APPLICATION OF THE PSLQ THEOREM TO FEYNMAN INTEGRALS

DIEGO ANDRÉS RIVERA ORONA

Abstract. This pedagogical paper seeks to present two unlikely friends: Diophantine approximation and Quantum Field Theory calculations. We begin with a quick motivation and a 30,000-foot view of Feynman diagrams and integrals. We then segue into the PSLQ algorithm, which makes this friendship possible in the first place. We explain both the advantages and common uses of the algorithm, as well as a complete definition and a geometric interpretation. We then give a worked-out toy example of the algorithm as well as examples from the literature of the algorithm being applied to QFT calculations.

Contents

1. Introduction

The field of Diophantine approximation has a rich history and a close relationship to number theory and analysis. However, the field is often not thought of as applicable to physics. Quantum field theory (QFT), particularly in its study of Feynman integrals, provides an interesting bridge between the two. This paper provides a pedagogical connection between Diophantine approximation and QFT calculations through the use of the PSLQ algorithm. Physicists use this algorithm to establish highly probable analytic expressions for values obtained from computer-aided QFT calculations.

Quantum mechanics broadly seeks to understand the nature, behavior, and interaction of subatomic particles–and in the non-quantum limit–all matter. This task is often much more complicated than it might at first seem. Such interactions are typically studied using scattering amplitudes.

Date: May 12, 2026.

Scattering amplitudes are, generally speaking, complex numbers associated with an interaction between two or more particles. The main issue is that there are many different interactions (also called paths) to get from *the* initial configuration to *a* final configuration. The Hamiltonian (the operator which time-evolves the state) can be expanded perturbatively using a 'perturbative parameter' (expansion variable) λ. Each power of λ is more and more suppressed, but consists of more and more complex 'Feynman integrals'. These integrals can be graphically expressed as 'Feynman diagrams'. Each integral has a corresponding 'Feynman amplitude', whose square gives the probability of that configuration happening.

With a few simple exceptions, most Feynman integrals can only be evaluated numerically using computers. For many experimental physics applications this could suffice and we could stop here, but we ought not to. A famous example of this is the magnetic moment of the electron. The electron's dipole magnetic moment (how much it behaves like a bar magnet) is related to its angular momentum by a constant g, which was 'naively' expected to be $g = 2$. In reality, it has been experimentally found to be $g = 2.00231930436092(36)$. The discrepancy arises because of effects from quantum electrodynamics. The corrections are theoretically computed using Feynman diagrams. The sixth order correction consists of evaluating 3-loop diagrams, which produces terms including $\zeta(3)$, where $\zeta(k)$ is the Riemann ζ-function, defined as

$$\zeta(k) = \sum_{n=1}^{\infty} \frac{1}{n^k}. \tag{1.1}$$

3-loop diagrams are also the first Feynman diagrams to contain non-trivial topologies [Kre00]. Through this example, we see that these Feynman integrals, which often evaluate to transcendental numbers, represent fundamental quantities about fundamental interactions between fundamental particles, so we expect the numbers to be fundamental.

This paper will provide a concise introduction to both Feynman integrals and the PSLQ algorithm, which is used often by particle physicists to recover the analytic structure of numerically-calculated Feynman integrals. The PSLQ algorithm, developed in the early 90s, uses partial sums and lower triangular orthogonal decomposition of matrices to find integer relations. It is numerically stable, and requires a number of iterations which varies as a polynomial on the length of the input vectors. Due to its efficiency and reliability, it has become a staple in particle physics calculations (see [AB21],[Wei22, Ch. 15.4]). The purpose of the paper is thus mainly educational: mathematicians might not be aware of the application, while physicists might not think much about the algorithm beyond its four-letter name. Section 2 will provide necessary background information about Feynman integrals and how they are postulated from Feynman diagrams. We then shift gears into Section 3, which gives background on integer relations, including common applications. Section 4 introduces the PSLQ algorithm, along with theorems showing its capabilities and a short worked example. Finally, Section 5 comes back to Feynman integrals and presents some examples from the literature.

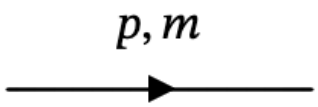

FIGURE 1. Propagating particle with momentum p and mass m

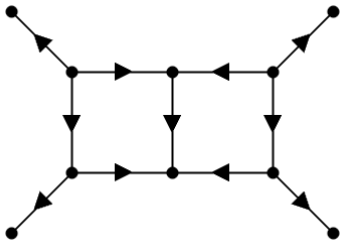

FIGURE 2. Oriented 2-loop graph with 11 vertices and 10 edges.

2. Feynman Integrals

This section will give a quick introduction to Feynman Diagrams and how we get Feynman integrals from them. For the sake of simplicity, we will work only with connected graphs. A graph G is made up of vertices and edges, where an edge simply connects two vertices. An *oriented graph* has a direction chosen for every edge. Each edge, and its orientation, represents a particle propagating in space-time with a corresponding momentum and mass, as seen in Fig. 1.

