Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 209 (2005) 340–353

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Automatically generating Feynman rules for improved lattice field theories

A. Hart [a,*], G.M. von Hippel [b,c], R.R. Horgan [b,*], L.C. Storoni [b]

[a] *School of Physics, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, UK*
[b] *DAMTP, CMS, University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, UK*
[c] *Department of Physics, University of Regina, Regina, SK, Canada S4S 0A2*

Received 22 November 2004; accepted 2 March 2005
Available online 23 May 2005

Abstract

Deriving the Feynman rules for lattice perturbation theory from actions and operators is complicated, especially when improvement terms are present. This physically important task is, however, suitable for automation. We describe a flexible algorithm for generating Feynman rules for a wide range of lattice field theories including gluons, relativistic fermions and heavy quarks. We also present an efficient implementation of this in a freely available, multi-platform programming language (Python), optimised to deal with a wide class of lattice field theories.
© 2005 Elsevier Inc. All rights reserved.

PACS: 11.15.Ha; 12.38.Gc

MSC: 81-04; 81T13; 81T15; 81T18; 81T25; 81V05; 65S05; 41A58

Keywords: Lattice field theory; Feynman rules; Perturbation theory

1. Introduction

Non-abelian quantum field theories such as QCD are believed to explain much of particle physics, at least at energy scales probed by current particle accelerators. Perturbative expansions of the theory do not, however, converge at hadronic energy scales. That, and the belief that non-perturbative physics may also contribute to certain states, makes the lattice regularisation of quantum field theories extremely important. Inherently non-perturbative calculations can then be carried out using Monte-Carlo simulations.

* Corresponding authors. Tel.: +44 131 650 5264 (A. Hart).
E-mail address: a.hart@ed.ac.uk (A. Hart).

Dividing space and time into a grid with lattice spacing a, however, excludes ultraviolet modes with momenta of π/a or higher. A renormalisation programme is, therefore, required to connect lattice measurements to their continuum counterparts. Such renormalisation factors are particularly important for QCD matrix elements and fixing the couplings and masses present in the Lagrangian. Renormalisation is also needed to determine the strong coupling α_s and to relate the lattice regularisation scale Λ_{lat} to the more familiar Λ_{QCD}. It is also used to "improve" the lattice actions in an attempt to reduce the discretisation errors at given lattice spacing.

In a limited number of cases, the renormalisation constants can be determined using non-perturbative techniques. Results at finite lattice spacing, however, can depend upon the method used (e.g. [1]), and non-perturbative methods do not cope well with mixing of operators under renormalisation. For these reasons, there is a strong interest in lattice perturbation theory.

Given that perturbation theory fails in low energy QCD, we may ask why it should work on the lattice. An argument for its use is given in [2]: the renormalisation factors may be thought of as compensating for the ultraviolet modes excluded by the lattice regulator. For typical lattices, $a \lesssim 0.1$ fm and the excluded modes have momenta in excess of 5 GeV. At these scales, the running QCD coupling α_s is small enough that perturbation theory should rapidly converge. The wide range of results recently reviewed in [3,4] show perturbation theory can be used for a large range of lattice QCD processes. It is an assumption that non-perturbative effects do not contribute on these short length scales. In a few cases, we can test this directly by comparing high order perturbative calculations with Monte-Carlo simulations at a range of weak couplings [5–9]). The non-perturbative contributions to the studied quantities are very small. Other comparisons, such as [1], cannot distinguish non-perturbative effects from higher loop perturbative corrections. It therefore remains that lattice perturbation theory provides the only systematically improvable method for determining the full range of renormalisation constants [3].

As in the continuum, the calculation of lattice Feynman diagrams is a two stage process. The lattice action and operators must first be Taylor expanded to give the propagators and vertices that form the Feynman rules (which we refer to as the "vertex expansion" stage). Following this, these rules must be used to construct and evaluate Feynman diagrams, possibly after algebraic simplification (the "Feynman diagram evaluation" stage).

The main obstacles in the latter task are the presence of Lorentz symmetry violating terms at finite lattice spacing and the complications of replacing momentum integrals by discrete sums. The calculations are, therefore, usually done using computer programs like Vegas [10], Form [11] or other proprietary mathematical packages.

Expanding the lattice action and operators to obtain Feynman rules is far more complicated than in the continuum. Firstly, lattice gauge fields are elements of the Lie group rather than the algebra of the gauge group. We must therefore expand exponentials of non-commuting fields to obtain the Feynman rules. Secondly, modern lattice theories contain a large number of irrelevant (in the renormalisation group sense of the word) 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.

There is, however, no unique prescription for these terms, and the choice depends on that quantities we are most interested in simulating. As a result, a large number of actions and operators are currently in use. Although the differences may be subtle, each choice provides a separate regularisation of QCD with its own set of renormalisation constants and, most relevantly here, Feynman rules. At present, the complications of the expansions has meant that the availability of renormalisation factors has lagged far behind developments in lattice improvement. In many cases, this has restricted the physical predictions obtained from the simulations.

As a result, there is a strong need for an automated method for deriving lattice Feynman rules in a flexible way for a range of different theories. The generation should be rapid enough not to constrain our choice of action, and to avoid errors we should be able to specify the action in a compact and intuitive manner (such as using nested link smearing prescriptions). The evaluation of the Feynman diagrams can be computationally intensive, and may be carried out on costly supercomputing facilities. Parsimony and software availability dictate that the rules should be separately calculable in advance, and rendered in a machine readable format that can be copied to any computer for later Feynman diagram evaluation.

In this paper we describe such a method.

Automated expansion of lattice actions[1] is not a new concept, having been described for gluonic actions by Lüscher and Weisz in 1986 [12]. An implementation of this has been used in [13–15]. A similar method is employed in [16]. We present here a new algorithm suited to expansion of not only gluonic actions, but also those of complicated relativistic fermionic actions and heavy quarks, such as in NRQCD. As in [12], the expansion is independent of the boundary conditions allowing, for instance, the use of twisted boundary conditions to regulate infrared divergences in a gauge-invariant manner [17,18] or otherwise change the discrete momentum spectrum [19]. We also describe details of an implementation of this algorithm which we have used for calculations of the renormalised anisotropy in gauge theories [20,21], to study the mean link in Landau gauge for tadpole improvement [9] and to measure the electromagnetic decays of heavy quark systems using NRQCD [22,23]. The code is flexible and can be easily extended to cope with a full range of problems, some of which we discuss in Section 5. We are happy to share this code with interested readers.

