FEDILA Software Documentation

Version 1.0

Prepared by:

M. Costa, C. Costa, P. Chrysanthis, G. Spanoudes, H. Panagopoulos

Rinnoco Ltd and University of Cyprus

Email: info@rinnoco.com

January 18, 2025

This document contains the complete documentation for using the FEDILA software for calculating Feynman diagrams. It covers installation, configuration, and operational guidance along with examples and troubleshooting.

A Guide to Calculating Feynman Diagrams with FEDILA software

M. Costa,[1,2,3,*] C. Costa,[1,†] P. Chrysanthis,[1,‡] G. Spanoudes,[3,§] and H. Panagopoulos[3,¶]

[1]*Rinnoco Ltd, Limassol, CY-3047, Cyprus*

[2]*Department of Mechanical Engineering and Material Science and Engineering, Cyprus University of Technology, Limassol, CY-3036, Cyprus*

[3]*Department of Physics, University of Cyprus, Nicosia, CY-1678, Cyprus*

FEDILA is a pioneering software solution designed to revolutionize analytic computations of the fundamental forces in the universe. Its design focuses on optimizing computations in the continuum but also in a discreted version of the theory on a spacetime lattice. This documentation and tutorial series aims to provide comprehensive guidance on utilizing FEDILA software for calculating Feynman diagrams. FEDILA provides a flexible Mathematica package capable of handling a diverse spectrum of renormalization schemes, including a continumm Gauge Invariant Renormalization Scheme, as well as of extracting information from Supersymmetric Theories.

We present explanations of the underlying principles and step-by-step tutorials to effectively harness the capabilities of each Mathematica command.

1. INTRODUCTION

FEDILA is a state-of-the-art software package for perturbative calculations designed for the in-depth exploration of physical properties at the microscopic scale, particularly within the framework of Quantum Field Theories (QFTs). Developed using the symbolic high-level Mathematica language, FEDILA provides a versatile and robust environment for theoretical physicists and researchers engaged in the study of QFTs. This documentation serves as both an introduction to the software and a guide to its various features and functionalities.

One of the central challenges in QFT is dealing with infinities that arise in calculations. Renormalization is the process of redefining quantities to make these infinities manageable and physically meaningful. FEDILA supports several renormalization schemes and several regularizations:

- **Gauge Invariant Renormalization Scheme (GIRS)**: This is a continuum-based approach that preserves gauge invariance, ensuring that the physical predictions of the theory remain consistent with the underlying symmetries. GIRS is particularly useful in non-Abelian gauge theories and is crucial for maintaining the integrity of calculations in gauge theories.
- **Dimensional Regularization**: FEDILA incorporates dimensional regularization, a technique that extends the number of dimensions in space-time to regulate divergent integrals. By working in non-integer dimensions, this method allows for the systematic handling of infinities, making it a staple in QFT calculations.
- **Lattice Regularization**: This non-perturbative approach involves discretizing space-time into a lattice, enabling the study of QFTs beyond perturbation theory. Lattice regularization is particularly important for numerical simulations and for investigating QFTs in strong coupling regimes.

2. INSTALLATION AND SETUP

Since FEDILA is a Mathematica package, you must first install Mathematica. We provide a link to the instructions for installing the Wolfram Mathematica platform.

To begin using FEDILA, follow these steps:

1. **System Requirements**: Ensure that your system meets the necessary requirements for running Mathematica and that you have the appropriate version installed.

[*]Electronic address: marios.c@rinnoco.com
[†]Electronic address: costa.c@rinnoco.com
[‡]Electronic address: panos@rinnoco.com
[§]Electronic address: spanoudes.gregoris@ucy.ac.cy
[¶]Electronic address: haris@ucy.ac.cy

2. **Installation**: Download the FEDILA package from the official repository and install it within your Mathematica environment. Detailed installation instructions are provided in the package's README file.

   To install the package, copy the tar file `FEDILA-main.zip` to your disk and unpack it with:

   ```
   >> unzip FEDILA-main.zip
   ```

   Change to the Fedila directory:

   ```
   >> cd FEDILA-main
   ```

   Open the file `input.m` and replace `"username"` with your own username. After making the changes, save the file.

   ```
   >> emacs input.m
   [Replace "username" with your username]
   ```

   For better organization of the calculations, we suggest creating two additional directories for each project:

   ```
   >> mkdir fortranfiles
   >> mkdir outfiles
   ```

   Open a Mathematica kernel and you can read the input.m. To read input.m, you need to provide the path where the Fedila folder is saved. The reader can follow the inputs In[1] to In[100] in Mathematica to become more familiar with this software.

```
In[1]:= << /home/username/FEDILA-main/input.m
```

Now, all the commands needed to calculate a Feynman diagram can be used in your Mathematica kernel. Note that these new Mathematica commands start with a lowercase letter, in contrast to the commands from the Mathematica library, which start with an uppercase letter.

Getting Started

The software is designed to be user-friendly, with a command structure that is intuitive for those familiar with QFT and Mathematica. After launching Mathematica and loading the FEDILA package by reading the appropriate `input.m` file, users should become familiar with the software's notation. In general, our commands use the following notations:

- nDim is the number of dimensions of the theory.
- e is related to the dimension (nDim) of spacetime, through: nDim = 4 - 2 e.
- Nc is the number of colors of the theory.
- Nf is the number of flavors of the theory.
- im is the imaginary unit.
- hv = 0 (1) in naive ('t Hooft-Veltman) definition of γ_5.
- k[i] is the momentum of the i^{th} leg of the vertex.
- q[i] is the momentum of the i^{th} external leg of the Feynman diagram.
- p[i] is the loop momenta of the Feynman diagram.
- a[i] is the i^{th} color external index in the adjoint representation.
- c[i] is the i^{th} color internal index in the adjoint representation.
- af[i] is the i^{th} color external index in the fundamental representation.
- cf[i] is the i^{th} color internal index in the fundamental representation.

- mu[i] (or nu[i]) is the ith external Lorentz index.
- rho[i] is the ith Lorentz index to be summed over.
- fnu[i] is the ith Dirac index.
- fl[i] is the ith flavor index.
- f[a,b,c] are the antisymmetric structure constants.
- epsilon[mu,nu,rho,sigma] is the Ricci tensor.
- d[a,b,c] is the symmetric trace of a 3-generator product: d[a,b,c] = $2Tr(T^aT^bT^c + T^aT^cT^b)$, where T is the generator of the gauge group. This notation takes into account the ciclicity of the trace.
- traceGen[a,b,...,c,d] is the trace of the product of generators: $Tr(T^aT^b...T^cT^d)$. For example, traceGen[a,b] = $\delta_{ab}/2$ and traceGen[ ] = Nc.
- productGen[i,a,b,...,c,d,j] is the product of generators, where the first and the last arguments denote color indices in the fundamental representation: $(T^aT^b...T^cT^d)_{ij}$.
- delm[a,b] is the Kronecker delta for Lorentz indices.
- delc[a,b] is the Kronecker delta for color indices.
- delf[a,b] is the Kronecker delta for flavor indices.
- delta[a + b + ...] is the Dirac delta.
- deltaL[a,b,...,c] is a Kronecker delta whose 2 or more arguments are Lorentz indices.
- deltaColor[a,b] is a Kronecker delta of two color indices in the adjoint representation.
- traceDirac[a,b,...,c,d] is the trace of a product of Dirac matrices. The arguments of traceDirac can be: gamma5, nu[_] (an external Lorentz index taking values: 1,2,3,4), rho[_] (a dummy Lorentz index taking values: 1,2,3,4,...,nDim) or X, Y (unspecified Dirac matrices; each may appear at most once). For example: traceDirac[gamma5,nu[3],X,rho[2],gamma5,Y] = $\mathrm{Tr}(\gamma_5\gamma_{\nu_3}\mathrm{X}\gamma_{\rho_2}\gamma_5\mathrm{Y})$.
- productDirac[i,a,b,...,c,d,j] is a product of Dirac matrices, where the first and the last arguments denote the spinor indices. For example: productDirac[i,gamma5,nu[3],X,rho[2],gamma5,Y,j] = $(\gamma_5\gamma_{\nu_3}\mathrm{X}\gamma_{\rho_2}\gamma_5\mathrm{Y})_{ij}$

Further notations are provided in Section 3.5.

