  for any  $\mathbf{x} \in \mathbb{P}_+^E$ .

**Evaluation of the  $\mathcal{U}$  and  $\mathcal{F}$  polynomials** Surprisingly, the explicit polynomial expression for  $\mathcal{U}$  and  $\mathcal{F}$  from eq. (5) are *harder* to evaluate than the matrix and determinant expression (4) if the underlying graph exceeds a certain complexity. The reason for this is that the number of monomials in (5) increases exponentially with the loop number (see, e.g., [73] for the asymptotic growth rate of the number of spanning trees in a regular graph), while the size of the matrices in (4) only increases linearly. Standard linear algebra algorithms as the Cholesky or LU decompositions [74] provide polynomial time algorithms to compute the inverse and determinant of  $\mathcal{L}(\mathbf{x})$  and therefore values of  $\mathcal{U}(\mathbf{x})$  and  $\mathcal{F}(\mathbf{x})$  (see, e.g., [1, Sec. 7.1]). In fact, the linear algebra problems on *graph Laplacian* matrices that need to be solved to compute  $\mathcal{U}(\mathbf{x})$  and  $\mathcal{F}(\mathbf{x})$  fall into a class of problems for which *nearly linear runtime* algorithms are available [75].

**Explicit formulas for the  $\mathcal{V}$  derivatives** We need explicit formulas for the derivatives of  $\mathcal{V}$ . These formulas provide fast evaluation methods for  $\mathbf{X}$  and the Jacobian  $\mathcal{J}_\lambda(\mathbf{x})$ .

Consider the  $(|V| - 1) \times (|V| - 1)$  matrix  $\mathcal{M}(\mathbf{x}) = \mathcal{L}^{-1}(\mathbf{x}) \mathcal{P} \mathcal{L}^{-1}(\mathbf{x})$  with  $\mathcal{L}(\mathbf{x})$  and  $\mathcal{P}$  as defined in Section 2.1. For edges  $e$  and  $h$  that connect the vertices  $u_e, v_e$  and  $u_h, v_h$  respectively, we define

$$\begin{aligned} \mathcal{A}(\mathbf{x})_{e,h} &= \frac{1}{x_e x_h} (\mathcal{M}(\mathbf{x})_{u_e, u_h} + \mathcal{M}(\mathbf{x})_{v_e, v_h} - \mathcal{M}(\mathbf{x})_{u_e, v_h} - \mathcal{M}(\mathbf{x})_{v_e, u_h}) \\ \mathcal{B}(\mathbf{x})_{e,h} &= \frac{1}{x_e x_h} (\mathcal{L}^{-1}(\mathbf{x})_{u_e, u_h} + \mathcal{L}^{-1}(\mathbf{x})_{v_e, v_h} - \mathcal{L}^{-1}(\mathbf{x})_{u_e, v_h} - \mathcal{L}^{-1}(\mathbf{x})_{v_e, u_h}), \end{aligned} \quad (33)$$

where we agree that  $\mathcal{L}^{-1}(\mathbf{x})_{u,v} = \mathcal{M}(\mathbf{x})_{u,v} = 0$  if any of  $u$  or  $v$  is equal to  $v_0$ , the arbitrary vertex that was removed in the initial expression of the Feynman integral (1). It follows from (4) and the matrix differentiation rule  $\frac{\partial}{\partial x_e} \mathcal{L}^{-1}(\mathbf{x})_{u,v} = \left( -\mathcal{L}^{-1}(\mathbf{x}) \frac{\partial \mathcal{L}}{\partial x_e}(\mathbf{x}) \mathcal{L}^{-1}(\mathbf{x}) \right)_{u,v}$  that

$$\frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}) = -\mathcal{A}(\mathbf{x})_{e,e} + m_e^2, \quad \frac{\partial^2 \mathcal{V}}{\partial x_e \partial x_h}(\mathbf{x}) = 2\delta_{e,h} \frac{\mathcal{A}(\mathbf{x})_{e,e}}{x_e} - 2(\mathcal{A}(\mathbf{x}) \circ \mathcal{B}(\mathbf{x}))_{e,h}, \quad (34)$$

where we use the *Hadamard* or *element-wise matrix product*,  $(\mathcal{A}(\mathbf{x}) \circ \mathcal{B}(\mathbf{x}))_{e,h} = \mathcal{A}(\mathbf{x})_{e,h} \cdot \mathcal{B}(\mathbf{x})_{e,h}$ .

**Computation of the relevant factors in the integrands of (23)** We summarize the necessary steps to compute all the factors in the deformed and  $\epsilon$ -expanded tropical Feynman integral representation (23).

1. Compute the graph Laplacian  $\mathcal{L}(\mathbf{x})$  as defined in Section 2.1.
2. Compute the inverse  $\mathcal{L}^{-1}(\mathbf{x})$  (e.g. by Cholesky decomposing  $\mathcal{L}(\mathbf{x})$ ).
3. Use this to evaluate the derivatives of  $\mathcal{V}(\mathbf{x})$  via the formulas in (33) and (34).
4. Compute the values of the deformed  $\mathbf{X}$  parameters:  $X_e = x_e \exp(-i\lambda \frac{\partial \mathcal{V}}{\partial x_e}(\mathbf{x}))$ .
5. Compute the Jacobian  $\mathcal{J}_\lambda(\mathbf{x})$  using the formula in (11).
6. Evaluate  $\det \mathcal{J}_\lambda(\mathbf{x})$  (e.g. by using a LU decomposition of  $\mathcal{J}_\lambda(\mathbf{x})$ ).
7. Compute the deformed graph Laplacian  $\mathcal{L}(\mathbf{X})$ .
8. Compute  $\mathcal{L}^{-1}(\mathbf{X})$  and  $\det \mathcal{L}(\mathbf{X})$  (e.g. by using a LU decomposition of  $\mathcal{L}(\mathbf{X})$  as a Cholesky decomposition is not possible, because  $\mathcal{L}(\mathbf{X})$  is not a hermitian matrix in contrast to  $\mathcal{L}(\mathbf{x})$ ).
9. Use the formulas (4) to obtain values for  $\mathcal{U}(\mathbf{X})$ ,  $\mathcal{F}(\mathbf{X})$  and  $\mathcal{V}(\mathbf{X}) = \mathcal{F}(\mathbf{X})/\mathcal{U}(\mathbf{X})$ .

The computation obviously simplifies if we set  $\lambda = 0$ , in which case we have  $\mathbf{X} = \mathbf{x}$ . We are allowed to set  $\lambda = 0$  if we do not need the contour deformation. This is the case, for instance, in the Euclidean or the pseudo-Euclidean regimes. In our implementation we check if we are in these regimes and adjust the evaluation of the integrand accordingly.

## 5. The program `feynthrop`

We have implemented the contour-deformed tropical integration algorithm, which we discussed in the previous sections, in a C++ module named `feynthrop`. This module is an upgrade to previous code developed by the first author in [1].

`feynthrop` was checked against `AMFlow` [48] and `pySecDec` [41] for roughly 15 different diagrams with 1-3 loops and 2-5 legs at varying kinematics points, in both the Euclidean and Minkowski regimes, finding agreement in all cases within the given uncertainty bounds. In the Euclidean regime, the original algorithm was checked against numerous analytic computations that were obtained at high loop order using conformal four-point integral and graphical function techniques [76].

Note that our prefactor convention, which we fixed in eqs. (1) and (2), differs from the one in `AMFlow` and `pySecDec` by a factor of  $(-1)^{|V|}$ , where  $|V| = \sum_{e=0}^{|E|-1} v_e$ . In comparison to `FIESTA` [24], our convention differs by a factor of  $(-1)^{|V|} \exp(-L\gamma_E \epsilon)$ .