The structure of the paper is as follows. Our method clearly stems from [12] and in Section 2 we review their theory and notation. The algorithm itself differs markedly from [12], not least in being able to deal with fermionic actions, and is described in Section 3. In Section 4, we turn to implementation of the algorithm, explaining the steps taken to ensure the code can cope with the more complicated theories. Whilst the notation is tilted towards our version in the Python programming language, the optimisations are clearly applicable to any realisation of the algorithm. We present our conclusions in Section 5. Technical details of the data structures employed are relegated to Appendices.

2. The lattice

A cubical space-time lattice Λ in D dimensions consists of sites labelled by a vector $\boldsymbol{x} \in \Lambda$ with components that are integer multiples of a lattice spacing a, which we will set to be one (a (bare) lattice anisotropy can be introduced through rescaling of coupling constants in the action [20]). The directions of the lattice axes are labelled $\mu \in \{1,2,\ldots,D\}$. If $\boldsymbol{e}_\mu$ is a right-handed basis set consisting of unit vectors, we define corresponding backward vectors: $\boldsymbol{e}_{-\mu} = -\boldsymbol{e}_\mu$.

A path consisting of l links starting at site $\boldsymbol{x}$ can be specified on the lattice by an ordered set of signed integers, $s_i \in [-D,\ldots,-1,1,\ldots,D]$

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

The jth point on the path is

$$\boldsymbol{z}_j = \begin{cases} \boldsymbol{x}, & j = 0, \\ \boldsymbol{z}_{j-1} + a\boldsymbol{e}_{s_{j-1}}, & 0 < j \leqslant l \end{cases} \tag{2}$$

and the endpoint of the path is $\boldsymbol{y} \equiv \boldsymbol{z}_l$.

For a periodic lattice with L_μ sites in the μ direction (and volume $V = \prod_\mu L_\mu$) the momentum vectors are

$$\boldsymbol{k} = \frac{2\pi}{a}\left(\frac{\bar{k}_1}{L_1},\ldots,\frac{\bar{k}_D}{L_D}\right), \qquad 0 \leqslant \bar{k}_\mu < L_\mu, \quad \bar{k}_\mu \in \mathbb{Z} \tag{3}$$

[1] We shall understand the term "actions" to include measurement operators from now on.

and $\sum_{\boldsymbol{k}}$ stands for sums over the integers $\bar{k}_\mu$. The Fourier expansion of a field ϕ is

$$\tilde{\phi}(\boldsymbol{k}) = \sum_{\boldsymbol{x}} \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}}\phi(\boldsymbol{x}), \quad \phi(\boldsymbol{x}) = \frac{1}{V}\sum_{\boldsymbol{k}} \mathrm{e}^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}}\tilde{\phi}(\boldsymbol{k}). \tag{4}$$

Different boundary conditions (e.g. twisted [9,12,20]) change the colour factors and momentum spectrum. Since neither are used explicitly in the vertex expansion below, the same reduced vertex function output can be used in each case.

2.1. Matter fields

We now turn to the description of lattice fields. The notation follows [12].

The gauge field associated with a link is $U_{\mu>0}(\boldsymbol{x}) \in SU(N)$. Let U denote the full configuration of such links. The perturbative gauge potential associated with the link is defined through

$$U_{\mu>0}(\boldsymbol{x}) = \exp\left(agA_\mu\left(\boldsymbol{x} + \frac{a}{2}\boldsymbol{e}_\mu\right)\right) = \sum_{r=0}^{\infty} \frac{\left(agA_\mu(\boldsymbol{x} + \frac{a}{2}\boldsymbol{e}_\mu)\right)^r}{r!}, \tag{5}$$

where g is the bare coupling constant. The potential $A_\mu \in \mathrm{alg}(SU(N))$ is associated with the midpoint of the link. Expanding in the anti-Hermitian generators of $SU(N)$:

$$A_\mu = A_\mu^a T_a, \quad [T_a, T_b] = -f_{abc}T_c, \quad \mathrm{Tr}(T_a T_b) = -\tfrac{1}{2}\delta_{ab}. \tag{6}$$

We define $U_{-\mu}(\boldsymbol{x}) = U_\mu^\dagger(\boldsymbol{x} - a\boldsymbol{e}_\mu)$.

Quark fermion fields $\psi(\boldsymbol{x})$ transform according to the representation chosen for the generators T_a. From now on we assume this to be the fundamental representation (other choices will affect the colour factors, but not the underlying expansion algorithm).

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

$$L(\boldsymbol{x},\boldsymbol{y},U) \equiv \mathscr{L} : U = \prod_{i=0}^{l-1} U_{s_i}(\boldsymbol{z}_i) = \prod_{i=0}^{l-1} \exp\left[\mathrm{sgn}(s_i)agA_{|s_i|}\left(\boldsymbol{z}_i + \frac{a}{2}\boldsymbol{e}_{s_i}\right)\right]. \tag{7}$$

As all actions and operators can be written as a sum of Wilson lines (possibly terminated by fermion fields that are not themselves expanded), our goal is to efficiently render L as a Taylor series in the gauge potential in momentum space

$$L(\boldsymbol{x},\boldsymbol{y};A) = \sum_r \frac{(ag)^r}{r!} \sum_{\boldsymbol{k}_1,\mu_1,a_1} \cdots \sum_{\boldsymbol{k}_r,\mu_r,a_r} \tilde{A}_{\mu_1}^{a_1}(\boldsymbol{k}_1)\ldots\tilde{A}_{\mu_r}^{a_r}(\boldsymbol{k}_r) \times V_r(\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r). \tag{8}$$

We can write the vertex functions V_r as

$$V_r(\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r) = C_r(a_1,\ldots,a_r)Y_r^{\mathscr{L}}(\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r). \tag{9}$$

The matrix colour factor C_r plays the role of the Clebsch–Gordan factor

$$C_r(a_1,\ldots,a_r) = \prod_{i=1}^{r} T_{a_i}. \tag{10}$$

Up to differences in the colour trace structure of the action (e.g. a mixed fundamental/adjoint gauge action, and discussed in Appendix B), the C_r are path independent. We can therefore represent the vertex functions more efficiently by calculating just the expansion of the reduced vertex functions, $Y_r^{\mathscr{L}}$ (with an appropriate description of the colour trace structure where ambiguous). The reduced vertex function can be written as a sum of monomials

$$Y_r^{\mathscr{L}}(\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r)=\sum_{n=1}^{n_r}f_n\exp\frac{\mathrm{i}}{2}\left(\boldsymbol{k}_1\cdot\boldsymbol{v}_1^n+\cdots+\boldsymbol{k}_r\cdot\boldsymbol{v}_r^n\right). \tag{11}$$

For each combination of r Lorentz indices we have n_r terms, each with an amplitude f and the locations $\boldsymbol{v}$ of the r factors of the gauge potential. To simplify this expression, we have suppressed the dependence of f, $\boldsymbol{v}_i^n$ and n_r on the Lorentz structure. To construct Y for given momenta, we apply the $\boldsymbol{k}$'s to the position vectors of all monomials with the correct Lorentz indices.