3. COMMANDS OF THE SOFTWARE

3.1. Creation of vertices

The initial step in evaluating each Feynman diagram involves determining the algebraic expressions of the vertices. Each calculation focuses on evaluating Feynman diagrams that encompass various unique vertices. These vertices have distinct characteristics as they are derived from different actions and operators, such as improved gluon and fermion actions/operators, the Supersymmetric QCD (SQCD) action/operators, actions incorporating "background" and "quantum" gauge fields, and actions using stout links, among others.

The primary task in this section is to describe the Mathematica commands for calculating these vertices, which includes both pure gluon vertices and those involving fermions. The procedures of compute the lattice vertices involve the manipulation of the products of the link variables (like Wilson loops or Wilson lines), such as expanding the links in terms of the gluon field up to the required order, and performing Fourier transformations of fields to represent vertices in momentum space.

The conventions for the Fourier transformations are:

$$\text{Quark fields:}\quad \tilde{\psi}(k) = \int d^4x\, e^{-ik\cdot x}\, \psi(x), \tag{1}$$

$$\text{Squark fields:}\quad \tilde{A}_\pm(k) = \int d^4x\, e^{\mp ik\cdot x}\, A_\pm(x), \tag{2}$$

$$\text{Gluon fields:}\quad \tilde{u}_\mu(k) = \int d^4x\, e^{-ik\cdot x}\, u_\mu(x), \tag{3}$$

$$\text{Gluino fields:}\quad \tilde{\lambda}(k) = \int d^4x\, e^{-ik\cdot x}\, \lambda(x), \tag{4}$$

$$\text{Ghost fields:}\quad \tilde{c}(k) = \int d^4x\, e^{-ik\cdot x}\, c(x). \tag{5}$$

In general, an n-point vertex can be expressed as:

$$\frac{1}{(2\pi)^{4n}} \int d^4k_1 \ldots d^4k_n \sum_{\substack{\mu_1,\ldots,\mu_n \\ a_1,\ldots,a_n}} \left(\prod_{i=1}^{n} \tilde{X}^{a_i}_{\mu_i}(k_i) \right) V^{a_1,\ldots,a_n}_{\mu_1,\ldots,\mu_n}(k_1,\ldots,k_n) \tag{6}$$

where k_j denote momenta; a_j are color indices either in the adjoint or in the fundamental representation; μ_j are Lorentz indices. $\tilde{X}^{a_i}_{\mu_i}(k_i)$ are fields of the theory, which appear in the vertex, with momentum k and Lorentz (color) index μ (a). Note that only gluon fields have Lorentz indices.

As an example, we provide a lattice discretized SQCD action where we extend Wilson's formulation of the QCD action to encompass SUSY partner fields as well. In this standard discretization quarks, squarks and gluinos are defined on the lattice sites, while gluons are defined on the links of the lattice: $U_\mu(x) = e^{igaT^\alpha u^\alpha_\mu(x+a\hat{\mu}/2)}$; α is a color index in the adjoint representation of the gauge group. This formulation leaves no SUSY generators intact, and it also breaks chiral symmetry; thus, the need for fine-tuning will arise in numerical simulations of SQCD. For Wilson-type quarks and gluinos, the Euclidean action $\mathcal{S}^L_{\rm SQCD}$ on the lattice becomes (in the Wess-Zumino gauge):

$$\begin{aligned}
\mathcal{S}^L_{\rm SQCD} \;=\; & a^4 \sum_x \Big[\frac{N_c}{g^2} \sum_{\mu,\,\nu} \left(1 - \frac{1}{N_c}\,{\rm Tr} U_{\mu\,\nu}\right) + \sum_\mu {\rm Tr}\left(\bar{\lambda}\gamma_\mu \mathcal{D}_\mu \lambda\right) - a\frac{r}{2}{\rm Tr}\left(\bar{\lambda}\mathcal{D}^2\lambda\right) \\
+ & \sum_\mu \left(\mathcal{D}_\mu A^\dagger_+ \mathcal{D}_\mu A_+ + \mathcal{D}_\mu A_- \mathcal{D}_\mu A^\dagger_- + \bar{\psi}\gamma_\mu \mathcal{D}_\mu \psi \right) - a\frac{r}{2}\bar{\psi}\mathcal{D}^2\psi \\
+ & i\sqrt{2}g\big(A^\dagger_+ \bar{\lambda}^\alpha T^\alpha P_+ \psi - \bar{\psi} P_- \lambda^\alpha T^\alpha A_+ + A_- \bar{\lambda}^\alpha T^\alpha P_- \psi - \bar{\psi} P_+ \lambda^\alpha T^\alpha A^\dagger_- \big) \\
+ & \frac{1}{2} g^2 (A^\dagger_+ T^\alpha A_+ - A_- T^\alpha A^\dagger_-)^2 - m(\bar{\psi}\psi - m A^\dagger_+ A_+ - m A_- A^\dagger_-) \Big],
\end{aligned} \tag{7}$$

where: a is the lattice spacing, $U_{\mu\nu}(x) = U_\mu(x) U_\nu(x+a\hat{\mu}) U^\dagger_\mu(x+a\hat{\nu}) U^\dagger_\nu(x)$, and a summation over flavors is understood in the last three lines of Eq. (7). The 4-vector x is restricted to the values $x = na$, with n being an integer 4-vector. The terms proportional to the Wilson parameter, r, eliminate the problem of fermion doubling, at the expense of breaking chiral invariance. In the limit $a \to 0$ the lattice action reproduces the continuum one. The bare couplings for the Yukawa and quartic terms (last two lines of Eq.(7)) need not coincide with the gauge coupling g; this requirement is imposed on the respective renormalized values.

The definitions of the covariant derivatives are as follows:

$$\begin{aligned}
\mathcal{D}_\mu\lambda(x) &\equiv \frac{1}{2a}\Big[U_\mu(x)\lambda(x+a\hat\mu)U_\mu^\dagger(x) - U_\mu^\dagger(x-a\hat\mu)\lambda(x-a\hat\mu)U_\mu(x-a\hat\mu)\Big] &(8)\\
\mathcal{D}^2\lambda(x) &\equiv \frac{1}{a^2}\sum_\mu\Big[U_\mu(x)\lambda(x+a\hat\mu)U_\mu^\dagger(x) - 2\lambda(x) + U_\mu^\dagger(x-a\hat\mu)\lambda(x-a\hat\mu)U_\mu(x-a\hat\mu)\Big] &(9)\\
\mathcal{D}_\mu\psi(x) &\equiv \frac{1}{2a}\Big[U_\mu(x)\psi(x+a\hat\mu) - U_\mu^\dagger(x-a\hat\mu)\psi(x-a\hat\mu)\Big] &(10)\\
\mathcal{D}^2\psi(x) &\equiv \frac{1}{a^2}\sum_\mu\Big[U_\mu(x)\psi(x+a\hat\mu) - 2\psi(x) + U_\mu^\dagger(x-a\hat\mu)\psi(x-a\hat\mu)\Big] &(11)\\
\mathcal{D}_\mu A_+(x) &\equiv \frac{1}{a}\Big[U_\mu(x)A_+(x+a\hat\mu) - A_+(x)\Big] &(12)\\
\mathcal{D}_\mu A_+^\dagger(x) &\equiv \frac{1}{a}\Big[A_+^\dagger(x+a\hat\mu)U_\mu^\dagger(x) - A_+^\dagger(x)\Big] &(13)\\
\mathcal{D}_\mu A_-(x) &\equiv \frac{1}{a}\Big[A_-(x+a\hat\mu)U_\mu^\dagger(x) - A_-(x)\Big] &(14)\\
\mathcal{D}_\mu A_-^\dagger(x) &\equiv \frac{1}{a}\Big[U_\mu(x)A_-^\dagger(x+a\hat\mu) - A_-^\dagger(x)\Big] &(15)
\end{aligned}$$

Note that in Eqs. (12)-(15), in order not to involve more than two lattice points, we do not use the symmetric derivative.

