Automated generation of lattice QCD Feynman rules

A. Hart[a], G.M. von Hippel[b], R.R. Horgan[c], E.H. Müller[a]

[a] *SUPA, School of Physics and Astronomy, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, U.K.*
[b] *Deutsches Elektronen-Synchrotron DESY, Platanenallee 6, 15738 Zeuthen, Germany*
[c] *DAMTP, CMS, University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, U.K.*

Abstract

The derivation of the Feynman rules for lattice perturbation theory from actions and operators is complicated, especially for highly improved actions such as HISQ. This task is, however, both important and particularly suitable for automation. We describe a suite of software to generate and evaluate Feynman rules for a wide range of lattice field theories with gluons and (relativistic and/or heavy) quarks. Our programs are capable of dealing with actions as complicated as (m)NRQCD and HISQ. Automated differentiation methods are used to calculate also the derivatives of Feynman diagrams.

Key words: Quantum Chromodynamics, QCD, lattice QCD, perturbation theory
PACS: 11.15.Ha, 12.38.Gc
2000 MSC: 81-04, 81T13, 81T15, 81T18, 81T25, 81V05, 65S05, 41A58

PROGRAM SUMMARY

Manuscript Title: Automated generation of lattice QCD Feynman rules
Authors: A. Hart, G.M. von Hippel, R.R. Horgan, E.H. Müller.
Program Title: HiPPy, HPsrc
Journal Reference:
Catalogue identifier:
Licensing provisions: GPLv2 (see note in Sec. 1.1.)
Programming languages: Python, Fortran95
RAM: Problem specific, typically less than 1GB for either code.
Keywords: Quantum Chromodynamics, QCD, lattice QCD, perturbation theory
PACS: 11.15.Ha; 12.38.Gc
Classification: 4.4 Feynman diagrams; 11.5 Quantum Chromodynamics, Lattice Gauge Theory
Nature of problem:
Derivation and use of perturbative Feynman rules for complicated lattice QCD actions.
Solution method:
An automated expansion method implemented in Python (HiPPy) and code to use expansions to generate Feynman rules in Fortran95 (HPsrc).

Restrictions:
No general restrictions. Specific restrictions are discussed in the text.
Running time:
Very problem specific, depending on the complexity of the Feynman rules and the number of integration points. Typically between a few minutes and several weeks.

LONG WRITE-UP

1. Introduction

Non–abelian gauge theories are the most important ingredient in our present understanding of elementary particles and their interactions. In particular, quantum chromodynamics (QCD) is now universally believed to be the correct theory of the strong interactions. However, while perturbation theory has been used successfully in describing the scattering of particles by partons, the perturbative series does not converge at hadronic energy scales. Moreover, the phenomena of confinement and the hadronic spectrum are fundamentally beyond the reach of perturbation theory. Therefore, non-perturbative Monte Carlo simulations of lattice-regularised QCD are crucial in order to obtain a full description and understanding of QCD phenomena.

The lattice regularisation with a lattice spacing a does, however, introduce a sharp momentum cutoff at the momentum scale π/a. Connecting lattice measurements to their continuum counterparts therefore requires renormalisation factors accounting for the excluded high-frequency modes. In particular, renormalisation is needed for QCD matrix elements, and for fixing the bare quark masses to be used in the lattice Lagrangian. Renormalisation is also necessary to determine the running of the strong coupling constant α_s and to relate the lattice regularisation scale $\Lambda_{\rm lat}$ to $\Lambda_{\rm QCD}$. Since the lattice regularisation also introduces discretisation errors, renormalisations are also used to "improve" the lattice action in order to reduce these discretisation errors at a given lattice spacing.

While in some cases the renormalisation constants can be determined non-perturbatively [1, 2], results at finite lattice spacing can depend upon the method used (cf. e.g. [3]), and non-perturbative methods do not cope well with operators that mix under renormalisation. For these reasons, lattice perturbation theory plays an important role in determining the renormalisation constants needed to extract continuum predictions from lattice QCD.

Given the breakdown of perturbation theory in low energy QCD, one might doubt whether it could work on the lattice. An argument in favour of its use is given in [4]: since the renormalisation factors may be thought of as compensating for the ultraviolet modes excluded by the lattice regulator, and since for typical lattice spacings $a \lesssim 0.1$ fm, the excluded modes have momenta in excess of 5 GeV, the running QCD coupling $\alpha_s(\pi/a)$ is small enough that perturbation theory should rapidly converge. The wide range of results reviewed in [5, 6] show that perturbation theory is useful for a large range of lattice QCD processes. The assumption that non-perturbative effects do not contribute on these short length scales, can be tested directly in some cases by comparing higher order perturbative calculations with Monte Carlo simulations performed at a range of weak couplings [7, 8, 9, 10, 11, 12]), or by using so-called stochastic techniques [13]. In all these cases, the non-perturbative contributions turned out to be small. Other comparisons, such as [3], cannot distinguish non-perturbative effects from higher-order perturbative corrections.

Lattice perturbation theory therefore provides a reliable, and the only systematically improvable, method for determining the full range of renormalisation constants [5].

As in the continuum, the calculation of lattice Feynman diagrams is a two–stage process. First, the lattice action and any operator insertions are Taylor–expanded in the (bare) coupling constant to give the propagators and vertices that form the Feynman rules (the "vertex expansion" stage). Secondly, these rules are then used to construct and numerically evaluate Feynman diagrams, possibly after some algebraic simplification (the "Feynman diagram evaluation" stage).

The latter task is more complicated than in the continuum due to the presence of Lorentz symmetry violating terms at finite lattice spacing, and on a finite lattice volume also by the more complicated nature of discrete momentum sums as compared to momentum integrals. Diagrams are therefore usually evaluated using numerical routines like VEGAS [14, 15], or proprietary mathematical packages, possibly after manipulation using other computer algebra packages like FORM [16].

The greater difficulty is posed, however, by the task of vertex expansion. Deriving the Feynman results on the lattice is far more complicated than in the continuum for a number of reasons. Firstly, lattice gauge fields are elements of the gauge Lie group itself rather than of its Lie algebra. To obtain the perturbative expansion of the action, we must therefore expand exponentials of non–commuting objects. As a consequence, the Feynman rules even for the simplest lattice action are already much more complicated than their continuum counterparts.