Such an edge is equivalent to the *propagator* $-i/(m^2 - p^2)$, where $m^2 - p^2$ is the magnitude of the particle's momentum with respect to the Minkowski metric $\eta_{\mu\nu} = (+1, -1, \ldots, -1)$–for more information refer to [PS95, Ch. 4]. For our purposes, the propagator is simply a term in the Feynman integral. The *valency* (physicist lingo for degree) of a vertex is defined as the number of edges to which it is connected. A graph G will have *external edges* (which are connected to a vertex with valency 1) and *internal edges*. External edges can be thought of as the momenta that's controlled and measured in a lab, while the internal momenta might or might not be determined. In some cases, momentum conservation at each vertex is enough to determine every momenta in the graph, and so there is no need to compute the Feynman integral. In other cases, this is not the case. This is determined by the *loop number* l, defined as

$$l = n - r + 1, \tag{2.1}$$

where n is the number of vertices and r the number of edges: l is the number of *independent momenta*. For example, in Fig. 2 we see 11 vertices and 10 edges, which means that $l = 2$. Thus, there are 2 independent momenta. Quantum field theory (QFT) tells us (see [PS95, Ch. 4] for more) that we must integrate over all possible independent momenta (from $-\infty$ to $+\infty$). We use the measure used in [Wei22, Ch. 2]:

$$\int \frac{d^D k_r}{i\pi^{D/2}}, \tag{2.2}$$

where D is the number of space time dimensions and $1 \leq r \leq l$. Here k_r is a vector in $\mathbb{R}^D$: k_r^0 is the time component of the momentum and $k_r^1, \ldots, k_r^{D-1}$ are the space components. Integration happens over each component (hence the D-dimensional integral).

Now, we identify internal momenta q_j with three numbers: q_j, m_j, ν_j. As above, q_j, m_j represent the momentum and the mass, while ν_j represents the propagator power. If a vertex's valency is 2, then the vertex simply increases the propagator power, but there is no particle interaction. Momentum conservation at each vertex of valency > 1 allows us to express the internal momenta q_j as linear combinations of the external momenta p_j (known) and the independent momenta k_j. Thus, we integrate over the *independent* momenta, but our integrand consists of propagators of the *internal* momenta (which are in turn expressed in terms of the external and independent momenta).

Putting this all together, we get that the general expression for the Feynman integral of a graph with l independent momenta and n internal momenta is

$$I = e^{l\epsilon\gamma}(\mu^2)^{\nu - lD/2} \int \prod_{r=1}^{l} \frac{d^D k_r}{i\pi^{D/2}} \prod_{j=1}^{n} \frac{1}{(m_j^2 - q_j^2)^{\nu_j}}, \tag{2.3}$$

where the prefactor is a quasi-made-up term used by physicists to cancel constants and make the final answer dimensionless (see [Wei22, Ch. 2] for a complete explanation on this prefactor): we shall not worry about it in this paper. For small l, n and for no more than 1-loop diagrams, the integrals can be manipulated to give values of the Riemann ζ-function. These can be calculated directly for simple diagrams, but this becomes unfeasible for higher order diagrams. However, we can intuitively *expect* higher order integrals to be expressible in terms of these numbers. This is precisely shown in [AB21] using the PSLQ algorithm.

Heuristically speaking, Feynman integrals, as seen in (2.3), often produce values related to logarithms, ζ-functions, polylogarithms, etc. (See 5.1 for more on why this happens.) Most Feynman integrals, however, can only be calculated numerically, which completely obfuscates any analytic structure. The PSLQ algorithm helps to recover this structure *ex post facto*.

3. Integer Relations and Integer Relation Algorithms

Definition 3.1. A vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^t$ of real numbers has an integer relation if there exists a nonzero vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)^t$ of integers such that $\boldsymbol{a} \cdot \boldsymbol{x} = 0$.

Example 3.2. The vector $\boldsymbol{x} = \left(\frac{1+\sqrt{5}}{2}, \frac{1-\sqrt{5}}{2}, 2\right)^t$ has an integer relation because $\boldsymbol{a} = (2, 2, -1)^t$ satisfies

$$\boldsymbol{a} \cdot \boldsymbol{x} = (1 + \sqrt{5}) + (1 - \sqrt{5}) - (2) = 0. \tag{3.1}$$

Integer relation algorithms are algorithms that, given a computer's numerical precision, will determine the vector $\boldsymbol{a}$ for a given $\boldsymbol{x}$, or will determine there exists

no such $\boldsymbol{a}$ within given bounds. There are two standard applications of the PSLQ algorithm: subset sum problems and polynomial roots.

EXAMPLE 3.3. (Subset Sum Problems) The algorithm determines whether a subset of the integers in $\boldsymbol{x}$ adds up to zero or some other number: $\boldsymbol{a}$ would be composed entirely of ones and zeroes.