By analogy to the continuum case, a discrete version of a gauge-fixing term, together with the compensating ghost field term, must be added to the action, in order to avoid divergences from the integration over gauge orbits; these terms are the same as in the non-supersymmetric case. Although these terms can be found in literature, we present them here for the sake of completeness:

$$S_{GF}^L = \frac{1}{2\alpha}a^2\sum_x\sum_\mu \mathrm{Tr}\left(u_\mu(x+a\hat\mu/2) - u_\mu(x-a\hat\mu/2)\right)^2. \qquad (16)$$

$$\begin{aligned}
S_{Ghost}^L \;=\; & 2a^2\sum_x\sum_\mu \mathrm{Tr}\{(\bar c(x+a\hat\mu) - \bar c(x))(c(x+a\hat\mu) - c(x) &(17)\\
&+ig[u_\mu(x+a\hat\mu/2), c(x)] + \frac{1}{2}ig[u_\mu(x+a\hat\mu/2), c(x+a\hat\mu) - c(x)]\\
&-\frac{1}{12}g^2[u_\mu(x+a\hat\mu/2),[u_\mu(x+a\hat\mu/2), c(x+a\hat\mu) - c(x)]])\} + \mathcal{O}(g^3).
\end{aligned}$$

Similarly, a standard "measure" term must be added to the action, in order to account for the Jacobian in the change of integration variables: $U_\mu \to u_\mu$:

$$S_M^L = \frac{g^2N_c}{12}a^2\sum_x\sum_\mu \mathrm{Tr}\left(u_\mu(x+a\hat\mu/2)^2\right) + \mathcal{O}(g^4). \qquad (18)$$

It is important to assign unique dummy indices and momenta to the different powers of the fields; this is the only one of many parts of the computation that are straightforward by hand, but not in an automated computer evaluation. We write V in the form

$$\begin{aligned}
V = C_1\left(L_1(E_1+E_2+\ldots) + L_2(E_3+E_4+\ldots) + \ldots\right)\\
+ C_2\left(L_3(E_5+E_6+\ldots) + L_4(E_7+E_8+\ldots) + \ldots\right) + \ldots
\end{aligned} \qquad (19)$$

where C_i are 'colour structures', L_i are 'Lorentz structures' and E_i are monomials in trigonometric functions of momentum components. For instance, the 4-point vertex of gluons of QCD can be expressed as:

$$
\begin{aligned}
&V(k_1,k_2,k_3,k_4) = -g^2(2\pi)^4\delta(k_1+k_2+k_3+k_4)\mathrm{Tr}(T^{\alpha_1}T^{\alpha_2}T^{\alpha_3}T^{\alpha_4})\times\\
&\Big[\delta_{\mu_1\mu_2\mu_3\mu_4}\left(\frac{2}{3}-\frac{2}{3}\sum_\rho\cos\left(k_{1\,\rho}\right)+\frac{1}{2}\sum_\rho\cos\left(k_1+k_2\right)_\rho\right)\\
&+\delta_{\mu_1\mu_2\mu_3}\left(-\frac{4}{3}\sin\left(\frac{k_{4\,\mu_1}}{2}\right)\sin\left(\frac{k_{4\,\mu_4}}{2}\right)+2\sin\left(\frac{k_{4\,\mu_1}}{2}\right)\sin\left(\frac{(2k_1+k_4)_{\mu_4}}{2}\right)+2\sin\left(\frac{k_{4\,\mu_1}}{2}\right)\sin\left(\frac{(2k_3+k_4)_{\mu_4}}{2}\right)\right)\\
&+\delta_{\mu_1\mu_2}\delta_{\mu_3\mu_4}\left(\cos\left(\frac{(k_3+k_4)_{\mu_1}}{2}\right)\cos\left(\frac{(k_3+k_4)_{\mu_3}}{2}\right)-2\cos\left(\frac{(k_3-k_4)_{\mu_1}}{2}\right)\cos\left(\frac{(k_3+k_4)_{\mu_3}}{2}\right)\right)\\
&+\delta_{\mu_1\mu_3}\delta_{\mu_2\mu_4}\left(\cos\left(\frac{(k_1-k_3)_{\mu_2}}{2}\right)\cos\left(\frac{(k_2-k_4)_{\mu_1}}{2}\right)\right)\Big], \qquad (20)
\end{aligned}
$$

where T^a is a generator of the gauge group and ρ is an 'internal' Lorentz indices, appearing in structures which break rotational symmetry on the lattice. It is essential to take advantage of the fact that all vertices are completely symmetric with respect to the interchange of external lines. We use this symmetry to simplify the expressions for the vertices in three steps:

1. Minimize the color structures, for example, transform $\sum_c f^{a_1a_3c}f^{a_2a_4c}$ to the form $\sum_c f^{a_1a_2c}f^{a_3a_4c}$ using color algebra. The software includes a function, which is called 'colorExpand' and performs color simplifications in the expression in an efficient way, without expanding subexpressions which do not contain color indices.

2. Utilize the residual symmetry within each color structure to reduce all associated Lorentz structures to their simplest forms.

3. For each color-times-Lorentz structure, use its residual symmetry to minimize the number of accompanying monomials.

It is worth mentioning that for any given theory, once we calculate the vertices, we save them, and we can subsequently use them. The argument "vertices" in the command "contract" (see Sec. 3.2) will be of the form: {{Ngluon, Nghostpair, Ngluinobar, Ngluino, Nquarkbar, Nquark, NAplusdag, NAplus, NAminusdag, NAminus}, expression}, where: Ngluon is the number of gluon legs, and similarly for all other Nxxx (dag stands for dagger), and expression is the expression for the vertex itself. Therefore, the Mathematica expression for the 4-gluon vertex is:

```
{{4, 0, 0, 0, 0, 0, 0, 0, 0, 0}
- g^2 spur[c[1],c[2],c[3],c[4]] * delta[k[1]+k[2]+k[3]+k[4]]*
    (delm[mu[1],mu[2],mu[3],mu[4]] *
        ( 2/3 -2/3 c2[2k[1],rho[1]] +1/2 c2[2k[1]+2k[2],rho[1]])
 + delm[mu[1],mu[2],mu[3]] *
        (-4/3 s2[k[4],mu[1]] s2[k[4],mu[4]] +2 s2[k[4],mu[1]] s2[2k[1]+k[4],mu[4]]
        +2 s2[k[4],mu[1]] s2[2k[3]+k[4],mu[4]])
+ delm[mu[1],mu[2]]delm[mu[3],mu[4]] *
                    (c2[k[3]+k[4],mu[1]] c2[k[3]+k[4],mu[3]]
                     -2   c2[k[3]-k[4],mu[1]] c2[k[3]+k[4],mu[3]])
+ delm[mu[1],mu[3]]delm[mu[2],mu[4]] *
                    (c2[k[1]-k[3],mu[2]] c2[k[2]-k[4],mu[1]]))}
```

The convention for assigning indices and momenta to different fields in a vertex follows the order: gluon, gluino, antiquark, quark, antisquark (+), antisquark (−), squark (+), squark (−), antighost, ghost.

The first command presented in this tutorial is how to make fermionic (quark) vertices particular for stout links[1]

[1] Stout smearing according to Ref. [1]: Each link $U_\mu(x) = exp(i\,g\,a\,A_\mu(x+a\hat{\mu}/2))$ is replaced by a stout link $\widetilde{U}_\mu(x)$ defined as [2]

$$\widetilde{U}_\mu(x) = e^{i\,Q_\mu(x)}\,U_\mu(x)\,, \qquad (21)$$

where the definition of $Q_\mu(x)$ is

$$Q_\mu(x) = \frac{\omega}{2\,i}\left[V_\mu(x)U_\mu^\dagger(x) - U_\mu(x)V_\mu^\dagger(x) - \frac{1}{3}\mathrm{Tr}\left(V_\mu(x)U_\mu^\dagger(x) - U_\mu(x)V_\mu^\dagger(x)\right)\right]\,. \qquad (22)$$

ω is a tunable parameter, called a stout smearing parameter, and $V_\mu(x)$ represents the sum over all staples associated with the link, $U_\mu(x)$.

in the fermionic part: $\psi(x)\widetilde{U}_\mu(x)\psi(x)$ for one smearing step, the same command contains also the Background Field (BF) technique [2]:

```
makestoutvertexU[nQ_,nB_]
```

where 'nQ' is the number of quantum gluons and 'nB' is the number of background gluons of the vertex. It also returns an expression multiplyed with the parameter 'w' that corresponds to the omega parameter (ω) of the stout step, which can be set to 0 if no stout smearing is needed.