Secondly, modern lattice theories generally contain a large number of additional (renormalisation group irrelevant) terms chosen to improve specific aspects of the Monte Carlo simulation, such as the rate of approach to the continuum or chiral limits of QCD. Since there is no unique prescription for these terms, and the best choice depends on which quantities we are most interested in simulating, a large number of different lattice actions and operators are in use. Subtle though the differences between the lattice formulations may be, each choice provides a completely separate regularisation of QCD with its own set of renormalisation constants and in particular its own lattice Feynman rules. For a long time, the complications of the perturbative expansion have led to the calculations of renormalisation factors lagging far behind new developments in the improvement of lattice theories. In many cases this has restricted the physical predictions that could be obtained from the simulations.

An automated method for deriving lattice Feynman rules for as wide a range of different theories as possible is therefore highly desirable. The vertex expansion should be fast enough not to impose undue constraints on the choice of action. To avoid human error, the user should be able to specify the action in a compact and intuitive manner. Since the evaluation of the Feynman diagrams can be computationally intensive, and will often need to be carried out in parallel on costly supercomputing facilities, parsimony dictates that the Feynman rules should be calculated in advance on a different system, and rendered as ma-

chine readable files that can be copied to the supercomputer for the Feynman diagram evaluation stage.

In this paper we describe a pair of software packages for deriving the Feynman rules for arbitrary lattice actions[1] and for evaluating the resulting vertices in a numerical Feynman diagram calculation. Our algorithm is based through our older algorithm [17] on the seminal work of Lüscher and Weisz [18]. A different implementation of the latter has been used in [19, 20, 21], and a similar method is employed in [22].

The new feature of the algorithm presented here is that it is capable of expanding not only gluonic actions like the algorithm of [18], and fermionic actions like our algorithm from [17], but also far more complicated multiply-smeared fermionic actions with reunitarisation such as HISQ [23], and that it supports taking advantage of the factorisation inherent in some lattice actions, such as improved lattice formulations of NRQCD [24].

As in [18] and [17], the vertex expansion is performed completely independently of any boundary conditions, allowing for instance, the use of twisted periodic boundary conditions as a gauge–invariant infrared regulator [25, 26] or for changing the discrete momentum spectrum in other ways [27].

We have used the software packages described for calculations of the renormalised anisotropy in gauge theories [28, 29], to study the mean link in Landau gauge for tadpole improvement [11], to measure the electromagnetic decays of heavy quark systems using NRQCD [30, 31, 32, 33], to calculate the radiative corrections to the gluonic action due to Highly Improved Staggered (HISQ) sea quarks [34, 35] and the renormalisation of the self energy of heavy quarks using moving NRQCD (mNRQCD) [36].

The code is flexible and can be easily extended, as has already been done for lattice–regularised chiral perturbation theory [37], perturbative calculations in the Schrödinger functional [38, 39] and anisotropy calculations [40].

The structure of this paper is as follows. In the next Section, we review the basic expansion algorithm described in Ref. [17], outlining some improvements in, for instance, the handling of automatic derivatives and spin matrices. In Sec. 3, we present novel extensions to the algorithm that are needed to describe complicated fermion actions, including HISQ, NRQCD and mNRQCD.

Sec. 4 provides details of the implementation of the algorithm in the HiPPy and HPsrc codes and of their installation, testing and use. We make some concluding remarks in Sec. 5. Technical details are relegated to the appendices.

1.1. Licence

The HiPPy and HPsrc codes are released under the second version of the GNU General Public Licence (GPL v2). Therefore anyone is free to use or modify the code for their own calculations.

[1] We shall use the term "action" so as to include measurement operators here and in the following.

As part of the licensing, we ask that any publications including results from the use of this code or of modifications of it cite Refs. [18, 17] as well as this paper.

Finally, we also ask that details of these publications, as well as of any bugs or required or useful improvements of this core code, would be communicated to us.

2. Theoretical background

In this section we describe the algorithms used to give the most efficient, yet generic, implementation of the Feynman rules for lattice actions. These extend the original work of Lüscher and Weisz [18] and developments described in Ref. [17].

2.1. Fields on the lattice

We consider a D-dimensional hypercubic spacetime lattice with lattice spacing a and extent $L_\mu a$ in the μ-direction:

$$\Lambda = \left\{ (x_1, \ldots, x_D) \in \mathbb{R}^D \;\middle|\; \forall\, \mu \in \{1, \ldots, D\} :\; \frac{x_\mu}{a} \in \{0, \ldots, L_\mu - 1\} \right\} \tag{1}$$

where lattice sites are labelled by a vector $\boldsymbol{x} \in \Lambda$. In the following, we will usually set $a = 1$ (a lattice anisotropy can be introduced by rescaling the coupling constants in the action [28]). Let $\boldsymbol{e}_\mu$ be a right-handed orthonormal basis, and $\boldsymbol{e}_{-\mu} \equiv -\boldsymbol{e}_\mu$.

A lattice path consisting of l links starting at site $\boldsymbol{x}$ can be specified by an ordered set of directions given by integers, $s_i \in \{-D, \ldots, -1, 1, \ldots, D\}$:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{s}) \equiv \{\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{s} = [s_0, s_1, \ldots, s_{l-1}]\} \,, \tag{2}$$

with the $j^{\,\text{th}}$ point on the path being

$$\boldsymbol{z}_j = \begin{cases} \boldsymbol{x} \,, & j = 0 \,, \\ \boldsymbol{z}_{j-1} + a\boldsymbol{e}_{s_{j-1}} \,, & 0 < j \leq l \,, \end{cases} \tag{3}$$

and $\boldsymbol{y} \equiv \boldsymbol{z}_l$.