EXAMPLE 3.4. (Polynomial Roots) The algorithm determines if some real number ξ is a root to a polynomial in integer coefficients with degree $n-1$ or less: $\boldsymbol{x} = (1, \xi, \ldots, \xi^{n-1})$ and we seek for a corresponding $\boldsymbol{a}$. In other words, the algorithm determines whether ξ is algebraic with degree $n-1$ or less.

The question of finding integer relations has been tackled as far back as Euclid: the Euclidean algorithm can be used to determine a continued fraction expansion for any real number. If the expansion terminates then the number is rational. But the question of finding integer relations is much deeper and important than subset sum problems: it is a multi-dimensional version of the most basic question in Diophantine approximation.

3.1. **Connection to Diophantine Approximation.** The "first day of class problem" in Diophantine approximation is finding a pair of integers p, q such that

$$\left|\xi - \frac{p}{q}\right| \tag{3.2}$$

is as small as possible for some real number ξ. But what if we rewrote this in terms of integer relations? Say we have a vector $\boldsymbol{x} = (\xi, -1)^t$, then we are looking for an integer relation $\boldsymbol{m} = (q, p)^t$ such that $\boldsymbol{m} \cdot \boldsymbol{x}$ is zero. We thus see that finding integer relations is a higher dimensional version of this primordial Diophantine approximation problem. However, things get a bit complicated now.

In the standard Diophantine problem, the possibility of finding an integer relation is summed up in one question: *is ξ rational?* As seen in the next section, $\xi = 3^{1/4} - 2^{1/4}$ is an irrational number, but it does satisfy an integer relation problem. These are precisely *algebraic numbers*, which form a great subfield of Diophantine approximation.

3.2. **Efficiency and Numerical Stability.** In the late 80s and 90s, two algorithms rose as principal contenders in the field of integer relation algorithms. The first was the HJLS algorithm, which is noteworthy for being the first integer relation algorithm to recover a relation with a number of iterations bounded by a polynomial in n. However, it is very numerically unstable, and thus the PSLQ algorithm was introduced by Helaman R. P. Ferguson and David H. Bailey in 1992 [FB92]. The PSLQ algorithm also terminates with a number of iterations bounded by a polynomial in n.

DEFINITION 3.5. (Numerical Instability) We say that an algorithm is numerically unstable if it magnifies errors in measurement or initial conditions such that obtaining a correct result necessitates internal numerical precision far exceeding that of the input values.

Given that we are dealing with algorithms, which are run on computers, we must be wary of the degree of internal numerical precision needed to get results.

Example 3.6. (Numerical Instability of the HJLS Algorithm) The authors of [FB92] provide the following test case. Using one hundred digits of $\xi = 3^{\frac{1}{4}} - 2^{\frac{1}{4}}$ as input, one finds that the vector $\boldsymbol{x} = (1, \xi, \ldots, \xi^{16})^t$ has the integer relation $\boldsymbol{a} = (1, 0, 0, 0, -3860, 0, 0, 0, -666, 0, 0, 0, -20, 0, 0, 0, 1)^t$. However, the machine must use over 10,000 digits of internal precision to arrive at this vector (more than the square of the input precision!).

4. The PSLQ Algorithm

We now move on to the PSLQ algorithm itself and how it works geometrically. Given the educational nature of this paper, we shall relegate a complete and rigorous proof to the following papers: [FB92], [FBA99], and [Bor02]. The first reference is Ferguson and Bailey's original paper introducing PSLQ; the second is a later paper by Ferguson and Bailey which further explicates many details; and the third is a more geometrical formulation by Borwein. I recommend Borwein's as a first introduction and then Ferguson and Bailey's second paper as a next step.

We begin with two basic objects: $\boldsymbol{x}$ and $\mathbb{Z}^n$. The former is our input vector in $\mathbb{R}^n$, while the latter is the lattice of integer points. The hyperplane orthogonal to our vector is defined as

$$\boldsymbol{x}^\perp = \{\boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{v} \cdot \boldsymbol{x} = 0\}. \tag{4.1}$$

The main takeaway is that the PSLQ algorithm searches (although does not necessarily find) for the smallest vector in the intersection of $\boldsymbol{x}^\perp$ and $\mathbb{Z}^n$. It does this by constructing a sequence of bases $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ of the lattice (that is, a basis of $\mathbb{R}^n$ which lies in $\mathbb{Z}^n$) which converges to the line $\mathbb{R}\boldsymbol{x}$. This, in turn, will produce a basis of $\boldsymbol{x}^\perp$ which will possibly contain a vector in $\mathbb{Z}^n$.