Let us extract the lattice vertex with gluon-antiquark-quark fields. There are two contributions. The first comes from $\sum_x \sum_\mu \bar{\psi}(x)\gamma_\mu \mathcal{D}_\mu \psi(x) = \frac{1}{2}\sum_x \sum_\mu \bar{\psi}(x)\Big[U_\mu(x)\psi(x+\hat{\mu}) - U^\dagger_\mu(x-\hat{\mu})\psi(x-\hat{\mu})\Big] = \frac{1}{2}\sum_x\sum_\mu \Big[\bar{\psi}(x)U_\mu(x)\psi(x+\hat{\mu}) - \bar{\psi}(x+\hat{\mu})U^\dagger_\mu(x)\psi(x)\Big]$, where we omit powers of the lattice spacing, which can be recovered through dimensional analysis. The second contribution is coming from the Wilson term: $-\frac{r}{2}\sum_x \bar{\psi}(x)\mathcal{D}^2\psi(x) = -\frac{r}{2}\sum_x\sum_\mu \bar{\psi}(x)\Big[U_\mu(x)\psi(x+\hat{\mu}) - 2\psi(x) + U^\dagger_\mu(x-\hat{\mu})\psi(x-\hat{\mu})\Big] = -\frac{r}{2}\sum_x\sum_\mu\Big[\bar{\psi}(x)U_\mu(x)\psi(x+\hat{\mu}) - 2\bar{\psi}(x)\psi(x) + \bar{\psi}(x+\hat{\mu})U^\dagger_\mu(x)\psi(x)\Big]$.

Figure 1 shows this vertex. Using some commands, we are able to construct the vertex from this part:

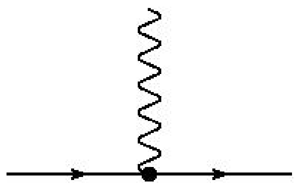

FIG. 1: Interaction lattice vertex with gluon-antiquark-quark fields. A wavy (solid) line represents gluons (quarks).

```
In[2]:=  makestoutvertexU[1,0] /. w -> 0 ;

In[3]:= 1/2 dtrace[fmu[2],mu[1],fmu[3]] * (% * phi2[2k[3], mu[1]] - (% * phi2[-2k[2], mu[1]] /.
                              im g -> -im g)) +
        (-r/2)* (% * phi2[2k[3], mu[1]] + (% * phi2[-2k[2], mu[1]] /.
                              im g -> -im g)) ;

In[4]:= % /. phi2[a_,b_]*phi2[c_,b_] :> phi2[a+c,b] ;

In[5]:= % /. delta[k[1]-k[2]+k[3]] phi2[k[1]+a_,b_] :> delta[k[1]-k[2]+k[3]] phi2[k[2]-k[3]+a,b] ;

In[6]:= % /. phi2[a_,b_] :> c2[a,b] + im s2[a,b] ;

In[7]:= (Expand[%] /. im^2 -> -1) // canonical // simplifydelm ;

In[8]:= Vertex[value] = { {1, 0, 0, 0, 1, 1, 0, 0, 0, 0}, %} ;
```

Note that $phi2[k,\mu] = \exp(ik_\mu/2)$, $c2[k,\mu] = \cos(k_\mu/2)$ and $s2[k,\mu] = \sin(k_\mu/2)$. Furthermore, in the above we use also the command 'simplifydelm' so as to simplify contractions in Lorentz indices.

```
simplifydelm[expression]
```

[2] On the lattice, the background field technique [3, 4] can be approached in more than one way. Different lattice actions may be chosen and the precise way in which the background field is introduced is arbitrary to some extent. However, the differences between the choices of lattice actions should be irrelevant in the continuum limit. The background field on the lattice is introduced by decomposing the gauge link variable as follows [3] (Q_μ: quantum field, A_μ: background field):

$$\begin{aligned} U_\mu(x) &= U^Q_\mu(x)U^B_\mu(x), \\ U^Q_\mu(x) &\equiv e^{ig_0 Q_\mu(x)}, \\ U^B_\mu(x) &\equiv e^{iag_0 B_\mu(x)} \end{aligned} \tag{23}$$

where $Q_\mu(x) = Q^a_\mu(x)T^a$ is the quantum field, $B_\mu(x) = B^a_\mu(x)T^a$ is the background field, and T^a are the generators of SU(N) with the convention of $\text{Tr}(T^aT^b) = \delta^{ab}/2$.

The command ‘canonical’ processes an expression by checking the function f when it has the argument $-x$. If $f(-x)$ appears, the function applies one of the following rules: If f is even, it simplifies $f(-x)$ to $f(x)$, meaning the function’s value remains the same when its argument changes sign. If f is odd, it simplifies $f(-x)$ to $-f(x)$, meaning the function’s value is negated when its argument changes sign.

```
canonical[expression]
```

We also present a command for clover-like vertices in which we need to put the list of the links that connect quark and antiquark fields in the clover term of the fermion action: $\sigma_{\mu\nu}\psi(x)U_{\mu\nu}(x)\psi(x)$ This command has the following form:

```
makecloververtex[listOFlinks_, nQ_, nB_]
```

where the first argument is the list of links, and by definition, the ’listOFlinks’ for $U_{\mu\nu}$ is {1,2,-1,-2}. By making the appropriate substitutions, one can obtain pure gluonic vertices; we leave this exercise to the reader to reproduce, for example, Eq. (20).

3.2. Contraction among vertices

The next step in evaluating Feynman diagrams is the contraction among vertices, which is done automatically once the algebraic expressions of the vertices and the “incidence matrix”, of the diagram are specified. This incidence matrix is used to systematically organize and track the contractions between pairs of field operators. This matrix is a square $n \times n$ matrix for each type of contraction where n is the number of the used vertices.

The result of the contraction is a preliminary expression for the diagram. This is followed by simplifications of the color dependence, Dirac matrices, and tensor structures. Additionally, symmetries of the theory, such as periodicity, reflection, charge conjugation, parity and hypercubic symmetry, are fully utilized to reduce the complexity and proliferation of the algebraic expressions.

Note that before contraction, we must ensure the symmetrization of the vertices and the proper renaming of indices (Lorentz, color) and momenta. We have the functions ‘symmetrizeauto’, ‘symmetrize’, ‘symmetrizepartially’ which are used by the function ‘contract’; given a vertex which contains several identical fields (typically gluons), these functions symmetrize the vertex w.r.t. these fields (only to the extent that is needed, for the sake of economy). As a side note, we have also the ‘antisymmetrize’ function in order to antisymmetrize expressions over some indices. The function ‘contract’ applies contractions leading to a Feynman diagram and have the following arguments:

- ‘vertices’ which is a list of N vertices, which may contain the fields of the theory (gluons, gluinos, ghosts, quarks and/or squarks (plus/minus)).
- ‘contractionsG’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of gluons joining the i^{th} to the j^{th} vertex.
- ‘contractionsGh’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among antighosts from the i^{th} and ghosts from the j^{th} vertex.
- ‘contractionsQ’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among antiquarks from the i^{th} and quarks from the j^{th} vertex.
- ‘contractionsg’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among antigluino from the i^{th} and gluino from the j^{th} vertex.
- ‘contractionsgbgb’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among antigluinos from the i^{th} and the j^{th} vertex.
- ‘contractionsgg’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among gluinos from the i^{th} and the j^{th} vertex.
- ‘contractionsAp’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among squark $A_+^\dagger$ from the i^{th} and A_+ from the j^{th} vertex.
- ‘contractionsAm’ an $N \times N$ incidence matrix whose $(\mathrm{ij})^{\mathrm{th}}$ element indicates the number of contractions among squark $A_-^\dagger$ from the i^{th} and A_- from the j^{th} vertex.