For periodic boundary conditions, the momentum vectors are

$$\boldsymbol{k} = \frac{2\pi}{a} \left(\frac{\bar{k}_1}{L_1}, \ldots, \frac{\bar{k}_D}{L_D} \right) \,, \quad 0 \leq \bar{k}_\mu < L_\mu \,, \quad \bar{k}_\mu \in \mathbb{Z} \,, \tag{4}$$

and the Fourier expansion of a function ϕ is

$$\tilde{\phi}(\boldsymbol{k}) = \sum_{\boldsymbol{x}} e^{-i\boldsymbol{k}\cdot\boldsymbol{x}} \phi(\boldsymbol{x}) \,, \qquad \phi(\boldsymbol{x}) = \frac{1}{V} \sum_{\boldsymbol{k}} e^{i\boldsymbol{k}\cdot\boldsymbol{x}} \tilde{\phi}(\boldsymbol{k}) \,, \tag{5}$$

where $V = \prod_\mu L_\mu$ is the lattice volume.

Twisted boundary conditions [41] provide a useful gauge-invariant infrared regulator in perturbative calculations [18]. These change the momentum spectrum, converting colour factors into "twist matrices" associated with momenta interstitial to the reciprocal lattice (with the introduction of an additional quantum number, "smell", for fermions [42, 43, 11]). The HPSRC code fully supports such boundary conditions but for simplicity we only discuss periodic boundary conditions in this paper.

Following Ref. [18], we denote the gauge field associated with a link as $U_{\mu>0}(\boldsymbol{x}) \in SU(N)$, and define $U_{-\mu}(\boldsymbol{x}) = U_\mu^\dagger(\boldsymbol{x} - a\boldsymbol{e}_\mu)$. The gauge potential $A_\mu \in \mathrm{alg}(SU(N))$ associated with the midpoint of the link is defined through

$$U_{\mu>0}(\boldsymbol{x}) = \exp\left(agA_\mu\left(\boldsymbol{x} + \frac{a}{2}\boldsymbol{e}_\mu\right)\right) = \sum_{r=0}^{\infty} \frac{\left(agA_\mu(\boldsymbol{x} + \frac{a}{2}\boldsymbol{e}_\mu)\right)^r}{r!} \tag{6}$$

where g is the bare coupling constant. In terms of the anti-Hermitian generators of $SU(N)$,

$$A_\mu = A_\mu^a\, T_a, \qquad [T_a, T_b] = -f_{abc}T_c, \qquad \mathrm{Tr}\,(T_aT_b) = -\frac{1}{2}\,\delta_{ab}\ . \tag{7}$$

Quark fields $\psi(\boldsymbol{x})$ transform according to the representation chosen for the generators T_a, which we take to be the fundamental representation (other choices will affect the colour factors, but not the structure of our algorithm).

2.2. Perturbative expansion of Wilson lines

The Wilson line $L(\boldsymbol{x}, \boldsymbol{y}, U)$ associated with the lattice path $\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{s})$ is a product of links

$$L(\boldsymbol{x}, \boldsymbol{y}, U) = \prod_{i=0}^{l-1} U_{s_i}(\boldsymbol{z}_i) = \prod_{i=0}^{l-1} \exp\left[\mathrm{sgn}(s_i)agA_{|s_i|}\left(\boldsymbol{z}_i + \frac{a}{2}\boldsymbol{e}_{s_i}\right)\right]\ . \tag{8}$$

As all actions and operators can be written as sums of Wilson lines (possibly terminated by fermion fields), our goal is to efficiently expand L in terms of the gauge potential in momentum space:

$$L(\boldsymbol{x}, \boldsymbol{y}; A) = \sum_r \frac{(ag)^r}{r!} \sum_{\boldsymbol{k}_1,\mu_1,a_1} \cdots \sum_{\boldsymbol{k}_r,\mu_r,a_r} \tilde{A}_{\mu_1}^{a_1}(\boldsymbol{k}_1) \ldots \tilde{A}_{\mu_r}^{a_r}(\boldsymbol{k}_r) \times V_r(\boldsymbol{k}_1, \mu_1, a_1; \ldots; \boldsymbol{k}_r, \mu_r, a_r)\ . \tag{9}$$

The vertex functions V_r factorise as

$$V_r(\boldsymbol{k}_1, \mu_1, a_1; \ldots; \boldsymbol{k}_r, \mu_r, a_r) = C_r(a_1, \ldots, a_r)\ Y_r^{\mathcal{L}}(\boldsymbol{k}_1, \mu_1; \ldots; \boldsymbol{k}_r, \mu_r) \tag{10}$$

with a momentum- and path-independent Clebsch–Gordan (colour) factor C_r

$$C_r(a_1, \ldots, a_r) = \prod_{i=1}^{r} T_{a_i}\ . \tag{11}$$

It is therefore more efficient to calculate just the expansion of the reduced vertex functions, $Y_r^{\mathcal{L}}$ (with an appropriate description of the colour trace structure where ambiguous — see Appendix B of Ref. [17] for further details). The reduced vertex function can be written as a sum of terms, each of which contains an exponential. For convenience, we will call each term a "monomial":

$$Y_r^{\mathcal{L}}(\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r) = \sum_{n=1}^{n_r(\{\mu\})} f_n^{(r,\{\mu\})} \exp\left(\frac{i}{2}\sum_{j=1}^{r}\boldsymbol{k}_j\cdot\boldsymbol{v}_{n,j}^{(r,\{\mu\})}\right) , \qquad (12)$$

where for each combination of r Lorentz indices we have n_r terms, each with an amplitude f and locations $\boldsymbol{v}$ of the r factors of the gauge potential, which are drawn from the locations of the midpoints of the links in the path $\mathcal{L}$. To avoid floating point ambiguities, we express the components of all position D-vectors as integer multiples of $\frac{a}{2}$ (accounting for the factor of $\frac{1}{2}$ in the exponent).

In the HPSRC code, we use the convention that all momenta flow into the vertex, so $\sum_{i=1}^{r}\boldsymbol{k}_i = 0$.

Eqn. (12) makes clear that the number of monomials depends not just on the number of gluons r, but also on the choice of Lorentz indices $\{\mu\}$, and that each monomial has a different amplitude and set of r positions. For clarity of presentation, we will, however, suppress these additional arguments in later expressions (notably Eqns. (17, 21, 26, 44, 45)).