Let us begin with a basis $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ of vectors in the lattice. We define A as the matrix with $\boldsymbol{a}_i^t$ as its i-th row vector. A is a matrix in $\mathrm{GL_n}(\mathbb{Z})$, which means that $\det A = \pm 1$. We also define B as its inverse, thus B is also in $\mathrm{GL_n}(\mathbb{Z})$ and thus also has integer values. (Since the row vectors of A are linearly independent, we know A is invertible.) Define $\boldsymbol{b}_j$ as the j-th row vector of B. We thus see that

$$(AB)_{i,j} = (\boldsymbol{a}_i \cdot \boldsymbol{b}_j) = \delta_{i,j}, \tag{4.2}$$

which means that the 'closer' $\boldsymbol{a}_i$ is to $\boldsymbol{x}$ (the larger $\boldsymbol{a}_i \cdot \boldsymbol{x}$ is), the closer $\boldsymbol{b}_j$ is to $\boldsymbol{x}^\perp$.

We now define H as the $n \times (n-1)$ matrix whose column vectors $\boldsymbol{h}_j$ form a basis of $\boldsymbol{x}$. We also, define $H' = AH$, such that $h'_{i,j} = \boldsymbol{a}_i \cdot \boldsymbol{h}_j$. We now come to one of the most important definitions.

Definition 4.1. (Lower Trapezoidal) An $m \times n$ matrix C is lower trapezoidal if $m > n$ and each $c_{i,j} = 0$ is $i < j$.

Now, simply forcing the vectors $\boldsymbol{a}_i$ to be closer to $\boldsymbol{x}$ will not guarantee that we get a $\boldsymbol{b}_j$ in $\boldsymbol{x}^\perp$, but we can now set a bound on the size of a possible relation.

Theorem 4.2. *Let A be an invertible $n \times n$ matrix with integer coefficients, $\boldsymbol{x}$ a vector in $\mathbb{R}^n$, and H an $n \times (n-1)$ matrix with column vectors that form an orthonormal basis of $\boldsymbol{x}^\perp$. If $H' = AH$ is lower trapezoidal with each diagonal entry non-zero, then*

$$\frac{1}{\max_{1 \le i \le n-1} |h'_{i,i}|} \le |\boldsymbol{m}|, \tag{4.3}$$

for any integer relation $\boldsymbol{m}$ of $\boldsymbol{x}$.

Proof. For any integer relation $\boldsymbol{m}$, $HH^t\boldsymbol{m} = \boldsymbol{m}$ since HH^t is the projection matrix onto $\boldsymbol{x}^\perp$ and $\boldsymbol{m}$ is in $\boldsymbol{x}^\perp$ by definition. Thus, we have that $A\boldsymbol{m} = H'(H^t\boldsymbol{m})$. Let us define $\boldsymbol{a}_i^t$ as the i-th row vector of A, $\boldsymbol{h}_j^t$ as the j-th row vector of H^t and $h'_{j,j}$ as the j-th diagonal element of H'. We also know that A is invertible so $A\boldsymbol{m} \neq 0$.

Now, let j be the smallest integer such that $\boldsymbol{a}_j \cdot \boldsymbol{m} \neq 0$, then $\boldsymbol{a}_k \cdot \boldsymbol{m} = 0$ for $1 \leq k < j$. We now see that

$$A\boldsymbol{m} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{a}_j \cdot \boldsymbol{m} \\ \vdots \\ \boldsymbol{a}_n \cdot \boldsymbol{m} \end{bmatrix} = H'(H^t\boldsymbol{m}) = \begin{bmatrix} * & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & \cdots & * \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \cdot \boldsymbol{m} \\ \vdots \\ \boldsymbol{h}_{j-1} \cdot \boldsymbol{m} \\ \boldsymbol{h}_j \cdot \boldsymbol{m} \\ \vdots \\ \boldsymbol{h}_n \cdot \boldsymbol{m} \end{bmatrix}, \tag{4.4}$$

which implies that $\boldsymbol{h}_k \cdot \boldsymbol{m} = 0$ for $1 \leq k < j$ and $\boldsymbol{a}_j \cdot \boldsymbol{m} = (h'_{j,j})(\boldsymbol{h}_j \cdot \boldsymbol{m})$. Now, $\boldsymbol{a}_j \cdot \boldsymbol{m}$ is a non-zero integer since $\boldsymbol{a}_j$ forms the lattice basis that is 'close' to $\boldsymbol{x}$, while $\boldsymbol{m}$ forms part of the basis of $\boldsymbol{x}^\perp$. Thus, we have that

$$1 \leq |h'_{j,j}| \leq |h'_{j,j}||\boldsymbol{m}||, \tag{4.5}$$

where the last inequality follows because the projection of $\boldsymbol{m}$ onto $\boldsymbol{x}^\perp$ cannot be longer than $\boldsymbol{m}$. We thus get that

$$\frac{1}{\max_{1 \leq j \leq n-1} |h'_{j,j}|} \leq |\boldsymbol{m}|, \tag{4.6}$$

for any integer relation $\boldsymbol{m}$ of $\boldsymbol{x}$. □

We now present the algorithm as formulated by Borwein in [Bor02].