### 5.1. Installation

The source code of `feynthrop` is available in the repository <https://github.com/michibo/feynthrop> on github. It can be downloaded and built by running the following sequence of commands

```
git clone --recursive https://github.com/michibo/feynthrop.git
cd feynthrop
make
```

in a Linux environment. `feynthrop` is interfaced with `python` [4] via the library `pybind11` [5]<sup>4</sup>. Additionally, it uses the optimized linear algebra routines from the `Eigen3` package [2], the `OpenMP` C++ module [77] for the parallelization of the Monte Carlo sampling step and the `xoshiro256+` pseudo random number generator [3].

<sup>4</sup> Note added in proof: Due to compatibility issues on some hardware, the newest version of `feynthrop` available and described at <https://github.com/michibo/feynthrop> does not make use of `pybind11` anymore. This new version also provides a low-level command-line interface that works without any dependency on `python`. This interface enables the easy use of `feynthrop` in high-performance computing environments.



`feyntrop` can be loaded in a python environment by importing the file `py_feyntrop.py`, located in the top directory of the package. To ensure that `feyntrop` was built correctly, one may execute the python file `/tests/test_suite.py`. This script compares the output of `feyntrop` against pre-computed values. To do so, it will locally compute six examples with 1-2 loops and 2-5 legs, some in the Euclidean and others in the Minkowski regime.

The file `py_feyntrop.py` includes additional functionality for the python interface serving three purposes. Firstly, it simplifies the specification of vertices and edges of a Feynman diagram in comparison to the C++ interface of `feyntrop`. Secondly, it allows for self-chosen momentum variables given by a set of replacement rules, instead of having to manually specify the full scalar product matrix  $\mathcal{P}^{u,v}$  from (4). Lastly, the output of the  $\epsilon$  expansion can be printed in a readable format. To do so, the `py_feyntrop.py` program uses the `sympy` [6] library.

As already indicated in Section 2.1, we employ zero-indexing throughout. This means that edges and vertices are labeled as  $\{0, 1, \dots\}$ . This facilitates seamless interoperability with the programming language features of python.

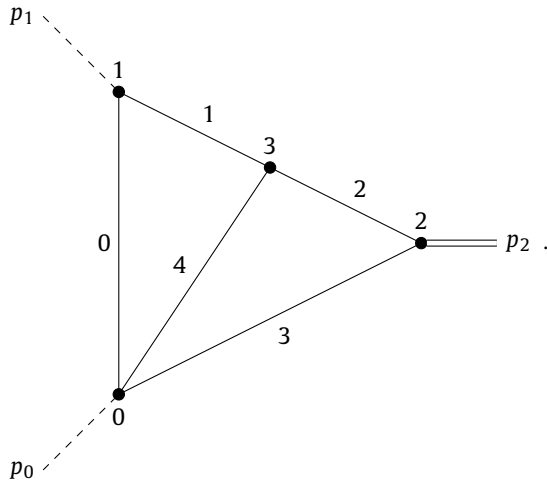
## 5.2. Basic usage of `feyntrop`

In this section, we will illustrate the basic workflow of `feyntrop` with an example. The code for this example can be executed and inspected with `jupyter` [78] by calling

```
jupyter notebook tutorial_2L_3pt.ipynb
```

within the top directory of the `feyntrop` package.

We will integrate the following 2-loop 3-point graph in  $D = 2 - 2\epsilon$  dimensional spacetime:



The dashed lines denote on-shell, massless particles with momenta  $p_0$  and  $p_1$  such that  $p_0^2 = p_1^2 = 0$ . The solid, internal lines each have mass  $m$ . The double line is associated to some off-shell momentum  $p_2^2 \neq 0$ . For the convenience of the reader, both vertices and edges are labeled explicitly in this example. `feyntrop` requires us to label the external vertices (as defined in Section 2.2) *before* the internal vertices. In the current example, the vertices are  $V = V^{\text{ext}} \sqcup V^{\text{int}} = \{0, 1, 2\} \sqcup \{3\}$ .

The momentum space Feynman integral representation (1) with unit edge weights reads

$$\mathcal{I} = \pi^{-2+2\epsilon} \int \frac{d^{2-2\epsilon}k_0 d^{2-2\epsilon}k_1}{(q_0^2 - m^2 + i\epsilon)(q_1^2 - m^2 + i\epsilon)(q_2^2 - m^2 + i\epsilon)(q_3^2 - m^2 + i\epsilon)(q_4^2 - m^2 + i\epsilon)}, \quad (35)$$

where we integrated out the  $\delta$  functions in eq. (1) by requiring that  $q_0 = k_0$ ,  $q_1 = k_0 + p_1$ ,  $q_2 = k_0 + k_1 + p_1$ ,  $q_3 = p_0 - k_0 - k_1$  and  $q_4 = k_1$ . We choose the phase space point

$$m^2 = 0.2, \quad p_0^2 = p_1^2 = 0, \quad p_2^2 = 1, \quad (36)$$

which is in the Minkowski regime because  $p_2^2 > 0$  - see Section 2.2. To begin this calculation, first open a python script or a jupyter notebook and import `py_feyntrop`:

```
from py_feyntrop import *
```

Here we are assuming that `feyntrop.so` and `py_feyntrop.py` are both in the working directory.

To define the graph, we provide a list of edges with edge weights  $v_e$  and squared masses  $m_e^2$ :

$$((u_0, v_0), v_0, m_0^2), \dots, ((u_{|E|-1}, v_{|E|-1}), v_{|E|-1}, m_{|E|-1}^2). \quad (37)$$

The notation  $(u_e, v_e)$  denotes an edge  $e$  incident to the vertices  $u_e$  and  $v_e$ . We therefore write

```
edges = [((0,1), 1, 'mm'), ((1,3), 1, 'mm'), ((2,3), 1, 'mm'),
          ((2,0), 1, 'mm'), ((0,3), 1, 'mm')]
```

in the code to input the graph which is depicted above. The ordering of vertices  $(u_e, v_e)$  in an edge is insignificant. Here we set  $v_e = 1$  for all  $e$ . The chosen symbol for  $m^2$  is `mm`, which will be replaced by its value 0.2 later on. It is also allowed to input numerical values for masses already in the `edges` list, for instance by replacing the first element of the list by `((0,1), 1, '0.2')`.

Next we fix the momentum variables. Recall that the external vertices are required to be labeled  $\{0, 1, \dots, |V^{\text{ext}}| - 1\}$ , so the external momenta are  $p_0, \dots, p_{|V^{\text{ext}}|-1}$ . Moreover, the last momentum is inferred automatically by `feyntrop` using momentum conservation, leaving  $p_0, \dots, p_{|V^{\text{ext}}|-2}$  to be fixed by the user. A momentum configuration is then specified by the collection of scalar products,

$$p_u \cdot p_v \text{ for all } 0 \leq u \leq v \leq |V^{\text{ext}}| - 2. \quad (38)$$

In the code, we must provide replacement rules for these scalar products in terms of some variables of choice. For the example at hand,  $|V^{\text{ext}}| = 3$ , so we must provide replacement rules for  $p_0^2$ ,  $p_1^2$  and  $p_0 \cdot p_1$ . In the syntax of `feyntrop` we thus write

```
replacement_rules = [(sp[0,0], '0'), (sp[1,1], '0'), (sp[0,1], 'pp2/2')]
```

where `sp[u,v]` stands for  $p_u \cdot p_v$ , the scalar product of  $p_u$  and  $p_v$ . We have immediately set  $p_0^2 = p_1^2 = 0$  and also defined a variable `pp2` which stands for  $p_2^2$ , as, by momentum conservation,

$$p_2^2 = 2p_0 \cdot p_1. \quad (39)$$

Eventually, we fix numerical values for the two auxiliary variables `pp2` and `mm`. This is done via

```
phase_space_point = [('mm', 0.2), ('pp2', 1)]
```

which fixes  $m^2 = 0.2$  and  $p_2^2 = 1$ . It is possible to obtain the  $\mathcal{P}^{u,v}$  matrix (as defined in Section 2.1) and a list of all the propagator masses, which are computed from the previously provided data, by

```
P_uv_matrix, m_sqr_list = prepare_kinematic_data(edges, replacement_rules,
                                                phase_space_point)
```

The final pieces of data that need to be provided are

```
D0 = 2
eps_order = 5
Lambda = 7.6
N = int(1e7)
```

`D0` is the integer part of the spacetime dimension  $D = D_0 - 2\epsilon$ . We expand up to, but not including, `eps_order`. `Lambda` denotes the deformation parameter from (9). `N` is the number of Monte Carlo sampling points.