2.2.1. Implementation notes

The generation of the Feynman rules for generic momenta thus reduces to a calculation of the amplitudes f and locations $\boldsymbol{v}$ of the monomials that build up the various reduced vertices Y_r.

This is all carried out in the HIPPY code, a description of which can be found in Sec. 4 of Ref. [17]. The amplitudes and locations defining each monomial are encoded as an instance of the class `Entity`, and the collection of monomials that make up the reduced vertices is encoded as an instance of class `Field`. The data structures have been chosen to ensure that equivalent monomials are combined to minimise the size of the reduced vertex description.

Once expanded, the monomials required for the reduced vertices at each order are written to disk as a text file.

The HPSRC code reads these (previously generated) vertex files at runtime. For given momenta $\{\boldsymbol{k}\}$, lorentz indices $\{\mu\}$ and colour indices, the Y_r are constructed as given in Eqn. (12), which is then multiplied by the appropriate Clebsch-Gordan colour factor(s) to form the (Euclidean) Feynman rule, $-V_r$.

2.3. Realistic actions: the fermion sector

Realistic lattice fermion and gauge actions require some refinements to this generic description. We begin with the fermion sector. The most general gauge- and translation-invariant action can be written as

$$S_F(\psi,U) = \sum_{\boldsymbol{x}}\sum_{\mathcal{W}} h_{\mathcal{W}}\,\bar{\psi}(\boldsymbol{x})\,\Gamma_{\mathcal{W}}\,W(\boldsymbol{x},\boldsymbol{y},U)\,\psi(\boldsymbol{y}) \qquad (13)$$

and consists of Wilson lines W defined by open paths $\mathcal{W}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{s})$, each carrying an associated coupling constant $h_{\mathcal{W}}$ and a spin matrix $\Gamma_{\mathcal{W}}$ (possibly the identity).

Using the convention that all momenta flow into the vertex, the perturbative expansion is

$$S_F(\psi, A) = \sum_r \frac{g^r}{r!} \sum_{\boldsymbol{k}_1,\mu_1,a_1} \dots \sum_{\boldsymbol{k}_r,\mu_r,a_r} \tilde{A}^{a_1}_{\mu_1}(\boldsymbol{k}_1) \dots \tilde{A}^{a_r}_{\mu_r}(\boldsymbol{k}_r) \times \sum_{\boldsymbol{p},\boldsymbol{q},b,c} \tilde{\bar{\psi}}^b(\boldsymbol{p})\, V_{F,r}(\boldsymbol{p}, b;\, \boldsymbol{q}, c;\, \boldsymbol{k}_1, \mu_1, a_1;\, \dots;\, \boldsymbol{k}_r, \mu_r, a_r)\, \tilde{\psi}^c(\boldsymbol{q}) \,. \quad (14)$$

The Euclidean Feynman rule for the r-point gluon–fermion–anti-fermion vertex is $-g^r V_{F,r}$, where the symmetrised vertex is:

$$V_{F,r}(\boldsymbol{p}, b; \boldsymbol{q}, c; \boldsymbol{k}_1, \mu_1, a_1;\, \dots; \boldsymbol{k}_r, \mu_r, a_r) = \frac{1}{r!} \sum_{\sigma \in \mathcal{S}_r} \sigma \cdot C_{F,r}(b, c; a_1, \dots, a_r)\; \sigma \cdot Y_{F,r}(\boldsymbol{p}, \boldsymbol{q}; \boldsymbol{k}_1, \mu_1;\, \dots; \boldsymbol{k}_r, \mu_r) \,, \quad (15)$$

where $\mathcal{S}_r$ is the permutation group of r objects and $\sigma \in \mathcal{S}_r$ is applied to the gluonic variables, $\{\boldsymbol{k}\}$, $\{\mu\}$ and $\{a\}$. The normalisation factor of $r!$ for this is additional to the $r!$ factor arising from the Taylor expansion of the exponential in Eqn. (14). The reduced vertex $Y_{F,r} = \sum_{\mathcal{W}} h_{\mathcal{W}} Y^{\mathcal{W}}_{F,r}$ is the sum of contributions from paths $\mathcal{W}$.

In most cases the Clebsch-Gordan colour factor is the matrix element:

$$C_{F,r}(b, c; a_1, \dots, a_r) = (T_{a_1} \dots T_{a_r})_{bc} \,, \quad (16)$$

and the reduced vertex function has the structure:

$$Y_{F,r}(\boldsymbol{p}, \boldsymbol{q}; \boldsymbol{k}_1, \mu_1; \dots; \boldsymbol{k}_r, \mu_r) = \sum_{n=1}^{n_r(\{\mu\})} \Gamma_n\, f_n \times \exp\left(\frac{i}{2} \left(\boldsymbol{p} \cdot \boldsymbol{x} + \boldsymbol{q} \cdot \boldsymbol{y} + \sum_{j=1}^{r} \boldsymbol{k}_j \cdot \boldsymbol{v}_{n,j} \right) \right) \,, \quad (17)$$

where we understand $\Gamma_n \equiv \Gamma_{r,\{\mu\},n}$. Cases with more complicated colour structures do arise, for example the use of traceless field strengths in QCD. Such structures are accommodated in the codes; monomials with different colour structures are distinguished using "pattern lists" (discussed in Appendix B of Ref. [17]) and appropriate colour factors are applied to each when the Feynman rule is constructed.

As there are no permutation symmetries in $C_{F,r}$, there is no advantage to carrying out any symmetrisation in the HiPPy expansion code. In the HPsrc code, symmetrisation of the Feynman rule shown in Eqn. (15) carries a potentially significant computational overhead: the reduced vertices must be calculated afresh for each permutation. Not all such permutations may be needed

because symmetries of a Feynman diagram can reduce the number of distinct contributions to its value from the terms in Eqn. (15). For this reason, symmetrisation is not carried automatically in the HPSRC code and the user must therefore explicitly construct all permutations requiring a different calculation from Eqn. (15), applying the appropriate normalisation factor.