4.1. **The Algorithm Itself.** The algorithm takes two inputs: the vector $\boldsymbol{x} = (x_1, \ldots, x_n)^t$ and a constant $T \geq 1$. We also assume that $\boldsymbol{x}$ is a unit vector: that is, $s_1 = |\boldsymbol{x}| = 1$. We can do this since normalizing $\boldsymbol{x}$ does not change whether $\boldsymbol{m}$ corresponds to an integer relation:

$$\boldsymbol{m} \cdot \boldsymbol{x} = 0 = \frac{1}{|\boldsymbol{x}|}(\boldsymbol{m} \cdot \boldsymbol{x}) = \boldsymbol{m} \cdot \frac{\boldsymbol{x}}{|\boldsymbol{x}|}. \tag{4.7}$$

The algorithm returns either an integer relation for $\boldsymbol{x}$ along with a lower bound on the norm of the shortest integer relation or a lower bound ($\geq T$) on the norm of any possible relation for $\boldsymbol{x}$.

This formulation of the algorithm comes from [Bor02, p. 160], while the original paper is [FB92].

4.1.1. *Step 1.* We initialize by fixing a constant $\gamma > \sqrt{4/3}$: more on this later. Let the matrices A, B both be the identity matrix. As before, the i-th row vector of A is $\boldsymbol{a}_i^t$ and the j-th column vector of B is $\boldsymbol{b}_j$. Recall that the $n-1$ column vectors of H form an orthonormal basis of $\boldsymbol{x}^\perp$. We now want to encode information about $\boldsymbol{x}$ into the basis of $\boldsymbol{x}^\perp$. We do this using partial sums of squares:

$$h_{ij} = \begin{cases} 0 & 1 \leq i < j \leq n-1 \\ \frac{s_{i+1}}{s_i} & 1 \leq i = j \leq n-1 \\ -\frac{x_i x_j}{s_j s_{j+1}} & 1 \leq j < i \leq n \end{cases}, \qquad s_j^2 = \sum_{j \leq k \leq n} x_k^2. \tag{4.8}$$

Remark 4.3. PSLQ obtains the *PS* in its name from its use of Partial Sums of squares.

Let $\boldsymbol{h}'_i$ be the i-th row vector of H' and $\boldsymbol{h}_i$ the i-th column vector of H. It is not difficult to check that indeed, $\boldsymbol{h}_i \cdot \boldsymbol{x} = 0$ for all i and $\boldsymbol{h}_i \cdot \boldsymbol{h}_j = \delta_{i,j}$, as we wanted.

Remark 4.4. PSLQ obtains the *LQ* in its name from the lower trapezoidal orthogonal (LQ decomposition) of H, H'.

4.1.2. *Step 2.* We now want to size reduce H'. We do this recursively. For i from 2 to n and for j from $i-1$ to 1, we set $t = \lfloor h'_{i,j}/h'_{j,j} \rceil$ ($\lfloor \cdot \rceil$ is the closest integer function) and replace $\boldsymbol{a}_i$ for $\boldsymbol{a}_i - t\boldsymbol{a}_j$, $\boldsymbol{b}_j$ for $\boldsymbol{b}_j + t\boldsymbol{b}_i$, and $\boldsymbol{h}'_i$ for $\boldsymbol{h}'_i - t\boldsymbol{h}'_j$.

By reducing the values of $|h'_{i,i}|$, we reduce an upper bound on the values of $||\text{proj}_{\boldsymbol{x}^\perp} \boldsymbol{a}_i||$ and increase a lower bound on the size of the smallest possible norm for any integer relation of $\boldsymbol{x}$.

4.1.3. *Step 3.* We now present the algorithm's main iteration. We first choose an r such that $\gamma^i |h'_{i,i}|$ is maximal when $i = r$. We now repeat the following until either $1/\max |h'_{i,i}| \geq T$ (and thus the size of all integer relations $\boldsymbol{m}$ is above the bound T), or both $h'_{n,n-1} = 0$ and $r = n-1$:

(1) Let $\alpha = h'_{r,r}$, $\beta = h'_{r+1,r}$, and $\lambda = h'_{r+1,r+1}$.
    Then interchange rows $\boldsymbol{a}^t_r$ and $\boldsymbol{a}^t_{r+1}$ of A, columns $\boldsymbol{b}_r$ and $\boldsymbol{b}_{r+1}$ of B, and rows $\boldsymbol{h}'_r$ and $\boldsymbol{h}'_{r+1}$ of H'.
(2) If $r = n-1$, then H' is still lower trapezoidal. In this case, the value of $|h_{n-1,n-1}|$ was reduced by at least a factor of 2.
    If $r < n-1$, then H' is no longer trapezoidal. We fix this by replacing H by HQ and H' by $H'Q$, where Q is the $(n-1) \times (n-1)$ unitary matrix defined as follows:
    Set $Q = I_{n-1}$ and let $\delta = \sqrt{\beta^2 + \lambda^2}$. Then set $q_{r,r} = q_{r+1,r+1} = \beta/\delta$ and $q_{r+1,r} = -q_{r,r+1} = \lambda/\delta$.
    In addition to setting $h'_{r,r+1} = 0$, this also sets $h'_{r,r} = \delta$ and $h'_{r+1,r+1} = -\alpha\lambda/\delta$.
(3) Size reduce as in *Step 2*.
(4) Choose r such that $\gamma^i |h'_{i,i}|$ is maximal as above.