Tropical Monte Carlo integration of the Feynman integral, with the kinematic configuration chosen above, is now performed by running the command

```
trop_res, Itr = tropical_integration(
    N,
    D0,
    Lambda,
    eps_order,
    edges,
    replacement_rules,
    phase_space_point)
```

If the program runs correctly (i.e. no error is printed), `trop_res` will contain the  $\epsilon$ -expansion (16) *without* the prefactor  $\Gamma(\omega)/(\Gamma(v_1) \cdots \Gamma(v_{|E|})) = \Gamma(2\epsilon + 3)$ . `Itr` is the value of the normalization factor in (22). Running this code on a laptop, we get, after a couple of seconds, the output

```
Prefactor: gamma(2*eps + 3).
(Effective) kinematic regime: Minkowski (generic).
Generalized permutahedron property: fulfilled.
Analytic continuation: activated. Lambda = 7.6
Started integrating using 8 threads and N = 1e+07 points.
Finished in 6.00369 seconds = 0.00166769 hours.

-- eps^0: [-46.59 +/- 0.13] + i * [ 87.19 +/- 0.12]
-- eps^1: [-274.46 +/- 0.55] + i * [111.26 +/- 0.55]
-- eps^2: [-435.06 +/- 1.30] + i * [-174.47 +/- 1.33]
-- eps^3: [-191.72 +/- 2.15] + i * [-494.69 +/- 2.14]
-- eps^4: [219.15 +/- 2.68] + i * [-431.96 +/- 2.67]
```

These printed values for the  $\epsilon$  expansion are contained in the list `trop_res` in the following format:

$$[(\text{re}_0, \sigma_0^{\text{re}}), (\text{im}_0, \sigma_0^{\text{im}}), \dots, (\text{re}_4, \sigma_4^{\text{re}}), (\text{im}_4, \sigma_4^{\text{im}})],$$

where  $\text{re}_0 \pm \sigma_0^{\text{re}}$  is the real part of the 0th order term, and so forth.

The  $\epsilon$ -expansion, with prefactor included, can finally be output via

```
eps_expansion(trop_res, edges, D0)
```

giving

```
174.3842115*i - 93.17486662 + eps*(-720.8731714 + 544.3677186*i) +
eps**2*(-2115.45025 + 496.490128*i) + eps**3*(-3571.990969 - 677.5254794*i) +
eps**4*(-3872.475723 - 2726.965026*i) + O(eps**5)
```

If the `tropical_integration` command fails, for instance because a subdivergence of the input graph is detected, it prints an error message. The command also prints a warning if the kinematic point is too exceptional and convergence cannot be guaranteed due to the  $\mathcal{F}$  polynomial lacking the generalized permutahedron property (see Section 3.3).

### 5.3. Deformation parameter

The uncertainties on the integrated result may greatly vary with the value of the deformation parameter  $\lambda$  from (9) (what was called `Lambda` above). Moreover, the optimal value of  $\lambda$  might change depending on the phase space point. It is up to the user to pick a suitable value by trial and error, for instance by integrating several times with a low number of sampling points  $N$ . In Section 6, this method is used to evaluate multiple examples of Feynman integrals in the Minkowski regime. Typical values for the parameter  $\lambda$  can be found there. It would be beneficial to automate this procedure, possibly by minimizing the sampling variance with respect to  $\lambda$ , for instance by solving  $\partial_\lambda \sigma_f = 0$  with  $\sigma_f$  defined in (31), or by tightening the bounds in Assumption 3.2 (see the discussion after this assumption). We leave the exploration of such ideas to future research.

Note that  $\lambda$  has mass dimension  $1/\text{mass}^2$ . Heuristically, this implies that the value of  $\lambda$  should be of order  $\mathcal{O}(1/\Lambda^2)$ , where  $\Lambda$  is the maximum physical scale in the given computation.

## 6. Examples of Feynman integral evaluations

In this section, we use `feyntrop` to numerically evaluate certain Feynman integrals of interest. The first two examples, 6.1 and 6.2, show that `feyntrop` is capable of computing Feynman integrals at high loop-orders involving many kinematic scales. The four examples that follow, 6.4, 6.3, 6.6 and 6.5, demonstrate that `feyntrop` is capable of computing phenomenologically relevant diagrams. The final example, 6.7, is an invitation to study conformal integrals with our code, as they are important for, e.g.,  $\mathcal{N} = 4$  SYM and the cosmological bootstrap.

We have chosen phase space points which are not close to thresholds to insure good numerical convergence, and expand up to and including  $\epsilon^{2L}$  in all but up the last example.

Each of the following examples can be computed with `feyntrop` using  $10^8$  sampling points within a few minutes on a consumer laptop with 16 GBs of RAM. To crosscheck, we used the same machine to evaluate the examples using both `AMFlow`<sup>5</sup> and `pySecDec`. All computations agreed within the indicated error bounds. Our computations using `AMFlow` and `pySecDec` did not always terminate. Particularly for the Examples 6.1 and 6.6, neither software finished due to memory constraints of 16 GB on our test laptop. After the initial version of this article became available, Vitaly Magerya informed us that he was able to reproduce also Example 6.6 and verify our numbers using `pySecDec` with an only slightly more powerful computer. He also found indication that Example 6.1 is reproducible using a new version of `pySecDec` that was made available three months after the initial version of the present article was posted [83].

We emphasize that these additional computations using `AMFlow` and `pySecDec` should be seen as a crosscheck and not a benchmark comparison. A comparison of `feyntrop` and `AMFlow` is difficult as the former directly integrates via Monte Carlo while the latter integrates via differential equations. To integrate a Feynman integral using `AMFlow` an IBP system needs to be solved. Finding this solution is a memory constrained problem and a 16 GB laptop is not appropriate to systematically perform computations within this approach. If the IBP system is solved, `AMFlow` provides the evaluated integral at an accuracy which is almost unachievable using a Monte Carlo approach. The comparison to `pySecDec` is similarly flawed as it can also deal with inherently divergent integrals. To do so it has to check for divergences in each sector which takes time. Moreover, it can deal with completely general algebraic integrals, whereas `feyntrop` completely relies on the inherent mathematical structure of Feynman integrals. We postpone a proper benchmark comparison with the new version of `pySecDec` and updated versions of `AMFlow` to a future research project.

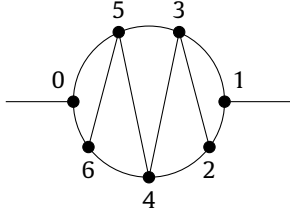
To further highlight the capabilities of `feyntrop`, we computed every example on a high-performance machine, namely a single AMD EPYC 7H12 64-core processor using all cores. For each example we use  $10^8$  sample points to get a relative accuracy of the order of  $10^{-2}$  to  $10^{-4}$ . The output for each example includes the total evaluation time that `feyntrop` needs to compute the respective diagram. This evaluation time includes all steps of the computation. The time needed for the preprocessing step is negligible in comparison to the sampling time as long as the number of edges is relatively small (i.e.  $|E| \leq 15$ ). Hence, for such moderate numbers of propagators, the evaluation time is proportional to the number of sample points. The sampling step is completely parallelizable. So, doubling the number of CPUs, halves the evaluation time. As the evaluation is based on Monte Carlo, increasing the relative accuracy is costly: one additional digit costs a 100-fold increase in CPU-time.