The $\boldsymbol{v}$'s have been drawn from the locations of the midpoints of the links in the path $\mathscr{L}$. To avoid floating point ambiguities, it is therefore more convenient to express the components of all D-vectors as integer multiples of $\frac{a}{2}$ (accounting for the factor of $\frac{1}{2}$ in the exponent).

2.2. Realistic actions: the fermion sector

We begin our discussion of realistic lattice actions with the fermion sector. The most general gauge- and translation-invariant action can be written as

$$S_F(\psi,U)=\sum_{\boldsymbol{x}}\sum_{\mathscr{W}}h_{\mathscr{W}}\bar{\psi}(\boldsymbol{x})\Gamma_{\mathscr{W}}W(\boldsymbol{x},\boldsymbol{y},U)\psi(\boldsymbol{y}). \tag{12}$$

It consists of Wilson lines W defined by open paths $\mathscr{W}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{s})$. Associated with each path is a coupling constant $h_{\mathscr{W}}$ and a spin matrix $\Gamma_{\mathscr{W}}$ (which might be unity).

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

$$S_F(\psi,A)=\sum_r\frac{g^r}{r!}\sum_{\boldsymbol{k}_1,\mu_1,a_1}\cdots\sum_{\boldsymbol{k}_r,\mu_r,a_r}\tilde{A}_{\mu_1}^{a_1}(\boldsymbol{k}_1)\ldots\tilde{A}_{\mu_r}^{a_r}(\boldsymbol{k}_r)\times\sum_{\boldsymbol{p},\boldsymbol{q},b,c}\tilde{\bar{\psi}}^b(\boldsymbol{p})V_{F,r}(\boldsymbol{p},b;\boldsymbol{q},c;\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r)\tilde{\psi}^c(\boldsymbol{q}). \tag{13}$$

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

$$V_{F,r}(\boldsymbol{p},b;\boldsymbol{q},c;\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r)=\frac{1}{r!}\sum_{\sigma\in\mathscr{S}_r}\sigma\cdot C_{F,r}(b,c;a_1,\ldots,a_r)\sigma\cdot Y_{F,r}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r), \tag{14}$$

where σ is an element of the permutation group of r objects, $\mathscr{S}_r$, applied to the gluonic variables and normalised by the factor of (r!). The reduced vertex $Y_{F,r}=\sum_{\mathscr{W}}h_{\mathscr{W}}Y_{F,r}^{\mathscr{W}}$ is the sum of contributions from paths $\mathscr{W}$.

For all simple cases, the Clebsch–Gordan colour factor is the matrix element

$$C_{F,r}(b,c;a_1,\ldots,a_r)=(T_{a_1}\ldots T_{a_r})_{bc}. \tag{15}$$

The symmetrisation and calculation of colour factors will be carried out separately when the vertex functions are reconstructed in a Feynman diagram calculation.

The reduced vertex function has the structure

$$Y_{F,r}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r)=\sum_{n=1}^{n_r}\Gamma_nf_n\times\exp\left(\frac{\mathrm{i}}{2}\left(\boldsymbol{p}\cdot\boldsymbol{x}+\boldsymbol{q}\cdot\boldsymbol{y}+\boldsymbol{k}_1\cdot\boldsymbol{v}_1^n+\cdots+\boldsymbol{k}_r\cdot\boldsymbol{v}_r^n\right)\right). \tag{16}$$

As we do not use explicit representations of the spin matrices, it is important that each monomial retains the correct spin dependence Γ_n.

2.3. Realistic actions: the gluon sector

A general gluonic action is

$$S(\psi,U)=\sum_{\boldsymbol{x}}\sum_{\mathscr{P}}c_{\mathscr{P}}Re\mathrm{Tr}[P(\boldsymbol{x},\boldsymbol{x},U)], \tag{17}$$

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

$$S_G(A) = \sum_r \frac{g^r}{r!} \sum_{\boldsymbol{k}_1,\mu_1,a_1} \cdots \sum_{\boldsymbol{k}_r,\mu_r,a_r} \tilde{A}^{a_1}_{\mu_1}(\boldsymbol{k}_1)\ldots\tilde{A}^{a_r}_{\mu_r}(\boldsymbol{k}_r) \times V_{G,r}(\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r). \tag{18}$$

The Euclidean Feynman rule for the r-point gluon vertex function is $(-g^r V_{G,r})$, and the vertex $V_{G,r}$ is [12]

$$V_{G,r}(\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r) = \frac{1}{r!}\sum_{\sigma\in\mathscr{S}_r} \sigma\cdot C_{G,r}(a_1,\ldots,a_r)\sigma\cdot Y_{G,r}(\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r). \tag{19}$$

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

$$Y^{\mathscr{P}}_{G,r}(\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r) = \sum_{n=1}^{n_r} f_n \exp\left(\frac{\mathrm{i}}{2}\left(\boldsymbol{k}_1\cdot\boldsymbol{v}^n_1 + \cdots + \boldsymbol{k}_r\cdot\boldsymbol{v}^n_r\right)\right). \tag{20}$$

In most cases, we expect the lattice action to be real. For every monomial in Eq. (20), then, there must be a corresponding term

$$(-1)^r f^*_n \exp -\left(\frac{\mathrm{i}}{2}\sum_i \boldsymbol{k}_i\cdot\boldsymbol{v}^n_i\right). \tag{21}$$

We can therefore speed up the evaluation of the Feynman rules by removing the latter term, and replacing the exponentiation in Eq. (20) with “cos” for r even, and with “i sin” for r odd. Clearly, we must identify to which terms this has been applied. 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 the reduction (as discussed in Section 4).