4.1.4. *Step 4.* Return $1/\max |h'_{i,i}|$ as a lower bound on the norm of any integer relation for $\boldsymbol{x}$. If $r = n-1$ and $h'_{n,n-1} = 0$, then return $\boldsymbol{b}_{n-1}$ as an integer relation for $\boldsymbol{x}$.

Remark 4.5. In *Step 1* we set the constant γ; this requires some explanation. In *Step 2*, we need the value of $|h'_{r,r}|$ to be reduced. This only happens if $\gamma > \sqrt{4/3}$. For more details, refer to [Bor02, Equation B.1].

4.2. **Algorithm Explanation.** As discussed, *Step 2* progressively reduces the value of $|h'_{i,i}|$. These, however, will never equal zero. We begin with all values of $h'_{i,i}$ being non-zero, and we replace the values of $h'_{r,r}$ and $h'_{r+1,r+1}$ with other non-zero values for $r < n-1$. Thus, the only way to obtain a vanishing $h'_{i,i}$ is when we interchange rows h'_{n-1} and h'_n of H' when $h_{n,n-1} = 0$. But at this point we exit the algorithm (see *Step 3*) and so we worry not about dividing by zero in setting $t = \lfloor h'_{i,j}/h'_{j,j} \rceil$ during the reduction of H'.

Now, the most important part is seeing how we extract the integer relation from B. If the algorithm terminates with $h_{n,n-1} = 0$, then the column vector $\boldsymbol{b}_{n-1}$ of

B is an integer relation for $\boldsymbol{x}$. We see this thanks to the following expressions: $\boldsymbol{x}^t H = 0$ (column vectors of H give a basis for $\boldsymbol{x}^\perp$), $BA = I$, $AH = H'$, and $h'_{n-1,n-1} \neq 0$. This gives

$$\mathbf{0} = \boldsymbol{x}^t A^{-1} AH = \boldsymbol{x}^t B H' = [\boldsymbol{x} \cdot (B\boldsymbol{h}'_1), \ldots, \boldsymbol{x} \cdot (B\boldsymbol{h}'_{n-1})], \tag{4.9}$$

where the only non-zero entry in $\boldsymbol{h}'_{n-1}$ is $h'_{n-1,n-1}$. We thus see that this last entry can also be written as

$$0 = \boldsymbol{x} \cdot (B\boldsymbol{h}'_{n-1}) = h'_{n-1,n-1}(\boldsymbol{x} \cdot \boldsymbol{b}_{n-1}), \tag{4.10}$$

which explicitly shows that indeed $\boldsymbol{b}_{n-1}$ is an integer relation for $\boldsymbol{x}$.

Two other essential topics in the discussion of any algorithm are termination and efficiency. We shall not discuss the former in this paper, as it is a bit beyond our purpose of applying PSLQ to Feynman integrals. For more information on the algorithm termination, see [Bor02, p. 163].

We now present a major result, without proof, concerning the efficiency of the PSLQ algorithm (the proof can be found in [Bor02, pp. 164-9]).

THEOREM 4.6. *Let $\mu = \min(M, T)$, where T is the bound passed on the PSLQ algorithm and M is the norm of a smallest integer relation for $\boldsymbol{x}$ (if $\boldsymbol{x}$ has no integer relations, then $M = \infty$). Then the PSLQ algorithm will terminate in fewer than*

$$\binom{n}{2} \frac{\log(\gamma^{n-1}\mu)}{\log \tau} \tag{4.11}$$

iterations, where $\tau > 1$ is defined by $\frac{1}{\tau^2} = \frac{1}{4} + \frac{1}{\gamma^2}$ and γ as in Step 1 of the algorithm. Upon termination, the algorithm will either return an integer relation for $\boldsymbol{x}$ of norm no larger than $\gamma^{n-2}M$ or return a lower bound $\geq T$ on the norm of any integer relation for $\boldsymbol{x}$.

The simple and unassuming term $\binom{n}{2} = n^2/2 - n/2$ in the theorem above is a powerful one as it means that the number of iterations will be given by a polynomial in n. This is one of the theorem's biggest advantages.

We now want to show how useful the algorithm can be for QFT calculations. Before we do this, however, let us present a toy example of the algorithm in practice.