All these matrices, except the incidence matrix of gluons, are not symmetric; in all these cases, the function ‘contract’ should be called separately for each version of the matrices. The function ‘contract’ must be called for each Feynman diagram.

Firstly, the function ‘contract’ detects how many fields are contained in each vertex and how many of them remain external after contraction. We have to note that the indices are assigned to external fields without any symmetrization; such a symmetrization (over indices of same-type external fields) will have to be performed by hand after the command ‘contract’ has finished. This will amount to adding together various mirror diagrams. However, the function ‘contract’ totally symmetrizes the vertex list.

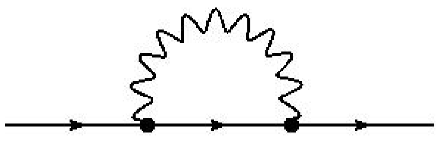

FIG. 2: One-loop Feynman diagrams contributing to the 2-pt Green’s function $\langle\psi(x)\bar{\psi}(y)\rangle$.

Specifically, we assume that a vertex could contain more than one gluon or squark field and we symmetrize over them. If two vertices are connected by n lines, only one of the two need be symmetrized with respect to those lines. The aforementioned command combines all symmetrized vertices by renaming all indices (and momenta); indices assigned to contracted legs are converted to internal ones. Afterward, both internal and external indices are arranged in ascending order so as to have unique names. This command performs self-contractions of vertices, and contractions of the i$^{\text{th}}$ and j$^{\text{th}}$ vertex; these two operations are performed simultaneously, in order to respect the order of indices. In particular, for each contraction, we substitute the related tree-level propagator of each pair of fields.

FEDILA features the following contract command that automates the process of applying Wick’s theorem. It involves contracting pairs of fields and replacing them with propagators. It has the following form:

```
contract[vertices_, contractionsG_, contractionsGh_, contractionsg_,
         contractionsgbgb_, contractionsgg_, contractionsQ_,
 contractionsAp_, contractionsAm_]
```

Taking the part of the total vertex that we created earlier (Vertex[value]), we apply Wick’s theorem to generate our first diagram, as shown in Fig. 2.

```
In[9]:= contract[{Vertex[value],Vertex[value]},
              {{0,1},{1,0}},
              {{0,0},{0,0}},
              {{0,0},{0,0}},{{0,0},{0,0}},{{0,0},{0,0}},
              {{0,1},{0,0}},
              {{0,0},{0,0}},{{0,0},{0,0}}] ;
```

The final expression of the Green’s function will depend on (after the integration of the loop momentum):

- q[i] (external momenta),
- nu[i] (Lorentz index),
- a[i] (color adjoint index),
- fnu[i] (Dirac index),
- af[i] (color fundamental index),

where the first values of the index ’i’ refer to external gluons, the next values to external antigluinos, and so on, in the order: {gluon, antigluino, gluino, antighost, ghost, antiquark, quark, $A_+^\dagger$, A_+, $A_-^\dagger$, A_-}, i=1,2,...,n (n: number of external fields). We have to take into account that in case the vertices of the Green’s function contain “open” indices, these indices must be given values which do not clash with the values assigned to external fields by “contract”. Note that combinatorial factors, or extra factors of (-1) for fermion closed loops, are not included in this function.

The expression of Green’s functions may also depend on:

- p[i] (internal loop momenta)

and on

- rho[i] (Lorentz index with a summation).

3.3. Simplifications of the color dependence

We can simplify color structures by using the identity:

$$T^a_{ij}T^a_{kl} = \frac{1}{2}\delta_{il}\delta_{kj} - \frac{1}{2N}\delta_{ij}\delta_{kl}, \tag{24}$$

which is valid for $SU(N)$. Using this identity, all internal color indices are completely eliminated. The algorithms used to implement this simplification are the same as those employed in the continuum [5]. The function 'colorExpand' takes an expression as an argument and performs color simplifications in it, without expanding subexpressions which do not contain color indices.

```
In[10]:=  (% /. openspur -> productGen /. spur -> traceGen) // colorExpand ;
```

```
In[11]:= % //. im^i_ :> -im^(i-2) ;
```

Another simplification can involve solving the delta function, which can be done as shown below.

```
In[12]:= Variables[%] // Sort
```

```
In[13]:= (#/(# /. delta[__]->1 ))& /@ List @@ %% // Union
```

```
In[14]:= Expand[%%%] /. f_ delta[a_ + b_. p[2]] :>
(replace[(f delta[a + b p[2]]),{p[2], -a/b}]) // canonical;
```

```
In[15]:= (#/(# /. delta[__]->1 ))& /@ List @@ % // Union
```

```
In[16]:= replace[Expand[%%], {q[2],q[1]}] /. delta[0] -> 1 ;
```

```
In[17]:= canonical[%];
```

```
In[18]:= Variables[%] // Sort
```

The following command was used:

```
replace[expresion, {a_, b_}]
```

to replace occurrences of a specific part of an expression (a) with another expression (b).
It is useful to observe the Dirac indices:

```
In[19]:= (#/(# /. dtrace[__]->1 /. propF[___] -> 1))& /@ List @@ %% // Union
```

At this point, we can replace the expression for all the tree-level Wilson propagators appearing in the diagrams. For completeness, we provide the propagators corresponding to the action in Eq. (7).

$$\begin{aligned}
&\text{Quark propagator :} && \frac{1}{i\not{q} + \frac{2r}{a}\sum_\mu \sin^2(aq_\mu/2) - m}, \quad \text{where :} \quad \not{q} = \frac{1}{a}\sum_\mu \gamma_\mu \sin(aq_\mu) \\
&\text{Gluon Propagator :} && \frac{1}{\hat{q}^2}\left(\delta_{\mu\nu} - (1-\alpha)\frac{\hat{q}_\mu\hat{q}_\nu}{\hat{q}^2}\right), \quad \text{where :} \quad \hat{q}_\mu = \frac{2}{a}\sin\frac{aq_\mu}{2}, \quad \hat{q}^2 = \sum_\mu \hat{q}^2_\mu \\
&\text{Ghost propagator :} && \frac{1}{\hat{q}^2} \\
&\text{Squark propagator :} && \frac{1}{\hat{q}^2 + m^2} \\
&\text{Gluino propagator :} && \frac{2}{i\not{q} + \frac{2r}{a}\sum_\mu \sin^2(aq_\mu/2)}
\end{aligned} \tag{25}$$

In the following equations we present the Mathematica expressions for these tree-level propagators.

```
In[20]:= %%%  /.
propF[a_, b_, c_, d_] :>
fhat[a,m] ((-m + 2 r s2sq[a]) dtrace[c, b]  - im dtrace[c, d, b] s2[2 a, d]) /.
prop[a_, b_, c_] :> hat2[a] (delm[b, c] - 4 beta hat2[a] s2[a, b] s2[a, c]) /.
propG[a_] :> hat2[a] /.
propA[a_] :> hat2[a, m] /.
propg[a_, b_, c_, d_] :> 2 fhat[a] ( 2 r s2sq[a] dtrace[c, b] - im dtrace[c, d, b] s2[2 a, d]) ;
```

As an exercise, the definitions of s2sq[a], hat2[a], hat2[a,m] can be derived by comparing the expressions above with Eq. (25). You can also see these definitions explicitly in Sections 3.4 and 3.5.