If, in addition to the reality, the action has the form Eq. (17) with a single trace in the fundamental representation, the colour factors are

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

When symmetrising, a lot of the terms have similar Clebsch–Gordan factors:

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

We can therefore partly symmetrise the vertex over $\mathscr{Z}_r$ (the subgroup of cyclic permutations and inversion) at the expansion stage. The $\chi_r(\sigma)$ go into the amplitudes of the new terms coming from the partial symmetrisation:

$$\begin{aligned} V_{G,r}(\boldsymbol{k}_1,\mu_1,a_1;\ldots;\boldsymbol{k}_r,\mu_r,a_r) &= \sum_{\sigma\in\mathscr{S}_r/\mathscr{Z}_r} \sigma\cdot C_{G,r}(a_1,\ldots,a_r)\times\sigma\cdot Y'_{G,r}(\boldsymbol{k}_1,\mu_1;\ldots;\boldsymbol{k}_r,\mu_r), \\ Y'_{G,r} &= \sum_{\substack{\mathscr{P}\\ \sigma\in\mathscr{Z}_r}} c_{\mathscr{P}}\chi_r(\sigma)\sigma\cdot Y^{\mathscr{P}}_{G,r}. \end{aligned} \tag{24}$$

The advantage of doing this is that many of the extra monomials are equivalent, and we can therefore cut down significantly the number of exponentiation operations required to construct $V_{G,r}$. The number of remaining symmetrisation steps (to be carried out in the Feynman diagram code) is the number of cosets in $\mathscr{S}_r/\mathscr{Z}_r$ (one for $r \leqslant 3$, three for $r = 4$, etc.).

2.4. Diagram differentiation

There are many cases where Feynman diagrams need to be differentiated with respect to one or more momenta. Whilst this can be done numerically using an appropriately local difference operator, this can lead to numerical instabilities.

It is clear from Eq. (11), that we can easily construct the differentiated Feynman vertex. Let the momentum component we wish to differentiate with respect to be 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

$$\frac{\mathrm{d}}{\mathrm{d}q_\nu} Y_r^{\mathscr{L}}(\boldsymbol{k}_1, \mu_1; \ldots; \boldsymbol{k}_r, \mu_r) = \sum_{n=1}^{n_r} \frac{\mathrm{i} f_n}{2} \left(\tau_1 v^n_{1;\nu} + \cdots + \tau_r v^n_{r;\nu} \right) \times \exp \frac{\mathrm{i}}{2} \left(\boldsymbol{k}_1 \cdot \boldsymbol{v}^n_1 + \cdots + \boldsymbol{k}_r \cdot \boldsymbol{v}^n_r \right) \tag{25}$$

and so on for higher derivatives. We may therefore simultaneously calculate as many differentials as we need for the cost of one exponentiation. If this momentum expansion is placed into an appropriate data structure with overloaded operations, it is easy to create the Taylor series for a Feynman diagram by multiplying the vertex factors together. For examples of such codes, see [24].

It may also be necessary to differentiate diagrams with respect to parameters in the action, which may be present in different multiples in the amplitude of the monomials. Depending on the situation, we can separately expand parts of the action containing different powers of the parameter. Alternatively, we can use a single expansion and append a label to each monomial that records how many powers of the given parameter are contained in the amplitude. This label is then used in the parameter differentiation in the Feynman diagram code.

2.5. Recursive path definitions

So far we have assumed that the paths in the action are constructed from single links. This is, of course, always true but it is often more compact to specify the action as built from composite objects, such as smeared links, giving a separate prescription for the link smearing. For instance, the smeared link might be defined as the gauge covariant, weighted sum of the link and its adjoining staples

$$W_\mu(\boldsymbol{x}) = c_0 U_\mu(\boldsymbol{x}) + c_1 \sum_{\substack{\pm\nu \\ |\nu| \neq \mu}} U_\nu(\boldsymbol{x}) U_\mu(\boldsymbol{x} + \boldsymbol{e}_\nu) U^\dagger_\nu(\boldsymbol{x} + \boldsymbol{e}_\mu), \tag{26}$$

where the coefficients $c_{0,1}$ define the smearing method and are chosen to optimise certain aspects of the Monte-Carlo simulation. The smearing may also be defined recursively, with smeared links inserted into additional smearing recipes. Examples include the "HISQ" improved staggered fermions discussed in [25], where the links are first "FAT7", then "ASQ" smeared. Rather than multiplying out the paths in the action to give a large sum of tangled paths built from links, it is more convenient to reflect the nested improvement structure in the expansion algorithm itself.

This is done by first defining the mapping $U \rightarrow W$ as a sum of paths as in Eq. (1). For instance, Eq. (26) is represented by path $\{\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{e}_\mu; [\mu]\}$ with coupling c_0 plus $\{\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{e}_\mu; [\nu, \mu, -\nu]\}$, $\{\boldsymbol{x}, \boldsymbol{x} + \boldsymbol{e}_\mu; [-\nu, \mu, \nu]\}$ with coupling c_1. Call the smearing definitions $\mathscr{W}$. An action is then compactly specified by a path $\mathscr{P}$ of composite objects. These are in turn are defined by an ordered list $[\mathscr{W}_1, \mathscr{W}_2, \ldots, \mathscr{W}_n]$ representing each step of nested improvement. The full field is then defined by the recursion:

$$\begin{aligned}
&\mathscr{W}_1 : U = W_1(U),\\
&\mathscr{W}_2 : W_1(U) = W_2(W_1(U)),\\
&\mathscr{W}_3 : W_2(U) = W_3(W_2(U)),\\
&\qquad\qquad\vdots\\
&\mathscr{W}_i : W_{i-1}(U) = W_i(W_{i-1}(U)),\\
&\qquad\qquad\vdots\\
&\mathscr{P}\text{:}\ W_n(U) = P(W_n(U)).
\end{aligned} \tag{27}$$

3. An expansion algorithm

In this section, we present a new algorithm for carrying out the Taylor expansion of lattice actions in a manner suited to computer implementation.