4.3. **Worked Toy Example.** We begin with the vector $\boldsymbol{x} = (1, 0.5, 0.25)^t$ and we seek an integer relation. We know that there are many possibilities, such as $\boldsymbol{m} = (1, 2, -8)^t$. We begin with our vector, $\gamma = 2$, and the three matrices:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad H' = \begin{bmatrix} 0.488 & 0 \\ -0.781 & 0.447 \\ -0.390 & -0.894 \end{bmatrix}. \tag{4.12}$$

We have thus completed *Step 1* of the algorithm. Now we move on to size reduce H'. We begin with $i = 2, j = 1$. We get that $\lfloor h'_{2,1}/h'_{1,1} \rceil = -2$. We apply the row and column operations:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad H' = \begin{bmatrix} 0.488 & 0 \\ 0.195 & 0.447 \\ -0.390 & -0.894 \end{bmatrix}. \tag{4.13}$$

We now do the same for $i = 3, j = 2, \lfloor h'_{3,2}/h'_{2,2} \rceil = -2$:

(4.14)
$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 4 & 2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}, \quad H' = \begin{bmatrix} 0.488 & 0 \\ 0.195 & 0.447 \\ 0 & 0 \end{bmatrix}.$$

And finally for $i = 3, j = 1$ we see that $\lfloor h'_{3,1}/h'_{1,1} \rceil = 0$, so the row and column operations do nothing, so the matrices stay as above. We now move on to *Step 3*. We recognize that $\gamma^i |h'_{i,i}|$ is maximal for $r = 2 = n - 1$ and we also see that $h'_{3,2} = 0$, so we have that $\boldsymbol{b}_2 = (0, 1, -2)^t$ is an integer relation.

This toy example did not get to employ the unitary matrix Q to make H' lower trapezoidal, but that is a bit self-explanatory, and after that we only need to repeat a process similar to the one above.

We will now see how the algorithm is employed in practical QFT calculations.

5. Feynman Integrals and the PSLQ Theorem

Ever since the introduction of the PSLQ theorem, it has been widely used by physicists to find nice analytic expressions for constants in QFT, particularly in the realm of Feynman Integrals. There are various examples of PSLQ being used in QFT calculations and leading to mathematically interesting expressions.

5.1. **Why ζ-functions?** We shall see in the subsequent paragraphs that Feynman integrals and related objects in QFT often produce ζ-functions as well as related numbers such as multiple zeta values and polylogarithms. But why do we expect this?

In Eq. (2.3), we have the space-time dimension D. Usually dimensions are positive integers. However, Feynman integrals diverge for integer dimensions, so we must 'regularize' them by, for example, writing

$$D = 4 - \epsilon/2, \tag{5.1}$$

where $\epsilon \ll 1$. This method of dimensional regularization is, admittedly, a bit gut-wrenching, but it gives experimentally-attested values. I simply state that it is used. When we use dimensional regularization, we often times get terms of the form

$$\log \Gamma(1 - \epsilon), \tag{5.2}$$

which can be expanded as

$$\log \Gamma(1 - \epsilon) = \gamma x + \sum_{k=2}^{\infty} \frac{\zeta(k)}{k} x^k, \tag{5.3}$$

where γ is the Euler-Mascheroni constant. Thus, by calculating to higher orders of ϵ, we get higher order ζ-functions.

5.2. **6-Loop Periods.** The most basic framework studied in QFT is ϕ^4 theory: it studies an interaction term that varies with the fourth power of a background field ϕ. Although many real-life applications of QFT are not graspable within ϕ^4 theory, it is an extremely useful model. As shown in [AB21], Broadhurst has studied periods from ϕ^4 theory and found the following relations using the PSLQ algorithm:

$$P_{6,1} = 168\zeta(9), \quad P_{6,2} = \frac{1063}{9}\zeta(9) + 8\zeta(3)^3, \quad 16P_{6,3} + P_{6,4} = 1440\zeta(5)\zeta(3), \tag{5.4}$$

where the $P_{6,j}$ are 6-loop periods. For more information on what these periods are, please look into [Sch10]. It is enough for the purposes of this paper to know that these periods share a functional form closely resembling (2.3). The higher the loop number, the harder it is to compute since there are more independent momenta involved. It is also worth noting that in expressions such as (5.4), the powers of $\zeta(3)$, for example, are not determined by the algorithm. The numbers $\zeta(3), \zeta^3(3), \zeta(5), \zeta(9)$ must be independent inputs in $\boldsymbol{x}$. This of course implies a fair bit of guess work, but it's assisted by expressions like Eq. (5.3). Experimenting with PSLQ led Broadhurst to also wonder whether $P_{6,3}$ could be expressed independently in terms of multiple zeta values (MZV). The MZV $\zeta(j,k)$ is defined as

$$\zeta(j,k) = \sum_{m>n>0} \frac{1}{m^j n^k}. \tag{5.5}$$

He found that indeed,

$$P_{6,3} = \frac{36}{5}(12\zeta(5,3) - 29\zeta(8)) + 252\zeta(5)\zeta(3). \tag{5.6}$$