2.4. Realistic actions: the gluon sector

A typical gluonic action is

$$S(\psi, U) = \sum_{\boldsymbol{x}} \sum_{\mathcal{P}} c_{\mathcal{P}} \,\mathrm{Re}\,\mathrm{Tr}\,[P(\boldsymbol{x}, \boldsymbol{x}, U)] \ , \tag{18}$$

built of Wilson loops P defined by closed paths $\mathcal{P}(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{s})$, each with coupling constant $c_{\mathcal{P}}$. The perturbative action is

$$S_G(A) = \sum_r \frac{g^r}{r!} \sum_{\boldsymbol{k}_1, \mu_1, a_1} \ldots \sum_{\boldsymbol{k}_r, \mu_r, a_r} \tilde{A}^{a_1}_{\mu_1}(\boldsymbol{k}_1) \ldots \tilde{A}^{a_r}_{\mu_r}(\boldsymbol{k}_r) \times V_{G,r}(\boldsymbol{k}_1, \mu_1, a_1; \ldots; \boldsymbol{k}_r, \mu_r, a_r) \ . \tag{19}$$

The Euclidean Feynman rule for the r-point gluon vertex function is $(-g^r V_{G,r})$, and the vertex $V_{G,r}$ is [18]

$$V_{G,r}(\boldsymbol{k}_1, \mu_1, a_1; \ldots; \boldsymbol{k}_r, \mu_r, a_r) = \frac{1}{r!} \sum_{\sigma \in \mathcal{S}_r} \sigma \cdot C_{G,r}(a_1, \ldots, a_r) \ \sigma \cdot Y_{G,r}(\boldsymbol{k}_1, \mu_1; \ldots; \boldsymbol{k}_r, \mu_r) \ , \tag{20}$$

The reduced vertex $Y_{G,r} = \sum_{\mathcal{P}} c_{\mathcal{P}} Y^{\mathcal{P}}_{G,r}$ is the sum of contributions from paths $\mathcal{P}$. As before, the $(r!)$ factor normalises the symmetrisation. $Y^{\mathcal{P}}_{G,r}$ can be expanded as

$$Y^{\mathcal{P}}_{G,r}(\boldsymbol{k}_1, \mu_1; \ldots; \boldsymbol{k}_r, \mu_r) = \sum_{n=1}^{n_r} f_n \ \exp\left(\frac{i}{2} \sum_i \boldsymbol{k}_i \cdot \boldsymbol{v}_{n,i} \right) \ . \tag{21}$$

In most cases we expect the lattice action to be real. Thus, for every monomial $(f_n; \{\boldsymbol{v}_{n,i}\})$ in Eqn. (21), there must be a corresponding term $((-1)^r f_n^*; \{-\boldsymbol{v}_{n,i}\})$. We can therefore speed up the evaluation of the Feynman rules by removing the latter term, and replacing the exponentiation in Eq. (21) with "cos" for r even, and with "$i\sin$" for r odd. This can either be done by recognising conjugate contours in the action (e.g. $S = \frac{1}{2}\,\mathrm{Tr}[P + P^\dagger]$) and expanding only one, or by attaching a flag to each monomial to signal whether its complex conjugate has already been removed.

If in addition the action has the form Eq. (18) with a single trace in the fundamental representation, the colour factors are

$$C_{G,r}(a_1, \ldots, a_r) = \frac{1}{2} \left[\mathrm{Tr}\ (T_{a_1} \ldots T_{a_r}) + (-1)^r \,\mathrm{Tr}\ (T_{a_r} \ldots T_{a_1}) \right] \ . \tag{22}$$

which has a number of permutation symmetries:

$$\sigma \cdot C_{G,r} = \chi_r(\sigma)\, C_{G,r}\ , \text{ where } \chi_r(\sigma) = \begin{cases} 1 & \text{for } \sigma \text{ a cyclic permutation,} \\ (-1)^r & \text{for } \sigma \text{ the inversion.} \end{cases} \tag{23}$$

There is thus a great advantage in carrying out some of the symmetrisation in Eqn. (20) at the expansion stage in the HIPPY code. Many of the extra monomials generated by symmetrising over subgroup $\mathcal{Z}_r$ (generated by cyclic permutations and inversion) are equivalent and can be combined in the HIPPY code, significantly reducing the number of exponentiation operations required to construct the partially-symmetrised $Y'_{G,r}$:

$$V_{G,r}(\boldsymbol{k}_1, \mu_1, a_1;\ ...;\ \boldsymbol{k}_r, \mu_r, a_r) = \sum_{\sigma \in \mathcal{S}_r/\mathcal{Z}_r} \sigma \cdot C_{G,r}(a_1, ..., a_r) \times \sigma \cdot Y'_{G,r}(\boldsymbol{k}_1, \mu_1;\ ...;\ \boldsymbol{k}_r, \mu_r)\ , \tag{24}$$

$$Y'_{G,r} = \sum_{\substack{\mathcal{P} \\ \sigma \in \mathcal{Z}_r}} c_{\mathcal{P}} \chi_r(\sigma)\, \sigma \cdot Y^{\mathcal{P}}_{G,r}\ . \tag{25}$$

The $\chi_r(\sigma)$ factors go into the amplitudes of the new monomials coming from the partial symmetrisation.

The number of symmetrisation steps remaining to be carried out in the HPSRC code is the number of cosets in $\mathcal{S}_r/\mathcal{Z}_r$ (one for $r \leq 3$, three for $r = 4$, twelve for $r = 5$ etc.). These symmetrisation steps (and the normalisation) are carried out automatically in the HPSRC gluon vertex modules.

2.5. Diagram differentiation

In many cases, such as when computing wavefunction renormalisation constants, one needs to calculate the derivative of a Feynman diagram with respect to one or more momenta. Whilst derivatives can be computed numerically using an appropriate local difference operator, such differencing schemes are frequently numerically unstable and require computing the Feynman diagram multiple times. Automatic differentiation methods [44] are a stable and cost-saving alternative.