The code for each example can be found on the `github` repository in the folder `examples`.

<sup>5</sup> As `AMFlow` relies on DEQs for Feynman integrals, it is necessary to link it to IBP software. In our examples, we tried the following two options for IBP software: 1) `FIRE` [79] combined with `LiteRed` [80,81], and 2) `Blade` [82].

### 6.1. A 5-loop 2-point zigzag diagram

We evaluate the following 5-loop 2-point function with all masses different in  $D = 3 - 2\epsilon$  dimensions



corresponding to the edge set

```
edges = [(0,6), 1, '1'), ((0,5), 1, '2'), ((5,6), 1, '3'),
         ((6,4), 1, '4'), ((5,3), 1, '5'), ((5,4), 1, '6'),
         ((4,3), 1, '7'), ((4,2), 1, '8'), ((3,2), 1, '9'),
         ((3,1), 1, '10'), ((2,1), 1, '11')]
```

Here we already input the chosen values for masses, namely  $m_e^2 = e + 1$  for  $e = 0, \dots, 10$ .

There is only a single independent external momentum  $p_0$ , whose square we set equal to 100 via

```
replacement_rules = [(sp[0,0], 'pp0')]
phase_space_point = [('pp0', 100)]
```

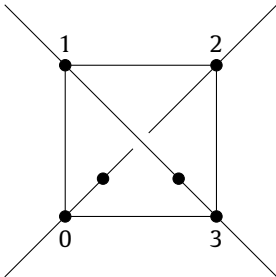
The value  $\lambda = 0.02$  turns out to give small errors, which is of order  $\mathcal{O}(1/p_0^2)$  in accordance with the comment at the end of the previous section. Using  $N = 10^8$  Monte Carlo sampling points, feynthrop's `tropical_integration` command gives

```
Prefactor: gamma(5*eps + 7/2).
(Effective) kinematic regime: Minkowski (generic).
Finished in 9.62 seconds.
-- eps^0: [0.0001976 +/- 0.0000016] + i * [0.0001415 +/- 0.0000018]
-- eps^1: [-0.004961 +/- 0.000023 ] + i * [-0.000802 +/- 0.000024 ]
-- eps^2: [ 0.04943 +/- 0.00017 ] + i * [-0.01552 +/- 0.00017 ]
-- eps^3: [-0.25468 +/- 0.00083 ] + i * [ 0.24778 +/- 0.00093 ]
-- eps^4: [ 0.5909 +/- 0.0033 ] + i * [ -1.7261 +/- 0.0038 ]
-- eps^5: [ 1.048 +/- 0.012 ] + i * [ 7.410 +/- 0.013 ]
-- eps^6: [-14.652 +/- 0.037 ] + i * [-20.933 +/- 0.038 ]
-- eps^7: [ 65.87 +/- 0.10 ] + i * [ 35.25 +/- 0.11 ]
-- eps^8: [-190.90 +/- 0.27 ] + i * [ -4.91 +/- 0.26 ]
-- eps^9: [ 393.08 +/- 0.70 ] + i * [-182.56 +/- 0.59 ]
-- eps^10: [-558.01 +/- 1.64 ] + i * [ 685.62 +/- 1.29 ]
```

We have not been able to compute this expansion with AMFlow for the sake of verification. The memory constraints of 16 GB were insufficient. pySecDec applied to this example exhausted the available memory while building the sector decomposition library on our test laptop, but Vitaly Magerya informed us that he was able to create the integration library on a 32 GB 8-core Intel i7 computer in a couple of hours. We again emphasize that, for a proper benchmark comparison, our AMFlow and pySecDec code should be put on a machine with more memory. Still, this example illustrates that feynthrop can operate at high loop order with little memory, CPU and time resources.

### 6.2. A 3-loop 4-point envelope diagram

Here, we evaluate a  $D = 4 - 2\epsilon$  dimensional, non-planar, 3-loop 4-point, envelope diagram:



The dots on the crossed lines represent squared propagators, i.e. edge weights equal to 2, rather than vertices. The weighted edge set with corresponding mass variables is thus

```
edges = [((0,1), 1, 'mm0'), ((1,2), 1, 'mm1'), ((2,3), 1, 'mm2'),
         ((3,0), 1, 'mm3'), ((0,2), 2, 'mm4'), ((1,3), 2, 'mm5')]
```

Let us define the two-index Mandelstam variables  $s_{ij} = (p_i + p_j)^2$ , which are put into `feyntrp`'s replacement rules in the form `(sp[i,j], '(sij - ppi - ppj)/2)')` for  $0 \leq i < j \leq 2$ . The chosen phase space point is

$$\begin{aligned} p_0^2 = 1.1, \quad p_1^2 = 1.2, \quad p_2^2 = 1.3, \quad s_{01} = 2.1, \quad s_{02} = 2.2, \quad s_{12} = 2.3, \\ m_0^2 = 0.05, \quad m_1^2 = 0.06, \quad m_2^2 = 0.07, \quad m_3^2 = 0.08, \quad m_4^2 = 0.09, \quad m_5^2 = 0.1. \end{aligned} \quad (40)$$

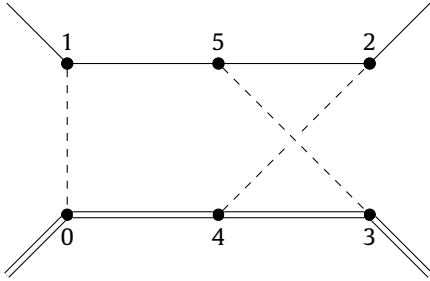
With additional settings  $\lambda = 1.24$  and  $N = 10^8$ , we find

```
Prefactor: gamma(3*eps + 2).
(Effective) kinematic regime: Minkowski (generic).
Finished in 5.12 seconds.
-- eps^0: [-10.8335 +/- 0.0084] + i * [-12.7145 +/- 0.0083]
-- eps^1: [ 47.971 +/- 0.059 ] + i * [-105.057 +/- 0.059 ]
-- eps^2: [ 413.05 +/- 0.23 ] + i * [ 7.29 +/- 0.23 ]
-- eps^3: [ 372.07 +/- 0.65 ] + i * [ 947.82 +/- 0.65 ]
-- eps^4: [-1412.36 +/- 1.45 ] + i * [1325.74 +/- 1.45 ]
-- eps^5: [-2726.00 +/- 2.67 ] + i * [-1295.36 +/- 2.69 ]
-- eps^6: [ 287.25 +/- 4.28 ] + i * [-3982.04 +/- 4.30 ]
```

We verified these numbers using `pySecDec`. The test machine's memory of 16 GBs was exhausted before `AMFlow` could finish the calculation. The examples in [84] indicate that using a computer with more memory might also make this 3-loop diagram accessible using `AMFlow`.

### 6.3. A 2-loop 4-point $\mu e$ -scattering diagram

We evaluate a non-planar, 2-loop 4-point diagram appearing in muon-electron scattering [85], which is finite in  $D = 6 - 2\epsilon$  dimensions. It was previously evaluated for vanishing electron mass in [86].