We start by defining an object that represents a single term in the Taylor expansion in Eq. (11). We call this an "entity" E, and it is an ordered list

$$E = (\mu_1, \ldots, \mu_r; \boldsymbol{x}, \boldsymbol{y}; \boldsymbol{v}_1, \ldots, \boldsymbol{v}_r; f). \tag{28}$$

The order of the entity is r. For instance, a single link comes from a path $\mathscr{L} = \{\mathbf{0}, \boldsymbol{e}_\mu; [\mu]\}$, and the rth term in its expansion in Eq. (5) is represented as

$$E_r = (\underbrace{\mu, \ldots, \mu}_{r\text{ terms}}; \mathbf{0}, 2\boldsymbol{e}_\mu; \underbrace{\boldsymbol{e}_\mu, \ldots, \boldsymbol{e}_\mu}_{r\text{ terms}}; 1). \tag{29}$$

Note that in units of $\frac{a}{2}$ the endpoint is $2\boldsymbol{e}_\mu$ and the midpoint $\boldsymbol{e}_\mu$. The reduced vertex function is a set of all the entities of all orders in the expansion, and we call this a "field" $F = \{E\}$.

In practise, we build a Wilson line by concatenating smaller paths (the smallest being the link). We therefore define the multiplication of two fields so as to give the Taylor expansion of the resulting, longer contour. The product is, therefore, the ordered product of each entity from the first contour with every entity from the second:

$$\begin{aligned}
F(\mathscr{L}_1 * \mathscr{L}_2) &= F(\mathscr{L}_1) * F(\mathscr{L}_2),\\
&= \{E^i * E^j \quad \forall E^i \in F(\mathscr{L}_1),\ E^j \in F(\mathscr{L}_2)\}.
\end{aligned} \tag{30}$$

To keep gauge covariance, entity E^j must be translated to start at the end point of E^i. The order of the product entity is the sum of those of the constituents. Further details are given in Appendix C.

Addition of fields should represent the expansion of a sum of gluonic paths. We therefore simply combine the lists of entities

$$F(\mathscr{L}_1 + \mathscr{L}_2) = F(\mathscr{L}_1) + F(\mathscr{L}_2). \tag{31}$$

In general, F will be a redundant representation of the polynomial containing two or more equivalent entities. As each entity represents a monomial which requires a computationally expensive exponentiation, we construct a compression operation which compares all pairs of entities in F and combines them if they are equivalent

$$[E^i, E^j] \to [E] \quad \text{if} \quad E^i \equiv E^j, \quad \text{where } E = \left(\mu_1^i, \ldots, \mu_r^i; \boldsymbol{x}^i, \boldsymbol{y}^i; \boldsymbol{v}_1^i, \ldots, \boldsymbol{v}_r^i; f^i + f^j\right). \tag{32}$$

Assuming we have a translationally equivalent theory, entities need only be equivalent up to translation by a constant vector. For details, see Eq. (C.4).

4. A practical implementation

In this section, we describe an implementation of the algorithm in a programming language called PYTHON. The PYTHON interpreter is freely available for a wide range of computational platforms at [26]. The complexity of some physical actions (notably high order NRQCD) require the implementation to be CPU and memory efficient. This can be achieved in PYTHON with appropriate programming techniques, and without sacrificing the object orientation and superior list handling features of the language.

As a first step, we choose a maximum order for the Taylor expansion. Any entity of higher order is discarded. The entities and all sub-lists (including D-vectors) are encoded as "tuples", which are immutable list objects to which the native hash function can be applied.

To minimise the size of the dictionary, it is important that the field does not contain entities that are equivalent. We choose data structures for the entity and field specifically to prevent this. Firstly, we exploit translational invariance of the action to arrange that all paths start at the origin ($\boldsymbol{x} = \boldsymbol{0}$). Entity equivalence in Eq. (C.4) then follows from an item by item tuple comparison.

This comparison is most efficiently done using a hash table (i.e., an associative array). The PYTHON "dictionary" is a native implementation of this, where a "key" indexes an "object". In our implementation, each dictionary entry represents a monomial in the reduced vertex function. The key is a list of all information in the entity bar the amplitude, which becomes the object indexed by this key. Searching for equivalent entities now amounts to enquiring if the key already exists in the dictionary. As all key entries are integer, we do not need to worry about machine precision issues. We can significantly speed up the hashing by omitting the now redundant $\boldsymbol{x} = \boldsymbol{0}$ from the entity data structure. Independently, a moderate performance gain can come from archiving the hash values for each entity.

On a technical note, we find a slight speed-up associated with implementing the field as a two-level dictionary. The upper level keys are tuples of Lorentz indices. These each index a lower level dictionary of all entities with the appropriate Lorentz structure.

If, on inserting a new entity into a field structure, an equivalent entity exists, rather than adding the new item to the field its amplitude is merely combined with that of the existing entity. If the new amplitude $f^i + f^j = 0$, the entry is removed. This test is robust for integer arithmetic. Otherwise, the absolute value is compared to some tolerance, e.g. 10^{-8}. We may worry about rounding errors: near $\boldsymbol{k}_i = \boldsymbol{0}$ the reduced vertex monomials are all adding in phase, and the deleted amplitudes may add to give a significant contribution. We therefore use a tolerance smaller than is finally required for the path expansion. As a last step only, we apply the final tolerance cut. The numerical values of the intermediate and final tolerances are found by trial and error, looking for robustness in the number of terms in the expansion. It is worth pointing out that PYTHON carries out all floating point arithmetic in double precision.

In the gluonic case of closed, traced contours the endpoint $\boldsymbol{y}$ is physically irrelevant. By ignoring it, we can identify more entities that are equivalent and further reduce the dictionary size. To find these, for each entity we impose $\boldsymbol{x} = \boldsymbol{y} = \boldsymbol{0}$ whilst translating all the $\boldsymbol{v}$'s such that $\boldsymbol{v}_1 = \boldsymbol{e}_{\mu_1}$. The field dictionary is then rehashed, looking for newly equivalent terms. Of course, care must be taken only to do this as a final step, and not to then multiply such contours together.

As discussed in Section 2.5, a recursive action definition is more compact and less error-prone. Each smearing definition (such as "APE" or "ASQ") is predefined, and labelled by a unique string. Each path in the action is specified by three items: the coupling for this path, a list of signed directions and a list of link improvement method names. A full action (gluonic or fermionic) is specified by a list of such path specifications. For each path, the expansion routine is executed recursively to implement Eq. (27), converting it into a field that is fed to the next level of the nested link improvement. Finally, these fields can be manipulated or combined, before being output.