We can easily construct the differentiated Feynman vertex using Eqn. (12). If we want to differentiate with repsect to momentum component q_ν, we first construct a rank r object $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_r]$ which represents the proportion of momentum $\boldsymbol{q}$ in each leg of the Feynman diagram. Momentum conservation dictates $\sum_i \tau_i = 0$. For instance, for a gluon 3-point function with incoming momenta $(\boldsymbol{p}, -\boldsymbol{p} + 2\boldsymbol{q}, -2\boldsymbol{q})$, we would have $\boldsymbol{\tau} = [0, 2, -2]$. The differentiated vertex is

$$\frac{d}{dq_\nu} Y^{\mathcal{L}}_r(\boldsymbol{k}_1, \mu_1; \ldots; \boldsymbol{k}_r, \mu_r) = \sum_{n=1}^{n_r} \frac{i f_n}{2} \left(\sum_{j=1}^{r} \tau_j v_{n,j;\nu} \right) \exp\left(\frac{i}{2} \sum_{j=1}^{r} \boldsymbol{k}_j \cdot \boldsymbol{v}_{n,j} \right) \tag{26}$$

and so on for higher derivatives. We may therefore simultaneously calculate as many differentials as we need for the cost of just one exponentiation. If this momentum expansion is placed into an appropriate data structure for which appropriately overloaded operations have been defined, it is straightforward to create the Taylor series for a Feynman diagram by simply multiplying the vertex factors together. We use a slightly modified version of the TaylUR package [45, 46] to do this, which encodes the Taylor series expansion in the HPSRC Fortran code as a derived type `taylor`, for which all arithmetic operations and elementary functions have been overloaded so as to respect Leibniz's and Faà di Bruno's rules for higher derivatives of products and functions.

In calculations that require only certain higher-order derivatives, the taylor multiplication can be significantly sped up by defining a mask that only propagates certain terms in the Taylor expansion. This has not, however, been implemented in the distributed version of the code.

2.6. Spin algebra

The Feynman rules for fermions contain spin matrices (which may be Pauli matrices, e.g. for NRQCD, or Dirac matrices for relativistic fermions). We can expand a generic spin matrix W using a basis $\{\Gamma_i\}$:

$$W = \sum_{i \in I(W)} w_i \Gamma_i \tag{27}$$

The product of any two spin basis matrices Γ_i and Γ_j is another basis matrix (with label n_{ij}) times a phase:

$$\Gamma_i \Gamma_j = \phi_{ij} \Gamma_{n_{ij}} \ , \tag{28}$$

We choose the basis so that these phases ϕ_{ij} are real. Where another convention is desired, the amplitudes w_i need to be adjusted appropriately.

For Pauli matrices, we use a basis $\{\Gamma_i\} = \{1, i\sigma_k\}$ and $I(W) \subseteq \{0, \ldots, 3\}$, giving:

$$i\sigma_j . i\sigma_k = \begin{cases} 1 & j = k, \\ \epsilon_{jkm}\, i\sigma_m & j \neq k. \end{cases} \tag{29}$$

For the (Euclidean) Dirac matrices, we choose $\{\Gamma_i\} = \{1, \gamma_\mu, \sigma_{\mu\nu}, \gamma_5\gamma_\mu, \gamma_5\}$ and $I(W) \subseteq \{0, \ldots, 15\}$. We define $\gamma_5 \equiv \gamma_1\gamma_2\gamma_3\gamma_4$ and note the definition here $\sigma_{\mu\nu} \equiv \frac{1}{2}[\gamma_\mu, \gamma_\nu]$. The multiplication for the basic γ matrices is thus

$$\gamma_\mu \gamma_\nu = \begin{cases} 1 & \mu = \nu, \\ \sigma_{\mu\nu} & \mu \neq \nu. \end{cases} \tag{30}$$

We can thus write the product of two general spin matrices as

$$\begin{aligned} WW' &= \left(\sum_{j \in I(W)} w_j \Gamma_j \right) \left(\sum_{k \in I(W')} w'_k \Gamma_k \right) \\ &= \sum_{i \in I(WW')} \left(\sum_{\substack{j \in I(W), k \in I(W') \\ n_{jk} = i}} w_j w'_k \phi_{jk} \right) \Gamma_i \end{aligned} \tag{31}$$

where $I(WW') = \{ n_{ij} \mid i \in I(W),\ j \in I(W') \}$.

We use this implicit representation of the spin matrices. Matrix operations on explicit Dirac matrices take $\mathcal{O}(4^3)$ operations and introduce additional rounding errors. Eqn. (31), by contrast, needs $|I(W)| \times |I(W')|$ multiplications, depending on how many basis matrices are needed to describe W and W'. If $|I(W)|, |I(W')| < 8$, the latter method is more efficient and this is almost always the case.

There are greater gains when inverting spin matrices of the form

$$S^{-1} = a_0 1 + \sum_{\mu=1}^{4} a_\mu \gamma_\mu \ , \tag{32}$$

as we might do when obtaining the propagator of a relativistic quark (for NRQCD, the propagator is spin diagonal in the Pauli matrices and trivial to invert). The inverse is

$$S = b_0 1 - \sum_{\mu=1}^{4} b_\mu \gamma_\mu \ , \qquad b_i = a_i \left(a_0^2 - \sum_{\mu=1}^{4} a_\mu^2 \right)^{-1} \ , \tag{33}$$

which is far more efficient than inverting a 4×4 matrix. Inversion of a general spin matrix (not of the form Eqn. (32)) is less efficient with an implicit representation, but this is irrelevant in most perturbative calculations.

Since all basis matrices except the identity are traceless, taking the trace of a spin matrix is a free operation in the implicit representation.

2.6.1. Implementation notes

In the HIPPY code, spin basis matrices are associated with monomials using an appropriate integer i, which is part of the `Entity` data structure. When terms are multiplied together, the factors ϕ_{jk} are absorbed into the amplitude f of the resulting monomial.

In the HPSRC code, we represent a spin matrix W as a defined type, `spinor`, which is encoded as a double array

$$(n; i_1, \ldots, i_n; w_1, \ldots, w_n) \equiv \sum_{k=1}^{n} w_k \Gamma_{i_k} \ . \tag{34}$$

It turns out to be significantly more efficient for the order of terms in this array to not necessarily match a standard order of basis elements $\{\Gamma_i\}$, hence the use of the index array i_k. In particular, this allows us to omit basis elements with zero coefficient.