The dashed lines represent photons, the solid lines are electrons with mass  $m$ , and the double lines are muons with mass  $M$  (which is approximately 200 times larger than  $m$ ). The edge set is

```
edges = [((0,1), 1, '0'), ((0,4), 1, 'MM'), ((1,5), 1, 'mm'), ((5,2), 1, 'mm'),
         ((5,3), 1, '0'), ((4,3), 1, 'MM'), ((4,2), 1, '0')]
```

where `MM` and `mm` stand for  $M^2$  and  $m^2$  respectively. With a phase space point similar to that of [86, Section 4.1.2]

$$\begin{aligned} p_0^2 = M^2 = 1, \quad p_1^2 = p_2^2 = m^2 = 1/200, \quad s_{01} = -1/7, \\ s_{12} = -1/3, \quad s_{02} = 2M^2 - 2m^2 - s_{01} - s_{12} = 2.49 \end{aligned} \quad (41)$$

and settings  $\lambda = 1.29$ ,  $N = 10^8$ , the result becomes

```
Prefactor: gamma(2*eps + 1).
(Effective) kinematic regime: Minkowski (exceptional).
Finished in 6.53 seconds.
-- eps^0: [1.16483 +/- 0.00083] + i * [0.24155 +/- 0.00074]
-- eps^1: [5.5387 +/- 0.0086 ] + i * [2.2818 +/- 0.0093 ]
-- eps^2: [15.171 +/- 0.058 ] + i * [10.079 +/- 0.064 ]
-- eps^3: [ 28.02 +/- 0.32 ] + i * [ 28.17 +/- 0.28 ]
-- eps^4: [ 38.20 +/- 1.42 ] + i * [ 56.94 +/- 0.85 ]
```

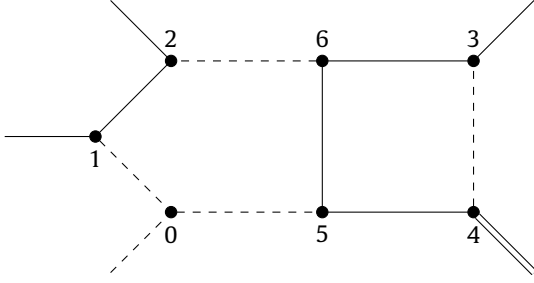
The momentum configuration is exceptional, so we cannot be sure that the generalized permutahedron property holds - see Section 3.3. In spite of that, `feyntrp` gives the correct numbers, which we confirmed using both `AMFlow` and `pySecDec`.

The leading order term differs from [86, eq. (4.20)] by roughly 10% due to our inclusion of the electron mass. We do, however, reproduce the computation in this reference if we set this mass to 0 in the `feyntrp` configuration.



#### 6.4. A QCD-like, 2-loop 5-point diagram

This example is a QCD-like,  $D = 6 - 2\epsilon$  dimensional, 2-loop 5-point diagram:



The dashed lines represent gluons, the solid lines are quarks each with mass  $m$ , and the double line is some off-shell momentum  $p_4^2 \neq 0$  fixed by conservation. The edge data are

```
edges = [((0,1), 1, '0'), ((1,2), 1, 'mm'), ((2,6), 1, '0'), ((6,3), 1, 'mm'),
          ((3,4), 1, '0'), ((4,5), 1, 'mm'), ((5,0), 1, '0'), ((5,6), 1, 'mm')]
```

where  $mm$  stands for  $m^2$ . Let us choose the phase space point

$$\begin{aligned} p_0^2 = 0, \quad p_1^2 = p_2^2 = p_3^2 = m^2 = 1/2, \quad s_{01} = 2.2, \quad s_{02} = 2.3, \\ s_{03} = 2.4, \quad s_{12} = 2.5, \quad s_{13} = 2.6, \quad s_{23} = 2.7, \end{aligned} \quad (42)$$

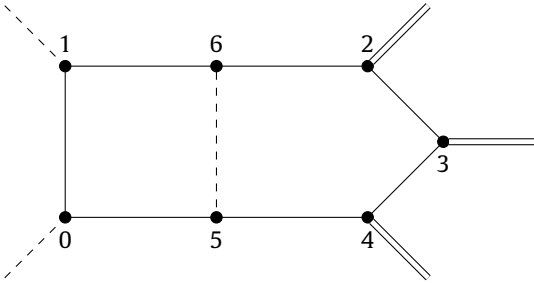
where again  $s_{ij} = (p_i + p_j)^2$ . Finally, setting  $\lambda = 0.28$ ,  $N = 10^8$ , we obtain

```
Prefactor: gamma(2*eps + 2).
(Effective) kinematic regime: Minkowski (exceptional).
Finished in 8.20 seconds.
-- eps^0: [0.06480 +/- 0.00078] + i * [-0.08150 +/- 0.00098]
-- eps^1: [0.4036 +/- 0.0045 ] + i * [ 0.3257 +/- 0.0035 ]
-- eps^2: [-0.7889 +/- 0.0060 ] + i * [ 0.957 +/- 0.016 ]
-- eps^3: [-1.373 +/- 0.030 ] + i * [-1.181 +/- 0.034 ]
-- eps^4: [ 1.258 +/- 0.088 ] + i * [-1.205 +/- 0.036 ]
```

The kinematic configuration is again exceptional. Nevertheless, `feyntrop` returns the correct numbers, which we verified with `pySecDec`.<sup>6</sup> We were not able to compute this diagram with `AMFlow` due to our memory constraints. As similarly intricate Feynman integrals can be evaluated with `AMFlow` using more memory (see [84]), these constraints are very likely the only obstruction for a crosscheck with `AMFlow`.

#### 6.5. Diagram contributing to triple Higgs production via gluon fusion

In this example, we evaluate the following diagram contributing to the process<sup>7</sup>  $gg \rightarrow HHH$  in  $D = 4 - 2\epsilon$  dimensions:



The dashed lines are massless propagators (representing gluons), the single solid lines are propagators containing the top quark mass, and the three external double lines are put on-shell to the Higgs mass. In this case, the list of edges reads

```
edges = [((0,1), 1, 'mm_top'), ((1,6), 1, 'mm_top'), ((5,6), 1, '0'),
          ((6,2), 1, 'mm_top'), ((2,3), 1, 'mm_top'), ((3,4), 1, 'mm_top'),
          ((4,5), 1, 'mm_top'), ((5,0), 1, 'mm_top')]
```

<sup>6</sup> An earlier version of this article wrongly stated that this computation was not verifiable with `pySecDec`. We thank both an anonymous referee and Vitaly Magerya for pointing this out to us.

<sup>7</sup> We thank Babis Anastasiou for suggesting this example.

with  $m_{\text{top}}^2$  being the square of the top quark mass,  $m_t^2$ .

Given  $s_{ij} := (p_i + p_j)^2$ , we employ the following kinematic setup:

$$\begin{aligned} p_0^2 = p_1^2 = 0, \quad p_2^2 = p_3^2 = p_4^2 = m_H^2, \\ s_{01} = 5m_H^2 - s_{02} - s_{03} - s_{12} - s_{13} - s_{23}. \end{aligned} \quad (43)$$

The kinematic space is then parameterized by  $(s_{02}, s_{03}, s_{12}, s_{13}, s_{23}, m_t^2, m_H^2)$ .

Let us evaluate this integral at the phase space point

$$\begin{aligned} m_t^2 = 1.8995, \quad m_H^2 = 1, \\ s_{02} = -4.4, \quad s_{03} = -0.5, \quad s_{12} = -0.6, \quad s_{13} = -0.7, \quad s_{23} = 1.8, \end{aligned} \quad (44)$$

which lies in the physical region, and has the physically relevant mass ratio  $m_t^2/m_H^2 = 1.8995$ . The remaining Mandelstam invariants are then fixed by momentum conservation to

$$(s_{01}, s_{04}, s_{14}, s_{24}, s_{34}) = (9.4, -1.5, -5.1, 7.2, 3.4).$$

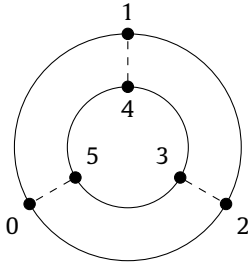
Setting  $\lambda = 0.64$  and  $N = 10^8$ , we get

```
Prefactor: gamma(2*eps + 4).
(Effective) kinematic regime: Minkowski (generic).
Finished in 8.12 seconds.
-- eps^0: [-0.0114757 +/- 0.0000082] + i * [0.0035991 +/- 0.0000068]
-- eps^1: [ 0.003250 +/- 0.000031 ] + i * [-0.035808 +/- 0.000041 ]
-- eps^2: [ 0.046575 +/- 0.000098 ] + i * [0.016143 +/- 0.000088 ]
-- eps^3: [ -0.01637 +/- 0.00017 ] + i * [ 0.03969 +/- 0.00016 ]
-- eps^4: [ -0.02831 +/- 0.00023 ] + i * [-0.00823 +/- 0.00024 ]
```

We were unable to evaluate this example in reasonable time with AMFlow. Again, adding more memory would likely solve this problem. With pySecDec we were able to confirm feyntrap's numbers within 3 hours<sup>8</sup> on a laptop, with relative errors around  $10^{-2}$ . Running feyntrap on the same laptop with  $10^8$  sampling points, we obtain the same numbers within 2.5 minutes and with relative errors of order  $10^{-3}$ .