In some cases, we need to combine complex conjugate monomials as per Eq. (21). This is most easily done by noting that $E = 2E_{\text{real}} - E^*$. We loop over all unprocessed entities $E \in F$, inserting $-E^*$ into F

and marking E as now processed (and doubling its amplitude). This marking can be done by adding a Boolean flag to the entity data structure in Appendix C.

Once we have the monomials, we have a choice of output format dictated by whether the output is to be read into the Feynman diagram evaluation code at compile time or run time. The former has the advantage that the compiler may be able to optimise the construction of the vertex functions; the disadvantage is that the size of the vertex functions may lead to code that is too long for the compiler to handle. Reading in the data at run time avoids this, but may in principle lead to slower code. In either case, the Python code can be easily adapted to produce the correct output format.

As an example, we describe the run time case. The output consists of a single ASCII file for each order of the perturbative expansion. Each file contains multiple entries, which could be a single line, or multiple lines for clarity. The entry contains the information in a single entity as whitespace separated values. For later storage in the Feynman diagram code, it is convenient if the Lorentz indices are represented as a single integer in base D: $n(\boldsymbol{\mu}) = \sum_{i=1}^{r}(\mu_i - 1)D^{i-1}$. It is also useful if the entries for given $\boldsymbol{\mu}$ are consecutively numbered (although the order does not matter). The file is terminated by a blank entry with $n(\boldsymbol{\mu}) = -1$.

In the Feynman diagram code, a set of arrays should be defined to hold the vertex function data. For languages without allocatable arrays, we can arrange for the Python to write a set of compile time header files that create arrays of the correct dimensions for a given set of vertex functions.

We use a set of Fortran90 modules to read in the data files, which can serve as a template for other languages.

5. Conclusions

Simulation of Symanzik and radiatively improved lattice field theory actions has become very popular in recent years. Associated renormalisation factors (and, indeed, the radiative improvement itself) can be systematically calculated using lattice perturbation theory. The complicated nature of the improved actions and operators has, however, contributed to a backlog in this perturbative renormalisation programme.

Having a flexible method for generating the Feynman rules automatically is crucial to overcoming this backlog, and permitting a greater range of renormalisation factors to be calculated. This paper provides just such a method, that is well suited to expanding all sectors of lattice QCD: gluons, relativistic fermions and heavy quarks. In addition to the Taylor expansion algorithm, an efficient implementation in the Python programming language is described, exploiting useful features of this language.

Particular strengths of this algorithm include coping with arbitrary spin and colour trace structures in the action, allowing a nested definition of link improvement and an intuitive way of defining the action to be expanded.

The code is also very flexible, and can be adapted to deal with most wrinkles met in perturbative expansion. The first of these is tadpole improvement. Tadpole (or mean field) improvement aims to speed the convergence of perturbation theory through dividing each link in the action by a factor u [27]. We can use perturbation theory to calculate $u = 1 + d_1\alpha_s + \cdots$ as an expansion in the coupling, and treat the quantum effects as radiative counterterms in the action with couplings $n_t d_i(\alpha_s)^i$ (plus combinatoric factors), where n_t was the number of factors of links in the path. We can expand the action as before, but must now do a separate expansion of the radiative counterterms. Consider building an action from links smeared as in Eq. (26). When we expand $W_\mu(\boldsymbol{x})W_\mu(\boldsymbol{x} + \boldsymbol{e}_\mu)$ we will get a link combination $[v, \mu, -v, v, \mu, v]$, and the question is whether such a term should be given a tadpole improvement factor of u^{-4} or u^{-6}. The former is more in keeping with the philosophy of mean field improvement, but in a simulation the latter is more convenient. The procedure, of course, is that the perturbative action should follow what is used in simulation. Either convention can be followed by assigning to each entity knowledge of the full length of the path from which it is derived. In the latter case of no cancellation, these lengths simply add on entity multiplication. In the former case, some directional knowledge must be maintained to allow factors of u to cancel. Terms with different numbers of tadpole factors should be grouped separately; for this reason, n_t should be included in the key in the Python implementation of the entities and fields.

This adaptability also makes the expansion algorithm described here useful in, for instance, chiral perturbation theory [28] or the double expansion needed for stochastic perturbation theory [6,29].

By describing and making available these algorithms and tools, we hope that lattice field theory calculations can reach a point where the choice of lattice action is not constrained by the availability (or not) of renormalisation constants.

Acknowledgments

We are pleased to acknowledge useful discussions with I.T. Drummond and Q. Mason. A.H. is grateful to the U.K. Royal Society for support.

Appendix A. Spin algebra

Each Wilson line in an action has an associated spin matrix. Where this is not uniformly unity, we must keep track of which spin matrix applies to each term in the reduced vertex. We do this by adding a single integer label to the entity list, τ. For Dirac gamma matrices $0 \leqslant \tau \leqslant 15$, whilst for Pauli sigma matrices $0 \leqslant \tau \leqslant 3$. By convention, $\tau = 0$ is the identity.

When two spin factors are multiplied, the product is proportional to a single element of the spin algebra

$$\tau^i \tau^j = \tfrac{1}{2}[\tau^i, \tau^j] + \tfrac{1}{2}\{\tau^i, \tau^j\} = \varepsilon_k \tau^k \quad \text{for some } \varepsilon_k \in \mathbb{R} \tag{A.1}$$

(with no sum over k). This reduction can be easily encoded in the Python vertex generation code through a small dictionary where each key (τ^i, τ^j) indexes a list: $[\tau^k, \varepsilon_k]$.

Appendix B. Pattern lists

A Wilson line may be composed of a number of parts to which separate colour traces may have been applied. This will affect the value of the associated Clebsch–Gordan coefficient. Physical examples include the traceless field strength operator, $U_{\mu\nu} - \mathrm{Tr}\,U_{\mu\nu}$, and the adjoint Yang–Mills action, $\mathrm{Tr}U_{\mu\nu}\mathrm{Tr}U^\dagger_{\mu\nu}$. In the former case, for instance, second order monomials either have colour structure of the form $(T_aT_b)_{cd}$ or $\mathrm{Tr}(T_aT_b)\delta_{cd} = -\frac{1}{2}\delta_{ab}\delta_{cd}$.