Arithmetic operations have been overloaded to act appropriately on objects of this type, including implemention of the multiplication table. During inversion, an additional function argument, `short_spinor`, is used to employ the more efficient expression in Eqn. (33).

3. Even more realistic fermionic actions

Sec. 2 summarised the general method that was described in Ref. [17]. In general, fermionic actions are much more complicated than gluonic actions and several algorithmic improvements are needed to efficiently calculate the associated Feynman rules. We stress that by efficient we mean speed-ups of at least an order of magnitude.

The algorithms described in this section can be used independently or together, with the choice configurable at runtime of the code. All of these features have also been implemented using taylor-valued variables (as per Sec. 2.5) to provide automatic differentiation of Feynman rules.

3.1. Summands and factors

In many cases an action has a block-like structure that we can exploit to make the evaluation of the reduced vertices more efficient. This is particularly useful in the case of NRQCD and mNRQCD actions, which can be heuristically written as $\bar{\psi}(1 - ABCA)\psi$. Such an action can be expanded directly in the HiPPy code but the extremely large number of monomials makes this is inefficient (or impossibly slow and memory-hungry). It is clear why: there is often little scope for monomial compression between blocks. For instance, in (m)NRQCD, blocks AB and CA live on different timeslices of the lattice, and no compression is possible when combining them.

Instead, we recognise that the blocks are combined in a gauge covariant manner, so that in AB, for instance, each contour in B starts where a contour in A finishes. Summing over the start/end location of each block we obtain a convolution of terms and can use the convolution theorem to construct the overall reduced vertex (i.e. the momentum-space Fourier transform) from those of the individual blocks.

We refer to the action as being a sum over terms that we call "summands". In the above example, there are two. Each summand is the convolution of a number of "factors", with one factor in the first summand and four in the second.

The overall reduced vertex $Y_{F,r}$ is the sum of the reduced vertices for each summand. For each summand, $Y_{F,r}$ is calculated by combining the reduced vertices $Y^{(k)}_{F,r}$ for each of N factors that make up that summand, $k = 1 \ldots N$.

Table 1: The elements P in the set of ordered partitions, $P^{(r)}$, of the ordered set of the first r integers, $\{1, 2, \ldots, r\}$, for $r = 1$, 2 and 3.

r	P	$\lvert P\rvert$
1	$\{\{1\}\}$	1
2	$\{\{1,2\}\}$	1
	$\{\{1\},\{2\}\}$	2
3	$\{\{1,2,3\}\}$	1
	$\{\{1,2\},\{3\}\}$	2
	$\{\{1\},\{2,3\}\}$	2
	$\{\{1\},\{2\},\{3\}\}$	3

We generate these by expanding each factor of the action separately in the HiPPY code, with the convolution then carried out in the HPsrc code.

Here we give the method for constructing $Y_{F,r}$ for a summand with N factors for general r. Expressions for specific $r = 0 \ldots 3$ are given in Appendix A.

In giving the general expression, we first establish some useful notation. Consider the ordered set of the first r integers: $\{1, 2, \ldots, r\}$. We can form an ordered partition of this set: $\{P_1, P_2, \ldots, P_z\}$, where the cardinality $z \equiv |\{...\}|$ is the number of elements in the partition. The set of all such partitions we denote $P^{(r)}$. For instance, the partitions $P^{(r)}$ for $r = 1$, 2, 3 are shown in Table 1. Note that we do not consider unordered partitions (e.g. $\{\{1,3\},\{2\}\}$) because the gauge fields are explicitly ordered in the paths (Wilson lines) making up the action.

The general reduced vertex for a summand with N factors is then:

$$
\begin{aligned}
Y_{F,r}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r) = \\
\sum_{P\in P^{(r)}} \sum_{1\le n_1<n_2<\cdots<n_{|P|}\le N} \left\{ \prod_{Q\in P} \left[\left(\prod_{k=n_{i-1}+1}^{n_i-1} Y^{(k)}_{F,0}(\boldsymbol{p}_i,-\boldsymbol{p}_i) \right) \right.\right. \\
\left.\left. \times Y^{(n_i)}_{F,|Q|}(\boldsymbol{p}_i,-\boldsymbol{p}_{i+1};\boldsymbol{k}_{Q_1},\mu_{Q_1};\ldots;\boldsymbol{k}_{Q_{|Q|}},\mu_{Q_{|Q|}}) \right] \right. \\
\left. \times \left(\prod_{k=n_{|P|}+1}^{N} Y^{(k)}_{F,0}(-\boldsymbol{q},\boldsymbol{q}) \right) \right\} \quad (35)
\end{aligned}
$$

where $i = 1 \ldots |P|$ is the position of element Q in the ordered set P and $n_0 = 0$. Momentum is conserved in the diagram, so $\boldsymbol{p}_1 = \boldsymbol{p}$ and subsequent $\boldsymbol{p}_{i+1} = \boldsymbol{p}_i + \boldsymbol{k}_Q$ (with i defined as above). Here $\boldsymbol{k}_Q$ refers to the summed momenta for the set Q (e.g. $\boldsymbol{k}_{\{1,2\}} \equiv \boldsymbol{k}_1 + \boldsymbol{k}_2$), implying $\boldsymbol{p}_{|P|} = -\boldsymbol{q}$.

3.2. Two-level actions

Fermion actions often use fattened links to reduce discretisation errors in numerical simulations. By fattening or smearing a link, we will, in general, end up with an $N \times N$ complex colour matrix M, which can be expanded in powers of the gauge field A_μ. It is often the case that this matrix is reunitarised by projecting it back onto the gauge group $SU(N)$ or, more usually, simply back

onto the related group $U(N)$. Fattened links can then be further fattened, in an iterative procedure. An example of this is the HISQ action.