#### 6.6. A QED-like, 4-loop vacuum diagram

Next we evaluate a QED-like, 4-loop vacuum diagram in  $D = 4 - 2\epsilon$  dimensions:



The dashed lines represent photons, and the solid lines are electrons of mass  $m$ . No analytic continuation is required in this case since there are no external momenta - the final result should hence be purely real. We specify

```
replacement_rules = []
```

in the code to indicate that all scalar products are zero.

The collection of edges is

```
edges = [((0,1), 1, 'mm'), ((1,2), 1, 'mm'), ((2,0), 1, 'mm'),
          ((0,5), 1, '0'), ((1,4), 1, '0'), ((2,3), 1, '0'),
          ((3,4), 1, 'mm'), ((4,5), 1, 'mm'), ((5,3), 1, 'mm')]
```

where mm stands for  $m^2$ . Choosing

```
phase_space_point = [('mm', 1)]
```

and setting  $\lambda = 0$ ,  $N = 10^8$ , we then find

<sup>8</sup> Three months after the initial version of this article was posted, a new version of pySecDec became available which is, in some cases, up to four times as efficient as the former version [83]. We postpone a systematic comparison of feyntrap with this new version to a future research project.

```

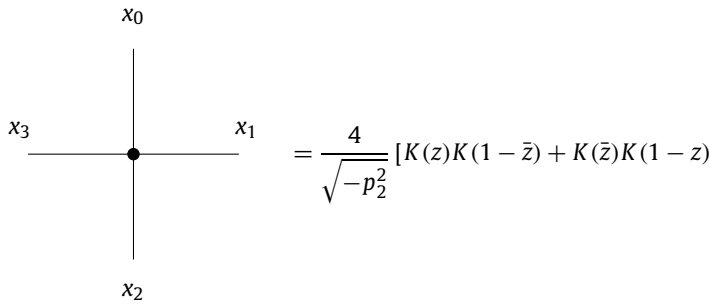
Prefactor: gamma(4*eps + 1).
(Effective) kinematic regime: Euclidean (generic).
Finished in 3.58 seconds.
-- eps^0: [3.01913 +/- 0.00047] + i * [0.0 +/- 0.0]
-- eps^1: [-7.0679 +/- 0.0021 ] + i * [0.0 +/- 0.0]
-- eps^2: [20.5399 +/- 0.0074 ] + i * [0.0 +/- 0.0]
-- eps^3: [-27.895 +/- 0.024 ] + i * [0.0 +/- 0.0]
-- eps^4: [62.043 +/- 0.074 ] + i * [0.0 +/- 0.0]
-- eps^5: [-59.46 +/- 0.23 ] + i * [0.0 +/- 0.0]
-- eps^6: [155.27 +/- 0.73 ] + i * [0.0 +/- 0.0]
-- eps^7: [-90.81 +/- 2.26 ] + i * [0.0 +/- 0.0]
-- eps^8: [403.78 +/- 6.71 ] + i * [0.0 +/- 0.0]

```

We were not able to verify this example with AMFlow or pySecDec within our memory constraints. However, Vitaly Magerya informed us that he was able to verify these numbers with pySecDec in under one hour using an only slightly larger computer.

### 6.7. An elliptic, conformal, 4-point integral

The final example is a 1-loop 4-point conformal integral with edge weights  $v_{1,\dots,4} = 1/2$  in  $D = 2$  dimensions, the result of which was computed in terms of elliptic  $K$  functions in [87, Sec. 7.2]:



$$= \frac{4}{\sqrt{-p_2^2}} [K(z)K(1-\bar{z}) + K(\bar{z})K(1-z)] \quad (45)$$

The denominator above differs from [87, eq. (7.6)] because we have used conformal symmetry to send  $x_3 \rightarrow \infty$ , thereby reducing the kinematic space to that of a 3-point integral. After identifying dual momentum variables  $x_i$  in terms of ordinary momenta as  $p_i = x_i - x_{i+1}$ , the conformal cross ratios, with the usual single-valued complex parameterization in terms of  $z$  and  $\bar{z}$ , read

$$z\bar{z} = \frac{p_0^2}{p_2^2}, \quad (1-z)(1-\bar{z}) = \frac{p_1^2}{p_2^2}. \quad (46)$$

In `feyntrop` we specify the associated 1-loop 3-point momentum space integral as

```
edges = [((0,1), 1/2, '0'), ((1,2), 1/2, '0'), ((2,0), 1/2, '0')]
```

where all internal masses are zero and edge weights are set to  $1/2$ .

We choose a momentum configuration in the Euclidean regime:

$$p_0^2 = -2, \quad p_1^2 = -3, \quad p_2^2 = -5. \quad (47)$$

Although `feyntrop` can compute integrals with rational edge weights in the Minkowski regime, it is most natural to study conformal integrals in the Euclidean regime.

With  $\lambda = 0$  and  $N = 10^8$ , we then obtain

```

(Effective) kinematic regime: Euclidean (generic).
Finished in 1.34 seconds.
-- eps^0: [9.97192 +/- 0.00027] + i * [0.0 +/- 0.0]

```

The result agrees with the analytic expression (45). This example also illustrates the high efficiency of `feyntrop` in the Euclidean regime where very high accuracies can be obtained quickly.

## 7. Conclusions and outlook

With this article we introduced `feyntrop`, a general tool to numerically evaluate quasi-finite Feynman integrals in the physical regime with sufficiently general kinematics. To do so, we gave a detailed classification of different kinematic regimes that are relevant for numerical integration. Moreover, we presented a completely projective integral expression for concretely  $i\epsilon$ -deformed Feynman integrals and their dimensionally regularized expansions. We used tropical sampling for the numerical integration, which we briefly reviewed, and we discussed the relevant issues on facet presentations of the Newton polytopes of Symanzik polynomials in detail. To be able to perform the numerical integration efficiently, we gave formulas and algorithms for the fast evaluation of Feynman integrals. To give a concise usage manual for `feyntrop` and to illustrate its capabilities, we gave numerous, detailed examples of evaluated Feynman integrals.

The most important restrictions of `feyntrop` are 1) it is not capable of dealing with Feynman integrals that have subdivergences (i.e. non-quasi-finite integrals) and 2) it is not capable of dealing with certain highly exceptional kinematic configurations.

The first restriction can be lifted by implementing an analytic continuation of the integrand in the spirit of [29,30,56] into `feyntrop`. Naively, preprocessing input integrals with such a procedure increases the number of Feynman integrals and thereby also the necessary computer time immensely. However, this proliferation of terms comes from the expansion of the derivatives of the  $\mathcal{U}$  and  $\mathcal{F}$  polynomials as numerators. This expansion can be avoided, because also the derivatives of  $\mathcal{U}$  and  $\mathcal{F}$  (mostly) have the generalized permutahedron property, and because we have fast algorithms to evaluate such derivatives. For instance, we derived a fast algorithm to evaluate the first and second derivatives of  $\mathcal{F}$  in Section 4.2. We postpone the elaboration and implementation of this approach to future work.