Notice how, in comparison to (5.4) above, this expression avoids $P_{6,4}$. Moreover, this led Broadhurst and Kreimer to develop the Broadhurst-Kreimer conjecture, which relates the number $N(w,d)$ of MZVs with weight w and depth d to a generating function as follows:

$$\prod_{w>2}\prod_{d>0}(1 - x^w y^d)^{N(w,d)} = 1 - \frac{x^3 y}{1-x^2} + \frac{x^{12}y^2(1-y^2)}{(1-x^4)(1-x^6)}. \tag{5.7}$$

It is important to point out that none of the expressions in this section have been proven in a mathematically rigorous manner, they are only highly probable. One assembles an $\boldsymbol{x} = (x_1, \ldots, , x_n)^t$ with likely candidates. If PSLQ returns an integer relation, then one can increase numerical precision in the input $\boldsymbol{x}$ and try again, but this is not rigorous proof for an expression such as Eq. (5.7).

5.3. **Boundary Values of Feynman Integrals.** Another example of QFT calculations being elucidated by PSLQ comes from the study of boundary values of Feynman integrals at kinetic points. (For more information see a reference textbooks such as [PS95].) In [Wei22, Ch. 15.4], Weinzierl reconstructs the quantity

$$G \approx 0.032893195194356041326359565602868932538 7 \tag{5.8}$$

in terms of

$$\begin{aligned}
\mathrm{Li}_4(1/2) &\approx 0.517479061673899386330758161898862945618,\\
\zeta(4) &\approx 1.082323233711138191516003696541167902776,\\
\zeta(3)\ln(2) &\approx 0.833202353297691993445762529661560103894,\\
\zeta(2)\ln^2(2) &\approx 0.790313530113954608772917335680644104204,\\
\ln^4(2) &\approx 0.230835098583083451887497717767812771517.
\end{aligned} \tag{5.9}$$

He supplied the vector $\boldsymbol{x} = (G, \mathrm{Li}_4(1/2), \zeta(4), \zeta(3)\ln(2), \zeta(2)\ln^2(2), \ln^4(2))$ to PSLQ and got $\boldsymbol{m} = (8, -24, 24, -22, 6, -1)$. This then gives the relation

$$G = 3\mathrm{Li}_4(1/2) - 3\zeta(4) + \frac{11}{4}\zeta(3)\ln(2) - \frac{3}{4}\zeta(2)\ln^2(2) + \frac{1}{8}\ln^4(2), \tag{5.10}$$

where $\mathrm{Li}_s(z)$ is defined as the polylogarithm

$$\mathrm{Li}_s(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^s}. \tag{5.11}$$

Here, the values of $\boldsymbol{x}$ are chosen as follows. First, G is simply the value to be reconstructed, so it has been computed to high precision numerically. The other values are educated guesses. For example, we know from [Kre00, Ch. 7] that Feynman graphs with a $(2, q)$ torus produce term with $\zeta(q)$, so by knowing the structure of the graph, a researcher like Weinzierl can intelligently guess which sets of numbers might construct the value G.

Thus far we have seen how the PSLQ algorithm takes the central question of Diophantine approximation and elevates it to higher dimensions by way of basis construction and matrix reduction. This algorithm, chosen as one of the "Top Ten Algorithms of the Century" in 2000 by Jack Dongarra and Francis Sullivan, can then be applied to numerically reconstruct physical constants from QFT calculations, thus providing an unlikely bridge between Physics and Diophantine Approximation.

References

[AB21] Kevin Acres and David Broadhurst. Empirical Determinations of Feynman Integrals Using Integer Relation Algorithms. In *Anti-Differentiation and the Calculation of Feynman Amplitudes*, pages 63–82. Springer, Cham, July 2021.

[Bor02] Peter Borwein. Appendix B: Lattice Basis Reduction and Integer Relations. In *Computational Excursions in Analysis and Number Theory*, CMS Books in Mathematics. Springer New York, New York, 1 edition, 2002.

[FB92] Helaman R. P. Ferguson and David H. Bailey. A Polynomial Time, Numerically Stable Integer Relation Algorithm. RNR Technical Report RNR-91-032, Supercomputing Research Center, 17100 Science Drive, Bowie, MD 20715, July 1992.

[FBA99] Helaman Ferguson, David Bailey, and Steve Arno. Analysis of PSLQ, an integer relation finding algorithm. *Mathematics of Computation*, 68(225):351–369, 1999.

[Kre00] D. Kreimer. *Knots and Feynman Diagrams*. Cambridge University Press, 2000.

[PS95] Michael Peskin and Daniel Schroeder. *An Introduction to Quantum Field Theory*. Perseus Books, Reading, Massachusetts, 1995.

[Sch10] Oliver Schnetz. Quantum periods: a census of ϕ^4-transcendentals. *Commun. Number Theory Phys.*, 4, 2010.

[Wei22] Stefan Weinzierl. *Feynman Integrals*. UNITEXT for Physics. Springer Cham, 1 edition, 2022.

Massachusetts Institute of Technology
Email address: dr_orona@mit.edu