As we do not calculate the Clebsch–Gordan coefficients in the vertex generation program, we need a method for distinguishing whether given gauge potentials within an entity are untraced, all traced together or in separate colour traces. This distinction is also important in ensuring we do not compress together entities with different colour structures. We distinguish these cases by adding an extra entry to the entity called a pattern list.

A pattern list of order r is $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_r)$. Each positive integer element ω_i is associated with the corresponding factor of the gauge potential A_i. If A_i has not been traced, $\omega_i = 0$. All gauge potentials with the same value of the pattern component are understood to be contained in a single colour trace. For instance, $A_1A_3\mathrm{Tr}(A_2A_4)$ would have a pattern list $(0,1,0,1)$.

Applying a colour trace to an entity modifies only the pattern list $\boldsymbol{\omega} \to \boldsymbol{\omega}'$, with

$$\omega'_i = \begin{cases} 1 + \max(\omega_1, \ldots, \omega_r), & \omega_i = 0, \\ \omega_i, & \omega_i \neq 0. \end{cases} \tag{B.1}$$

We stress that the actual value of ω_i has no meaning. It is therefore convenient to arrange at all stages that the first non-zero element in the list is 1, the second 2, etc. Taking the example above, $\mathrm{Tr}[A_1A_3\mathrm{Tr}(A_2A_4)] = \mathrm{Tr}(A_1A_3)\mathrm{Tr}(A_2A_4)$ has a relabelled pattern list (1,2,1,2). Gauge invariance precludes application of a trace to entities for which $\boldsymbol{x} \neq \boldsymbol{y}$ (i.e., to contours which are not closed). The $SU(N)$ generators are traceless, so when a colour trace applies to only one factor of the gauge potential, we may delete the entity (such as when tracing an entity of order $r = 1$).

We define multiplication $\boldsymbol{\omega}^1 * \boldsymbol{\omega}^2$ by

$$(\omega^i_1, \ldots, \omega^i_{r_i}) * (\omega^j_1, \ldots, \omega^j_{r_j}) = (\omega^i_1, \ldots, \omega^i_{r_i}, \omega'_1, \ldots, \omega'_{r_j}), \tag{B.2}$$

where

$$\omega'_k = \begin{cases} 0, & \omega^j_k = 0, \\ \omega^j_k + \max(\omega^i_1, \ldots, \omega^i_{r_i}), & \omega^j_k \neq 0. \end{cases} \tag{B.3}$$

When all elements of $\boldsymbol{\omega}$ are not 1, the symmetries of Eq. (23) are not present. The group of symmetrisation operations carried out in the vertex generation code must, therefore, be reduced from $\mathscr{L}_r$, or it may be simpler to postpone all symmetrisation until the Feynman diagram are evaluated for specific momenta.

We note that pattern lists can also be used to label the taking of real or imaginary parts in an action in a similar way, using positive and negative entries to distinguish the two.

Appendix C. Entity algebra

Taking into account Appendices A and B, an entity E consists of

$$E = (\boldsymbol{\mu}; \boldsymbol{x}, \boldsymbol{y}; \boldsymbol{v}_1, \ldots, \boldsymbol{v}_r; \boldsymbol{\omega}; \tau; f). \tag{C.1}$$

The colour trace pattern list $\boldsymbol{\omega}$ and spin index τ are optional and need not be included in all situations. The start site $\boldsymbol{x}$ may also be omitted (see Section 4).

The complex conjugate entity E^* has amplitude $(-1)^r f^*$ and the sign of all D-vectors reversed.

Multiplication by a scalar $p \in \mathbb{C}$ acts only on the amplitude

$$pE = (\boldsymbol{\mu}; \boldsymbol{x}, \boldsymbol{y}; \boldsymbol{v}_1, \ldots, \boldsymbol{v}_r; \boldsymbol{\omega}; \tau; pf). \tag{C.2}$$

We translate an entity by D-vector $\boldsymbol{c} \in \Lambda$ using

$$T_{\boldsymbol{c}}E = (\mu_1, \ldots, \mu_r; \boldsymbol{x} + \boldsymbol{c}, \boldsymbol{y} + \boldsymbol{c}; \boldsymbol{v}_1 + \boldsymbol{c}, \ldots, \boldsymbol{v}_r + \boldsymbol{c}; \boldsymbol{\omega}; \tau; f). \tag{C.3}$$

Two entities are said to be equivalent if the lists can be rendered identical under a translation and rescaling:

$$E^i \equiv E^j \quad \text{iff } \exists\ \boldsymbol{c} \in \Lambda, \quad p \in \mathbb{C} \quad \text{s.t.} \quad T_{\boldsymbol{c}}E^i = pE^j. \tag{C.4}$$

Non-commutative multiplication of two entities is defined by

$$E' = E^i * E^j = (\boldsymbol{\mu}'; \boldsymbol{x}^i, \boldsymbol{y}^j + \boldsymbol{C}; \boldsymbol{v}^i_1, \ldots, \boldsymbol{v}^i_{r_i}, \boldsymbol{v}^j_1 + \boldsymbol{C}, \ldots, \boldsymbol{v}^j_{r_j} + \boldsymbol{C}; \boldsymbol{\omega}^1 * \boldsymbol{\omega}^2; \tau^k; f'), \tag{C.5}$$

i.e., the path from which the second entity was derived is first translated by a vector $\boldsymbol{C} = \boldsymbol{y}^i - \boldsymbol{x}^j$, to start where the first finished. The resulting entity is of order $r = r_i + r_j$. The Lorentz list $\boldsymbol{\mu}'$ is the concatenation of lists $\boldsymbol{\mu}^i + \boldsymbol{\mu}^j$. The spin indices yield $\tau^i\ \tau^j = \varepsilon_k \tau^k$ as per Eq. (A.1). Note the amplitude $f' = \varepsilon_k\,{}^rC_{r_i} f^i f^j$ contains a combinatoric factor arising from having separated out the (r!) Taylor expansion factors from the amplitude in Eq. (8).