A promising approach to lift the second restriction is to try to understand the general shape of the  $\mathcal{F}$  polynomial's Newton polytope. Outside of the Euclidean and generic kinematic regimes, this polytope is not always a generalized permutahedron. In these exceptional kinematic situations, it can have new facets that cannot be explained by known facet presentations. It might be possible to explain these new facets with the help of the Coleman–Norton picture of infrared divergences [88] (see, e.g., [89] where explicit per-diagram factorization of Feynman integrals was observed in a position space based framework). An alternative approach to fix the issue is to implement the tropical sampling approach that requires a full triangulation of the respective Newton polytopes (see [1, Sec. 5]).

Besides this there are numerous, desirable, gradual improvements of `feyntrop` that we also postpone to future works. The most important such improvement would be to use the algorithm in conjunction with a *quasi-Monte Carlo* approach. The runtime to obtain the value of an integral up to accuracy  $\delta$  currently scales as  $\delta^{-2}$ , as is standard for a Monte Carlo method. Changing to a quasi-Monte Carlo based procedure would improve this scaling to  $\delta^{-1}$ .

Another improvement would be to find an entirely *canonical* deformation prescription. Currently, our deformation still relies on an external parameter that has to be fine-tuned to the respective integral. A canonical deformation prescription that does not depend on a free parameter would lift the burden of this fine-tuning from the user and would likely also produce better rates of convergence.

A more technical update of `feyntrop` would involve an implementation of the tropical sampling algorithm on GPUs or on distributed cluster systems. The current implementation of `feyntrop` is parallelized and can make use of all cores of a single computer. Running `feyntrop` on multiple computers in parallel is not implemented, but there are no technical obstacles to write such an implementation, which we postpone to a future research project.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The code is submitted as a tar.gz file and is made available at <https://github.com/michibo/feyntrop>.

## Acknowledgements

We thank Nima Arkani-Hamed, Aaron Hillman, Sebastian Mizera and Erik Panzer for helpful exchanges on facet presentations of Newton polytopes of Symanzik polynomials, Pierpaolo Mastrolia for stimulating discussions on applications to phenomenology, Yan-Qing Ma for helpful comments on the manuscript and Vitaly Magerya for comments and independently verifying our numbers in Example 6.6 using `pySecDec`. FT thanks Georgios Papathanasiou for continued support. HJM and FT thank the Institute for Theoretical Studies at the ETH Zürich for hosting the workshop ‘Tropical and Convex Geometry and Feynman integrals’ in August 2022, which was beneficial for the completion of this work. All authors thank the Institute for Advanced Studies, Princeton US, for hospitality during a stay in May 2023 where parts of this work were completed. MB was supported by Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation. Some of our calculations were carried out on the ETH Euler cluster.

## References

- [1] M. Borinsky, Ann. Inst. Henri Poincaré Comb. Phys. Interact. (2023), <https://doi.org/10.4171/AIHPD/158>, in press, arXiv:2008.12310.
- [2] G. Guennebaud, B. Jacob, et al., Eigen v3, <http://eigen.tuxfamily.org>, 2010.
- [3] D. Blackman, S. Vigna, ACM Trans. Math. Softw. 47 (2021) 36, 32.
- [4] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.
- [5] W. Jakob, J. Rhineland, D. Moldovan, pybind11 – seamless operability between C++11 and python, <https://github.com/pybind/pybind11>, 2017.
- [6] A. Meurer, C.P. Smith, M. Paprocki, O. Čertík, S.B. Kirpichev, M. Rocklin, et al., PeerJ Comput. Sci. 3 (2017) e103.
- [7] G. Heinrich, Phys. Rep. 922 (2021) 1, arXiv:2009.00516.
- [8] S.G. Karshenboim, Phys. Rep. 422 (2005) 1, arXiv:hep-ph/0509010.
- [9] J. Zinn-Justin, Quantum Field Theory and Critical Phenomena, vol. 171, Oxford University Press, 2021.
- [10] J.F. Donoghue, Phys. Rev. D 50 (1994) 3874, arXiv:gr-qc/9405057.
- [11] F. Brown, SIGMA 17 (2021) 103, arXiv:2101.04419.
- [12] Z. Bern, L.J. Dixon, D.C. Dunbar, D.A. Kosower, Nucl. Phys. B 425 (1994) 217, arXiv:hep-ph/9403226.
- [13] Z. Bern, L.J. Dixon, D.A. Kosower, Phys. Rev. D 73 (2006) 065013, arXiv:hep-ph/0507005.
- [14] K.G. Chetyrkin, F.V. Tkachov, Nucl. Phys. B 192 (1981) 159.
- [15] S. Laporta, Int. J. Mod. Phys. A 15 (2000) 5087, arXiv:hep-ph/0102033.
- [16] S. Bloch, H. Esnault, D. Kreimer, Commun. Math. Phys. 267 (2006) 181, arXiv:math/0510011.
- [17] F. Brown, Commun. Math. Phys. 287 (2009) 925, arXiv:0804.1660.
- [18] E. Panzer, Comput. Phys. Commun. 188 (2015) 148, arXiv:1403.3385.
- [19] E. Remiddi, Nuovo Cimento A 110 (1997) 1435, arXiv:hep-th/9711188.
- [20] J.M. Henn, Phys. Rev. Lett. 110 (2013) 251601, arXiv:1304.1806.
- [21] E. Panzer, Ann. Inst. Henri Poincaré Comb. Phys. Interact. 10 (2023) 31, arXiv:1908.09820.
- [22] F. Brown, Commun. Number Theory Phys. 11 (2017) 453, arXiv:1512.06409.