3.4. Gamma matrix algebra

In this section, we mainly describe how to simplify the Dirac matrices. We have the function 'traceDirac[a,b,...,c,d]' which is the trace of a product of Dirac matrices. The arguments of traceDirac can be: gamma5, nu[_] (an external Lorentz index taking values: 1,2,3,4), rho[_] (a dummy Lorentz index taking values: 1,2,3,4,...,nDim) or X, Y (unspecified Dirac matrices; each may appear at most once). For example: traceDirac[gamma5,nu[3],X,rho[2],gamma5,Y] = $\mathrm{Tr}(\gamma_5\,\gamma_{\nu_3}\,\mathrm{X}\,\gamma_{\rho_2}\,\gamma_5\,\mathrm{Y})$. There is another function which is called 'productDirac[i,a,b,...,c,d,j]' and have the same possible arguments as 'traceDirac[a,b,...,c,d]'. The function 'productDirac[i,a,b,...,c,d,j]' is a product of Dirac matrices. For example: productDirac[i,gamma5,nu[3],X,rho[2],gamma5,Y,j] = $(\gamma_5\,\gamma_{\nu_3}\,\mathrm{X}\,\gamma_{\rho_2}\,\gamma_5\,\mathrm{Y})_{ij}$.

```
In[21]:= Expand[%] //. dtrace[a__, i_]*dtrace[i_, b__] :> dtrace[a, b] ;

In[22]:= (#/(# /. dtrace[__]->1 ))& /@ List @@ % // Union

In[23]:= Expand[%%] /.
        {dtrace -> productDirac,
         gtrace -> traceDirac} ;
```

There are the following substitutions in order to simplify the gamma matrix algebra:

- productDiracRules4dim is a set of substitution rules which may be applied repeatedly in order to simplify products of Dirac matrices in 4 dimensions.
- productDiracRulesDdim is a set of substitution rules which may be applied repeatedly in order to simplify products of Dirac matrices in nDim = 4-2e dimensions.
- traceDiracRules4dim is a set of substitution rules which may be applied repeatedly in order to simplify traces of Dirac matrices in 4 dimensions.
- traceDiracRulesDdim is a set of substitution rules which may be applied repeatedly in order to simplify traces of Dirac matrices in nDim = 4-2e dimensions.

```
In[24]:= % //. productDiracRules4dim  //. traceDiracRules4dim;

In[25]:= (Expand[%] // simplifydelm) //reducerho // rorder ;
```

The commands 'reducerho' ('rorderall') and 'rorder' do the reordering, in the momenta, given the symmetry properties, and in the rho indices, respectively. The function 'reducerho' reduces rho indices after applying 'simplifydelm' function.

There is also the function 'reducegammaproductDirac' ('reducegammatraceDirac') which operates on a sum of terms, each of which is a product. For each term, which may contain up to one "productDirac" ("traceDirac"), it checks all pairs of consecutive indices of gamma matrices: If the rest of the term is symmetric under the interchange of a pair of two consecutive indices, then the term gets multiplied by a Kronecker delta of these two indices.

```
In[26]:= % // reducegammaproductDirac // reducegammatraceDirac;

In[27]:= lb[%]

In[28]:= Expand[%%] //. productDiracRules4dim  //. traceDiracRules4dim;

In[29]:= (Expand[%] // simplifydelm) //reducerho // rorder ;
```

```
In[30]:= Variables[%] // Sort
```

```
In[31]:= bl[%]
```

```
In[32]:= (Expand[%%%] // simplifydelm // reducerho) //.
productDiracRules4dim  //. traceDiracRules4dim // simplifydelm // reducerho;
```

```
In[33]:= {lb[%], Variables[%] // Sort}
```

The function 'lb' takes an expression as an argument and gives the length and the number of bytes of the expression. Similarly, the function 'bl' takes a list as an argument and gives the length and the number of bytes of the list.

In case you need to collect the expression with respect to specific variables, you can use the following:

```
In[34]:= collect[%%, {fhat[_,_], hat2[_]}]
```

The function 'collect' depends on the list given as the last argument. For example can collect all of the powers of the gauge coupling, g, of the lattice spacing, a, and of the number of colors of the theory, Nc. It also collects delm[ ], epsilon[ ], hat2[ ], etc. Note that the list of elements to collect may contain also Heads of variables as well as patterns, e.g. collect[...,g,s2[q[_],_],c2,...].

Note that Mathematica also has a command Collect which can be used as follows:

```
Collect[expression, {fhat[_,_], hat2[_]} ]
```

or even better:

```
Collect[expression, {fhat[_,_], hat2[_]}, Factor ]
```

3.5. Trigonometric manipulations

In this section, we give details on how the trigonometric simplifications are systematically applied to ensure all terms are in canonical form, making identifications and cancellations straightforward and automatic.

In our commands, we have the following trigonometric notation:

- s2[a_,b_] = Sin[a[b]/2]
- c2[a_,b_] = Cos[a[b]/2]
- s2sq[a_] = $\sum_r hoSin[a[rho]/2]^2$
- sisq[a_] = $\sum_r hoSin[a[rho]]^2$
- s2qu[a_] = $\sum_r hoSin[a[rho]/2]^4$
- phi2[a_,b_] = $\exp[I * a[b]/2]$, where I is the imaginary unit
- s2overs2[d_,a_,b_] = Sin[d*a[b]/2] / Sin[a[b]/2]
- s2sqdiff[a_,b_] = s2sq[a+b] - s2sq[b]
- hat2[p_] = 1/4 / Sum[Sin[p[i]/2]ˆ2, {i,4}]
- fhat[p_,mass_] = 1 / (Sum[Sin[p[i]]ˆ2,{i,4}] + (mass + 2r Sum[Sin[p[i]/2]ˆ2,{i,4}])ˆ2), where r is the Wilson coefficient
- fhat[p_] = fhat[p,0]
- m[p_] = mass + 2r Sum[Sin[p[i]/2]ˆ2,{i,4}], where 'mass' refers to the fermion mass. Note that the sign of the mass term differs from that in the quark propagator in Eq. (25).

For calculations in dimensional regularization, we use some of these symbols with different meanings. For example:

- s2[2 p_, b_] represents p_b,
- sisq[p_] represents p^2,

- hat2[p_] represents $\frac{1}{p^2}$.

When calculating Feynman diagrams, it is often convenient to perform a shift in the loop momentum, such as $p \to -p - q$, where p is the loop momentum and q is some external momentum. This shift is particularly useful when simplifying integrals over loop momenta or when aligning terms in the integrand to match a standard form. The integration measure d^4p remains invariant under such linear transformations, ensuring the validity of the shift. These shifts are most convenient in cases where the loop integrals become easier to evaluate after the redefinition of the loop variables, or when symmetries in the diagram allow for further simplifications. Such momentum shifts do not affect the physical content of the calculation, provided that regularization is handled consistently in the case of divergent integrals.

In the software package, there is a function called 'count[expr_,a_]', acting on a product 'expr' and it counts the factors in the expression which match the pattern 'a'. If a factor matches 'a^i', it will contribute 'i' to the total count. When 'expr' is not a product, then the result will be 1, if 'expr' matches 'a' as a whole, else it will be 0. An extension of this function is 'countlist[a_, b_]'; given a list 'b' of n doublets (where the first member of the doublet is a pattern, and the second member is a score): 'countlist[a_,b_]' provides a total score for product 'a', depending on the number of factors which match the patterns in 'b': countlist[a_, b_] = count[a,b[[1,1]]]*b[[1,2]] + ... + count[a,b[[n,1]]*b[[n,2]]]. If b is not a list, then countlist[a_,b_] = count[a,b]. Another useful function is 'hattosine', which rewrites hats in terms of sine and cosine functions.

It is convenient to define two lists to track the superficial degree of divergences (list1) and the overall powers (list2) of the lattice spacing:

```
In[35]:= list1 = {{m,1},{s2[_,_],1},{hat2[__],-2},{prop[__],-2},

{fhat[__],-2},{s2sq[_],2},{sisq[_],2},{s2qu[_],4}} ;

In[36]:= list2 = {{m,1},{s2[a_. q[1],_],1}} ;
```

The two lists can be used in combination with the function countlist, as follows:

```
In[37]:= {countlist[#,list1], countlist[#,list2]}& /@ List @@ Expand[%%%] // Union
```

The diagram in Fig. 2 contributes to the inverse fermion propagator, computed at one-loop order in perturbation theory. It is important to note that, for dimensional reasons, a global prefactor of $\frac{1}{a}$ multiplies our expressions for the inverse propagator. Consequently, the $\mathcal{O}(a^0)$ correction is obtained by including all terms up to $\mathcal{O}(a^1)$.