To complement these numerical simulations, we need to do perturbative calculations using the same actions. We confine our attention here to the HISQ action (and simpler variants of the same form, for testing), with an iterated, two-level smearing procedure with an intermediate reunitarisation:

$$U^{\mathrm{HISQ}} = (F_{\mathrm{ASQ}'} \circ P_{U(3)} \circ F_{\mathrm{FAT7}})[U] \tag{36}$$

where $U = \exp(gA)$ is the unsmeared gauge field, $P_{U(3)}$ denotes the polar projection onto $U(3)$ (as used in simulations, and *not* $SU(3)$ [47]), and the FAT7 and ASQ' (a slightly modified version of ASQ) smearings are defined in Ref. [23].

Straightforward application of the expansion algorithm above is theoretically possible but practically unfeasible: the number of monomials is huge and the memory requirements of the HiPPy code quickly become excessive. We get around this by taking advantage of the two-level structure inherent in the definition of the action and that the intermediate reunitarisation. This allows us to express the partially-smeared gauge field as a member of the associated gauge Lie algebra, allowing us to split the derivation and subsequent application of the Feynman rules into two steps.

In the first step, the Feynman rules for the outer (or "top") layer, the ASQ' action, are derived in the same way as before. Representing the reunitarised (and so far uncalculated) FAT7R smeared links by a new, Lie-algebra–valued gauge potential, B_μ:

$$U_\mu^{\mathrm{FAT7R}}(x) = (P_{U(3)} \circ F_{\mathrm{FAT7}})[U_\mu] = e^{B_\mu(x+\frac{1}{2}\hat{\mu})} \ , \tag{37}$$

we can use the HiPPy code to expand the ASQ' action in terms of B_μ as before. We obtain similar position-space contributions

$$V_r = \frac{g^r}{r!} \sum_i f_{r;i}^{\mathrm{ASQ}'} \bar{\psi}(x_{r;i}) B_{\mu_1}(v_{r;i,1}) \cdots B_{\mu_r}(v_{r;i,r}) \Gamma_{r;i} \psi(y_{r;i}) \tag{38}$$

that Fourier transform to give monomials of the usual form.

To complete the derivation of the HISQ Feynman rules, we also need to know the expansion of B_μ in terms of the original gauge potential A_μ. To obtain this, we write inner (or "bottom") layer, the FAT7-smeared link, as $F_{\mathrm{FAT7}}[U] = M = HW$, where $H^\dagger = H$ and $W \in U(3)$ (we suppress Lorentz and lattice site indices in the following). We again use the HiPPy code to obtain an expansion

$$M = c[\mathbf{1} + a_\mu A_\mu + a_{\mu\nu} A_\mu A_\nu + \ldots] \tag{39}$$

where c is a normalisation constant which, for simplicity in the text, we assume has been rescaled to unity. The notation here is, for instance,

$$a_{\mu\nu} A_\mu A_\nu = \sum_{x,y} a_{\mu\nu}(x,y) A_\mu(x + \frac{1}{2}\hat{\mu}) A_\nu(y + \frac{1}{2}\hat{\nu}) \ . \tag{40}$$

Then unitarity of W implies that $R \equiv MM^\dagger = H^2$ and hence $W = R^{-1/2}M$ using the expansion

$$R^{-1/2} = (1 + (R-1))^{-1/2} = 1 - \frac{1}{2}(R-1) + \frac{3}{8}(R-1)^2 + \ldots \tag{41}$$

Rearranging the result as $W = \exp(B)$, i.e.

$$B = \log(W) = (W-1) - \frac{1}{2}(W-1)^2 + \ldots \tag{42}$$

finally yields the desired expansion of B. These formulæ are implemented in this form in the HiPPy code.

Given this, we can now numerically reconstruct the HISQ Feynman rules for any given set of momenta from Eqn. (38) by a convolution of the ASQ' Feynman rules of Eqn. (38) with the expansion of B_μ in terms of A_μ, summing up all the different ways in which the gluons A_μ going into the vertex could have come from the fields B_μ appearing in Eqn. (38).

In more detail, we now separately have the expansions of the ASQ' action in terms of the fattened gauge fields B, and of B in terms of the unfattened gauge field A. To obtain the correct reduced vertex, we must find all the ways that we can get unfattened gluons of the correct Lorentz polarisations ("directions"). In doing this, we bear in mind that B_μ contains, in principle, gauge fields A_ν in all directions and not just $\nu = \mu$. Below we give an expression for the reduced vertex for general r. Explicit formulæ for $r \leq 3$, as implemented in the HPsrc code, are given in Appendix B. Using the partitions $P^{(r)}$ as before, the reduced vertex for a two-level action is::

$$\begin{aligned} Y_{F,r}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r) = \\ \sum_{P\in P^{(r)}} \sum_{\nu_1,\ldots,\nu_{|P|}} & Z_{F,|P|}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{k}_{P_1},\nu_1;\ldots;\boldsymbol{k}_{P_{|P|}},\nu_{|P|}) \\ & \times \prod_{Q\in P} X^{\nu_i}_{F,|Q|}(\boldsymbol{k}_{Q_1},\mu_{Q_1};\ldots;\boldsymbol{k}_{Q_{|Q|}},\mu_{Q_{|Q|}}) \end{aligned} \tag{43}$$

where $i = 1\ldots|P|$ is again the position of element Q in the ordered set P. In the algebra reduced vertex X, the momenta $\boldsymbol{k}_{Q_i}$ are each one of the arguments to the overall reduced vertex Y. As before, in the field reduced vertex Z, $\boldsymbol{k}_{P_j}$ refers to the summed momenta for the partition P_j.

In Fig. 1 we represent this expansion graphically for $r = 2$. Solid circles represent the $Y_{F,r}$, whilst crossed and open circles represent $Z_{F,r}$ and $X_{F,r}$ respectively. Top-level, fattened gluons B_μ are represented by dashed (brown) lines, whilst bottom-level, unfattened gluons A_μ are shown as wavy (blue) lines. The two sub-diagrams represent the two partition contributions listed in Eqn. (53).

As we shall discuss later, this partitioning translates naturally into blocks of code. For certain calculations, symmetries of the Feynman diagram will lead to the contributions of some of these blocks being zero. We can then improve the performance of the code by commenting them out in these circumstances.