- [23] G.V. Dunne, M. Meynig, *Phys. Rev. D* 105 (2022) 025019, arXiv:2111.15554.
- [24] A.V. Smirnov, N.D. Shapurov, L.I. Vysotsky, *Comput. Phys. Commun.* 277 (2022) 108386, arXiv:2110.11660.
- [25] M. Borinsky, A.-L. Sattelberger, B. Sturmfels, S. Telen, *SIAM J. Appl. Algebra Geom.* (2022), <https://doi.org/10.1137/22M1490569>, in press, arXiv:2204.06414.
- [26] N. Arkani-Hamed, A. Hillman, S. Mizera, *Phys. Rev. D* 105 (2022) 125013, arXiv:2202.12296.
- [27] N. Arkani-Hamed, J. Trnka, *J. High Energy Phys.* 10 (2014) 030, arXiv:1312.2007.
- [28] N. Arkani-Hamed, Y. Bai, T. Lam, *J. High Energy Phys.* 11 (2017) 039, arXiv:1703.04541.
- [29] L. Nilsson, M. Passare, *J. Geom. Anal.* 23 (2013) 24.
- [30] C. Berkesch, J. Forsgård, M. Passare, *Mich. Math. J.* 63 (2014) 101.
- [31] K. Schultka, Toric geometry and regularization of Feynman integrals, arXiv:1806.01086.
- [32] I.M. Gel'fand, M.M. Kapranov, A.V. Zelevinsky, *Adv. Math.* 84 (1990) 255.
- [33] L. de la Cruz, *J. High Energy Phys.* 12 (2019) 123, arXiv:1907.00507.
- [34] R.P. Klausen, *J. High Energy Phys.* 04 (2020) 121, arXiv:1910.08651.
- [35] A. Klemm, C. Nega, R. Safari, *J. High Energy Phys.* 04 (2020) 088, arXiv:1912.06201.
- [36] V. Chestnov, F. Gasparotto, M.K. Mandal, P. Mastrolia, S.J. Matsubara-Heo, H.J. Munch, et al., *J. High Energy Phys.* 09 (2022) 187, arXiv:2204.12983.
- [37] F. Tellander, M. Helmer, *Commun. Math. Phys.* 399 (2023) 1021, arXiv:2108.01410.
- [38] T. Binoth, G. Heinrich, *Nucl. Phys. B* 585 (2000) 741, arXiv:hep-ph/0004013.
- [39] C. Bogner, S. Weinzierl, *Comput. Phys. Commun.* 178 (2008) 596, arXiv:0709.4092.
- [40] T. Kaneko, T. Ueda, *Comput. Phys. Commun.* 181 (2010) 1352, arXiv:0908.2897.
- [41] S. Borowka, G. Heinrich, S. Jahn, S.P. Jones, M. Kerner, J. Schlenk, et al., *Comput. Phys. Commun.* 222 (2018) 313, arXiv:1703.09692.
- [42] D.E. Soper, *Phys. Rev. D* 62 (2000) 014009, arXiv:hep-ph/9910292.
- [43] C. Anastasiou, A. Daleo, *J. High Energy Phys.* 10 (2006) 031, arXiv:hep-ph/0511176.
- [44] S. Catani, T. Gleisberg, F. Krauss, G. Rodrigo, J.-C. Winter, *J. High Energy Phys.* 09 (2008) 065, arXiv:0804.3170.
- [45] Z. Capatti, V. Hirschi, D. Kermanschah, B. Ruijl, *Phys. Rev. Lett.* 123 (2019) 151602, arXiv:1906.06138.
- [46] X. Liu, Y.-Q. Ma, C.-Y. Wang, *Phys. Lett. B* 779 (2018) 353, arXiv:1711.09572.
- [47] M.K. Mandal, X. Zhao, *J. High Energy Phys.* 03 (2019) 190, arXiv:1812.03060.
- [48] X. Liu, Y.-Q. Ma, *Comput. Phys. Commun.* 283 (2023) 108565, arXiv:2201.11669.
- [49] M. Hidding, *Comput. Phys. Commun.* 269 (2021) 108125, arXiv:2006.05510.
- [50] T. Armadillo, R. Bonciani, S. Devoto, N. Rana, A. Vicini, *Comput. Phys. Commun.* 282 (2023) 108545, arXiv:2205.03345.
- [51] I. Dubovyk, A. Freitas, J. Gluza, K. Grzanka, M. Hidding, J. Usovitsch, *Phys. Rev. D* 106 (2022) L111301, arXiv:2201.02576.
- [52] Z.-F. Liu, Y.-Q. Ma, *Phys. Rev. Lett.* 129 (2022) 222001, arXiv:2201.11637.
- [53] T. Binoth, J.P. Guillet, G. Heinrich, E. Pilon, C. Schubert, *J. High Energy Phys.* 10 (2005) 015, arXiv:hep-ph/0504267.
- [54] R. Pittau, B. Webber, *Eur. Phys. J. C* 82 (2022) 55, arXiv:2110.12885.
- [55] S. Mizera, S. Telen, *J. High Energy Phys.* 08 (2022) 200, arXiv:2109.08036.
- [56] A. von Manteuffel, E. Panzer, R.M. Schabinger, *J. High Energy Phys.* 02 (2015) 120, arXiv:1411.7392.
- [57] B. Kol, M. Smolkin, *Class. Quantum Gravity* 25 (2008) 145011, arXiv:0712.4116.
- [58] N. Nakanishi, *Graph Theory and Feynman Integrals*, Gordon and Breach, 1971.
- [59] S. Weinzierl, *Feynman Integrals*, Springer, Cham, 2022, arXiv:2201.03593.
- [60] R.J. Eden, P.V. Landshoff, D.I. Olive, J.C. Polkinghorne, *The Analytic S-Matrix*, Cambridge University Press, 1966.
- [61] Z. Nagy, D.E. Soper, *Phys. Rev. D* 74 (2006) 093006, arXiv:hep-ph/0610028.
- [62] H.S. Hannesdottir, S. Mizera, *What Is the  $\mathfrak{t}$  for the S-Matrix?*, Springer, 2023, arXiv:2204.02988.
- [63] D. MacLagan, B. Sturmfels, *Introduction to Tropical Geometry*, Graduate Studies in Mathematics, vol. 161, American Mathematical Society, Providence, RI, 2015.
- [64] A. Postnikov, *Int. Math. Res. Not.* (2009) 1026.
- [65] M. Aguiar, F. Ardila, *Hopf monoids and generalized permutahedra*, arXiv:1709.07504.
- [66] K.G. Chetyrkin, V.A. Smirnov, *Theor. Math. Phys.* 56 (1983) 770.
- [67] E. Speer, *Ann. IHP, Phys. Théor.* 23 (1975) 1.
- [68] R. Beekveldt, M. Borinsky, F. Herzog, *J. High Energy Phys.* 07 (2020) 061, arXiv:2003.04301.
- [69] V.A. Smirnov, *Analytic Tools for Feynman Integrals*, Springer Tracts in Modern Physics, vol. 250, Springer, Heidelberg, 2012.
- [70] E. Panzer, *Feynman integrals and hyperlogarithms*, Ph.D. thesis, Humboldt U, 2015, arXiv:1506.07243.
- [71] A. Hillman, S. Mizera, E. Panzer, *Personal communication*, January–February 2023.
- [72] S. Fujishige, N. Tomizawa, *J. Oper. Res. Soc. Jpn.* 26 (1983) 309.
- [73] B.D. McKay, *Eur. J. Comb.* 4 (1983) 149.
- [74] R.A. Horn, C.R. Johnson, *Matrix Analysis*, second ed., Cambridge University Press, Cambridge, 2013.
- [75] D.A. Spielman, S.-H. Teng, *SIAM J. Matrix Anal. Appl.* 35 (2014) 835, arXiv:cs/0607105.
- [76] M. Borinsky, O. Schnetz, *J. High Energy Phys.* 08 (2022) 291, arXiv:2206.10460.
- [77] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, J. McDonald, *Parallel Programming in OpenMP*, Morgan Kaufmann, 2001.
- [78] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, et al., in: F. Loizides, B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 2016, pp. 87–90.
- [79] A.V. Smirnov, F.S. Chuharev, *Comput. Phys. Commun.* 247 (2020) 106877, arXiv:1901.07808.
- [80] R.N. Lee, *Presenting LiteRed: a tool for the Loop InTEgrals REDuction*, arXiv:1212.2685.
- [81] R.N. Lee, *J. Phys. Conf. Ser.* 523 (2014) 012059, arXiv:1310.1145.
- [82] X. Guan, X. Liu, Y.-Q. Ma, W.-H. Wu, <https://gitlab.com/multiloop-pku/blade/>, (Accessed 17 May 2023).
- [83] G. Heinrich, S.P. Jones, M. Kerner, V. Magerya, A. Olsson, J. Schlenk, *Numerical scattering amplitudes with pySecDec*, arXiv:2305.19768.
- [84] X. Liu, Y.-Q. Ma, *Phys. Rev. D* 105 (2022) L051503, arXiv:2107.01864.
- [85] A. Broggio, et al., *J. High Energy Phys.* 01 (2023) 112, arXiv:2212.06481.
- [86] S. Di Vita, S. Laporta, P. Mastrolia, A. Primo, U. Schubert, *J. High Energy Phys.* 09 (2018) 016, arXiv:1806.08241.
- [87] L. Corcoran, F. Loebbert, J. Miczajka, *J. High Energy Phys.* 04 (2022) 131, arXiv:2112.06928.
- [88] S. Coleman, R.E. Norton, *Nuovo Cimento* 38 (1965) 438.
- [89] M. Borinsky, Z. Capatti, E. Laenen, A. Salas-Bernárdez, *J. High Energy Phys.* 01 (2023) 172, arXiv:2210.05532.