There are also some functions which can be applied to the whole of an expression, in order to simplify cosines; the Head of the expression must be Plus in order to maximize the instances in which such simplifications will take place. For example, 'replacec2All[expr]' acts on expression 'expr', replacing cosines by use of (for explanation reasons we don to not keep the second argument for the Lorentz index):

$$c2(x) = 1 - 2\,s2(x/2)^2$$
$$c2(x)^2 = 1 - s2(x)^2,$$

provided that arguments of trigonometric functions remain integer multiples of momenta. The function 'replacec2q[expr]' acts on expression 'expr', replacing cosine terms using the following rule (for explanatory purposes, we omit the second argument for the Lorentz index), keeping up to $O(q^2)$ terms, by use of:

$$c2(aq) = 1 - 2\,a^2\,s2(q)^2 + O(q^4)$$
$$c2(aq)^2 = 1 - a^2\,s2(q)^2 + O(q^4).$$

Terms of $O(q^4)$ are ignored. The function 'replacec2qexact[expr]' acts on expression 'expr', replacing cosines of (a multiple of) external momentum q by use of the exact formulae:

$$c2(aq)^2 = 1 - s2(aq)^2$$
$$c2(aq) = 1 - 2\,s2(aq/2)^2.$$

A similar function is used for the internal momenta p, called 'replacec2p[expr]'.

In all these cases, implicit summation over the directions of the arguments of the cosines is considered. Furthermore, the Head of the expression 'expr' must be Plus to ensure maximum efficiency.

The function 'makes2sq' takes a typical integrand: prefactor * (sum of terms) and substitutes, wherever legitimate, s2[a_,rho[_]]^2 by s2sq[a], s2[2a_,rho[_]]^2 by sisq[a] and s2[a_,rho[_]]^4 by s2qu[a]. This reduces the number of rho indices, thus moderating the increase in the size of the expression once the rho's are made to be different. The propagator momenta must be p[1],p[2],p[3],p[1]+p[2],p[1]+p[3],p[3]-p[2] in order for this command to make the maximum amount of substitutions. To use the aforementioned function, one simply says (before rendering rho's different):

```
In[38]:= makes2sq[%%%%];

In[39]:= reducerho[%];

In[40]:= rorder[%];
```

The function can be used also for the external momenta, in the following naive way (given that p[2] does not appearing in the expression):

```
makes2sq[Expand[%%] /. q[1]->p[2]] /. p[2] -> q[1] ;
Expand[%] // simplifydelm // reducerho // rorder ;
```

3.6. Extraction of the dependence on external momenta

Determining the precise analytic momentum dependence of an n-point function as $a \to 0$ is one of the most challenging tasks, both conceptually and algorithmically.

The above simplifications are followed by the extraction of all forms of functional dependence on the external momentum q (logarithmically divergent, Lorentz non-invariant, polynomial terms) and the lattice spacing (terms of order a^0, a^1, a^2, and possibly $\ln a$).

As a first task we want to reduce the number of infrared divergent integrals to a minimal set. To do this, we use two kinds of subtractions among the propagators, using the simple equalities:

$$\frac{1}{\tilde{k}^2} = \frac{1}{\hat{k}^2} + \left\{ \frac{4\sum_\mu \sin^4(k_\mu/2) - 4\left(\sum_\mu \sin^2(k_\mu/2)\right)^2}{\tilde{k}^2\,\hat{k}^2} \right\} \tag{26}$$

$$\begin{aligned} D(k) &= D_{plaq}(k) + \Big\{ D(k) - D_{plaq}(k) \Big\} \\ &= D_{plaq}(k) + D_{plaq}(k)\Big\{ D^{-1}_{plaq}(k) - D^{-1}(k) \Big\} D(k) \end{aligned} \tag{27}$$

where k stands for p or $p + a\,q$, and p (q) is the loop (external) momentum.

```
 prop[p[i_],b_,c_]:> dprop[p[i],b,c] + delm[b,c] hat2[p[i]] ;
```

The denominator of the fermion propagator, $\tilde{k}^2$, is defined as

$$\tilde{k}^2 = \sum_\mu \sin^2(k_\mu) + \Big(m_f + \frac{r}{2}\hat{k}^2\Big)^2, \quad \hat{k}^2 = 4\sum_\mu \sin^2(\frac{k_\mu}{2}) \tag{28}$$

For the present work, one sets $m_f = 0$ and $r = 1$, as used in Eq. (26); D is the 4×4 Symanzik gluon propagator; the expression for the matrix $\Big(D^{-1}_{plaq}(k) - D^{-1}(k)\Big)$, which is $\mathcal{O}(k^4)$, is independent of the gauge parameter, β, and it can be easily obtained in closed form. Moreover, we have

$$\big(D_{plaq}(k)\big)_{\mu\nu} = \frac{\delta_{\mu\nu}}{\hat{k}^2} - \beta\frac{\hat{k}_\mu\,\hat{k}_\nu}{(\hat{k}^2)^2} \tag{29}$$

Terms in curly brackets of Eqs. (26) and (27) are less IR divergent than their unsubtracted counterparts, by two powers in the momentum. These subtractions are performed iteratively until all primitively divergent integrals (initially depending on the fermion and the Symanzik propagator) are expressed in terms of the Wilson gluon propagator.

Having reduced the number of distinct divergent integrals down to a minimum, the most laborious task is the computation of these integrals, which is performed in a noninteger number of dimensions $D > 4$. Ultraviolet divergences are explicitly isolated à la Zimmermann and evaluated as in the continuum. The remainders are D-dimensional, parameter-free, zero external momentum lattice integrals which can be recast in terms of Bessel functions, and finally expressed as sums of a pole part plus numerical constants. We analytically evaluate an extensive basis of superficially divergent loop integrals, listed in Eqs. (C1) - (C10) of Appendix B; a few of these were calculated in Ref. [30]. The integrals of Eqs. (C1), (C2), (C3), are the most demanding ones in the list; they must be evaluated to two further orders in a, beyond the order at which an IR divergence initially sets in. As a consequence, their evaluation requires going to $D > 6$ dimensions. Fortunately, they are a sufficient basis for all massless integrals which can appear in any

$\mathcal{O}(a^2)$ one-loop calculation; that is, any such computation can be recast in terms of (C1), (C2), (C3), plus other integrals which are more readily handled. A correct way to evaluate (C1), (C2), (C3) has not been presented previously in the literature, despite their central role in $\mathcal{O}(a^2)$ calculations, and this has prevented one-loop computations to $\mathcal{O}(a^2)$ thus far. The calculation of such an integral is given in detail in the next section.

Terms which are IR convergent can be treated by Taylor expansion in aq to the desired order. Alternatively, the extraction of the aq dependence may be performed using iteratively subtractions of the form

$$f(p + a\,q) = f(p) + \Big[f(p + a\,q) - f(p)\Big] \tag{30}$$

There follows some exact substitutions where we make a different type of subtraction, adding and subtracting the term $\frac{1}{\hat{k}^2+m^2}$.

```
If[countlist[#, list1] <= -3,
  (# /. c_ fhat[a_, b_]^(i_.):> c fhat[a, b]^(i - 1) hat2[a, b] +
      c fhat[a, b]^i hat2[a, b] * (4 s2qu[a] - 4 s2sq[a]^2 + 4 r b s2sq[a])
  ), #] & /@ Expand[%] ;
```

Similarly, with the term $\frac{1}{\hat{k}^2}$:

```
If[countlist[#, list1] <= -3,
  (# /. c_ fhat[a_]^(i_.):> c fhat[a]^(i - 1) hat2[a] +
      c fhat[a]^i hat2[a] * (4 s2qu[a] - 4 s2sq[a]^2)
  ), #] & /@ Expand[%] ;
```

Eq. 30 leads to exact relations such as the following ones

$$\begin{aligned}\frac{1}{\widetilde{p+a\,q}^{\,2}} &= \frac{1}{\tilde{p}^2} - \frac{\sum_\mu \sin(2p_\mu + a\,q_\mu)\sin(a\,q_\mu)}{\widetilde{p+a\,q}^{\,2}\,\tilde{p}^2} \\ &- \frac{\sum_\mu \sin(p_\mu + \frac{a\,q_\mu}{2})\sin(\frac{a\,q_\mu}{2})\left(\hat{p}^2 + \widehat{p+a\,q}^{\,2}\right)}{\widetilde{p+a\,q}^{\,2}\,\tilde{p}^2}\end{aligned} \tag{31}$$

$$\frac{1}{\widehat{p+a\,q}^{\,2}} = \frac{1}{\hat{p}^2} - \frac{4\sum_\mu \sin(p_\mu + \frac{a\,q_\mu}{2})\sin(\frac{a\,q_\mu}{2})}{\widehat{p+a\,q}^{\,2}\,\hat{p}^2} \tag{32}$$

In these relations the exact $a\,q$ dependence of the remainders is under full control; this type of subtraction is especially useful when applied to the Symanzik propagator.