The action of the permutation operator $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_r)$ on a list $\boldsymbol{l}$ yields $(l_{\sigma_1}, \ldots, l_{\sigma_r})$. We can apply it to entities which are closed $\boldsymbol{x} = \boldsymbol{y}$, simply traced ($\omega_i = 1\ \forall i$) and where the real part has been taken (to ensure Eq. (22) holds)

$$\sigma \cdot E = (\sigma \cdot \boldsymbol{\mu};\boldsymbol{x},\boldsymbol{x};\boldsymbol{v}_{\sigma_1},\ldots,\boldsymbol{v}_{\sigma_r};\sigma \cdot \boldsymbol{\omega};\tau;\chi_r(\sigma)f), \tag{C.6}$$

noting in this case that $\sigma \cdot \boldsymbol{\omega} \equiv \boldsymbol{\omega}$. Eq. (23) defines $\chi_r(\sigma)$.

By extension Eqs. (C.2), (C.3) and (C.6), and the colour trace are applied to a field F by operating on each of its constituent entities $E \in F$.

References

[1] T. Bhattacharya, R. Gupta, W.-J. Lee, S.R. Sharpe, Order a improved renormalization constants, Phys. Rev. D 63 (2001) 074505, Available from: <hep-lat/0009038>.

[2] G.P. Lepage, Redesigning lattice QCD, Available from: <hep-lat/9607076>.

[3] S. Capitani, Lattice perturbation theory, Phys. Rep. 382 (2003) 113–302, Available from: <hep-lat/0211036>.

[4] H.D. Trottier, Higher-order perturbation theory for highly-improved actions, Nucl. Phys. Proc. Suppl. 129 (2004) 142–148, Available from: <hep-lat/0310044>.

[5] G.P. Lepage, P.B. Mackenzie, N.H. Shakespeare, H.D. Trottier, Perturbative two- and three-loop coefficients from large β Monte-Carlo, Nucl. Phys. Proc. Suppl. 83 (2000) 866–871, Available from: <hep-lat/9910018>.

[6] F. Di Renzo, L. Scorzato, A consistency check for renormalons in lattice gauge theory: β^{-10} contributions to the SU(3) plaquette, JHEP 10 (2001) 038, Available from: <hep-lat/0011067>.

[7] R. Horsley, P.E.L. Rakow, G. Schierholz, Separating perturbative and non-perturbative contributions to the plaquette, Nucl. Phys. Proc. Suppl. 106 (2002) 870–872, Available from: <hep-lat/0110210>.

[8] H.D. Trottier, N.H. Shakespeare, G.P. Lepage, P.B. Mackenzie, Perturbative expansions from Monte-Carlo simulations at weak coupling: Wilson loops and the static-quark self- energy, Phys. Rev. D 65 (2002) 094502, Available from: <hep-lat/0111028>.

[9] A. Hart, R.R. Horgan, L.C. Storoni, Perturbation theory vs. simulation for tadpole improvement factors in pure gauge theories, Phys. Rev. D 70 (2004) 034501, Available from: <hep-lat/0402033>.

[10] G.P. Lepage, A new algorithm for adaptive multidimensional integration, J. Comput. Phys. 27 (1978) 192.

[11] J.A.M. Vermaseren, New features of FORM, Available from: <math-ph/0010025>.

[12] M. Lüscher, P. Weisz, Efficient numerical techniques for perturbative lattice gauge theory computations, Nucl. Phys. B 266 (1986) 309.

[13] M.A. Nobes, H.D. Trottier, G.P. Lepage, Q. Mason, Second order perturbation theory for improved gluon and staggered quark actions, Nucl. Phys. Proc. Suppl. 106 (2002) 838–840, Available from: <hep-lat/0110051>.

[14] M.A. Nobes, H. Trottier, One loop renormalization of Fermilab fermions, Nucl. Phys. Proc. Suppl. 119 (2003) 461–463, Available from: <hep-lat/0209017>.

[15] M.A. Nobes, H.D. Trottier, Progress in automated perturbation theory for heavy quark physics, Nucl. Phys. Proc. Suppl. 129 (2004) 355–357, Available from: <hep-lat/0309086>.

[16] B. Alles, M. Campostrini, A. Feo, H. Panagopoulos, Lattice perturbation theory by computer algebra: a three loop result for the topological susceptibility, Nucl. Phys. B 413 (1994) 553–566, Available from: <hep-lat/9301012>.

[17] M. Luscher, P. Weisz, On-shell improved lattice gauge theories, Commun. Math. Phys. 97 (1985) 59.

[18] M. Lüscher, P. Weisz, Computation of the action for on-shell improved lattice gauge theories at weak coupling, Phys. Lett. B 158 (1985) 250.

[19] G.M. de Divitiis, R. Petronzio, N. Tantalo, On the discretization of physical momenta in lattice QCD, Phys. Lett. B 595 (2004) 408–413, Available from: <hep-lat/0405002>.

[20] I.T. Drummond, A. Hart, R.R. Horgan, L.C. Storoni, One loop calculation of the renormalised anisotropy for improved anisotropic gluon actions on a lattice, Phys. Rev. D 66 (2002) 094509, Available from: <hep-lat/0208010>.

[21] I.T. Drummond, A. Hart, R.R. Horgan, L.C. Storoni, The contribution of $\mathcal{O}(\alpha)$ radiative corrections to the renormalised anisotropy and application to general tadpole improvement schemes, Phys. Rev. D 68 (2003) 057501, Available from: <hep-lat/0307010>.

[22] I.T. Drummond, A. Hart, R.R. Horgan, L.C. Storoni, Lattice perturbation theory for gluonic and fermionic actions, Nucl. Phys. Proc. Suppl. 119 (2003) 470–475, Available from: <hep-lat/0209130>.

[23] A. Gray, A. Hart, G. von Hippel, R. Horgan, S-wave QCD/NRQCD matching for the vector annihilation current at $\mathcal{O}(\alpha_s v^2)$ (in preparation).

[24] Automatic differentiation packages are discussed at http://www.autodiff.org.

[25] E. Follana, A. Hart, C.T.H. Davies, The index theorem and universality properties of the low-lying eigenvalues of improved staggered quarks, Phys. Rev. Lett. 93 (2004) 241601, Available from: <hep-lat/0406010>.
[26] Available from: <http://www.python.org>.
[27] G.P. Lepage, P.B. Mackenzie, On the viability of lattice perturbation theory, Phys. Rev. D 48 (1993) 2250–2264, Available from: <hep-lat/9209022>.
[28] G. von Hippel et al. (in progress).
[29] F. Di Renzo, L. Scorzato, Numerical stochastic perturbation theory for full QCD, JHEP 0410 (2004) 73, Available from: <hep-lat/0410010>.