When applicable, we can use expansions of the cosine terms. Furthermore, we may also expand certain sines avoiding IR divergences. Below is the commands for this process:

```
ExpandAll[%] //. s2[a_, b_]^i_. c2[a_, b_]^j_. :> s2[a, b]^(i-1) c2[a, b]^(j-1) * (1/2) * s2[2a, b] ;

Expand[%] /. c_ sisq[q[1] + p[1]]^(i_.) -> c * sisq[q[1] + p[1]]^(i-1) *
s2[2q[1] + 2p[1], rho[21]]^2 ;

Expand[% //. s2[a_ + b_, c_] :> s2[a, c] * c2[b, c] + c2[a, c] * s2[b, c]] ;

Expand[% //. c2[a_ + b_, c_] :> c2[a, c] * c2[b, c] - s2[a, c] * s2[b, c]] ;

Expand[%] // reducerho // rorder ;
```

All cosine functions that depend solely on q can be replaced exactly with equivalent expressions involving sine functions. This can be achieved by repeatedly using the ‘replacec2exact’ command.

```
replacec2exact[replacec2exact[Expand[%]]] ;
```

To prevent an explosion in the number of terms, one can omit the terms that are convergent and of higher order than the required power of the lattice spacing.

```
If[(countlist[#, list2] > 0) && (countlist[#, list1] > -3), 0, #] & /@ Expand[%] ;
```

Furthermore, substitutions for the cosine of the loop momentum p can be made to express the integrals in a form suitable for extracting the divergent part of the expressions.

```
If[countlist[#, list1] <= -3, replacec2p[#], #] & /@ Expand[%] ;
```

These commands can be followed by simplifications such as:

```
makes2sq[Expand[%] /. q[1] -> p[2]] /. p[2] -> q[1];
Expand[%] // simplifydelm // reducerho // rorder;
% //. s2sq[a_]^i_. hat2[a_]^j_. :> (1/4) * s2sq[a]^(i-1) * hat2[a]^(j - 1) ;
{Length[%], Variables[%]}
```

After the divergent terms have been brought to the appropriate form, the expression can be separated into two parts: the divergent and the convergent terms. This separation is achieved by evaluating the superficial divergence number.

```
(* Extract the divergent terms (superficial divergence number <= -4) *)
divergent = If[countlist[#, list1] <= -3, #, 0] & /@ Expand[%%] ;
{Length[%], Variables[%]}

(* Extract the convergent terms (superficial divergence number > -4) *)
convergent = If[countlist[#, list1] > -3, #, 0] & /@ Expand[%%%%] ;
{Length[%], Variables[%]}
```

For the convenience of the user, we have created a new command, 'subtractionIRdiv', to automate this procedure. This command is specifically designed for use with one-loop expressions. In particular, it simplifies one-loop expressions by isolating infrared (IR) divergent terms. It takes as input the full expression and the power of the powers of the lattice spacing (n) that we want to achieve. The function processes the input to separate and tag divergent terms with "div", while finite (convergent) terms remain untagged.

The command works only for mass independent renormalization schemes, so that $m = 0$. This simplifies the algebraic expressions, but at the same time requires special treatment when it comes to IR singularities.

```
In[41]:= Expand[%] /. m -> 0 /. hat2[a_,0] :> hat2[a] /. fhat[b_,0] -> fhat[b] ;

In[42]:= subtractionIRdiv[Expand[%],1] ;

In[43]:= convergent = % /. div -> 0 ;

In[44]:= divergent = %% - % ;
```

The divergent integrals can be replaced by the expressions included in this software by following the commands shown below:

```
In[45]:= {countlist[#, list1], countlist[#, list2]} & /@ List @@ divergent // Union

In[46]:= {Length[%%], Variables[%%]}

In[47]:= (#/(# /. hat2[__]->1 /. fhat[__] -> 1 /. s2[__]->1 /. c2[__]->1 /.
sisq[__] ->1 /. s2sq[__] -> 1 /. s2qu[__] -> 1 ))& /@ List @@
ExpandAll[divergent] // (reducerho /@ #)& // (rorder /@ #)& // Union

In[48]:= IntegralsResults

In[49]:=  DIVpart = divergent /. IntegralsResults /.  Log[q1^2] -> Log[q[1]^2];

In[50]:= {Length[%], Variables[%]}

In[51]:= Save["MyFirstDiagram.m", DIVpart]
```

For the convergent part, we can manipulate it using a naive Taylor expansion. Additionally, for these convergent terms, we can proceed to render rho's different between the external momentum q[1] and the loop momentum p[1]. The function 'matchindices' takes an integrand with propagators independent of external momenta, and matches the directions of sines, so as to lead to an even integrand under p[i] $\rightarrow$ -p[i]. Aside from a prefactor, the numerator is a sum of terms, where each term may be a function of p's times a function of external momenta. There is also a function, which is called 'matchsemiperiodic' and takes an integrand over p[1] whose denominators are invariant under: p[1]$_\mu$ $\rightarrow$ p[1]$_\mu$ + pi (where μ is any particular direction), and whose numerators have p-dependent parts made up exclusively of s2[2p[1] + a_., b_] and/or c2[2p[1] + a_., b_] and matches the directions of sines and/or cosines, so as to lead to an even integrand under p[1]$_\mu$ $\rightarrow$ p[1]$_\mu$ + pi.

In some cases, for the convergent terms, we can set q[1]=0, or we can Taylor expand the expression to the appropriate powers. In the first case:

```
In[52]:= {countlist[#, list1], countlist[#, list2]} & /@ List @@ convergent // Union
```

```
In[53]:= {Length[convergent], Variables[convergent]}
```

In order to carefully extract the external momentum, we must meticulously apply the following procedure. However, in many cases, we can set q to zero.

```
replace[ExpandAll[convergent], {q[1],0}] ;
{Length[%], Variables[%]}
```

```
In[54]:= {count[#,s2sq[q[1]+p[1]]],count[#,s2qu[q[1]+p[1]]]}& /@ List @@ convergent // Union
```

```
In[55]:= convergent /. s2sq[a_+b_]^i_. :> s2sq[a+b]^(i-1)*s2[a+b,rho[20]]^2 ;
```

```
In[56]:= % /. s2sq[a_+b_]^i_. :> s2sq[a+b]^(i-1)*s2[a+b,rho[21]]^2 ;
```

```
In[57]:= % /. s2qu[a_+b_]^i_. :> s2qu[a+b]^(i-1)*s2[a+b,rho[26]]^4 ;
```

```
In[58]:= Select[Variables[%],FreeQ[#,dtrace]&]
```

```
In[59]:= Module[{expr = ExpandAll[%%], exprtemp = 0, sum = 0, i=1},
       While[i<=Length[expr],
             exprtemp = Take[expr,{i,Min[i+99,Length[expr]]}] ;
     exprtemp = exprtemp //. s2[a_+ b_,c_] :> s2[a,c] c2[b,c] + c2[a,c] s2[b,c] // Expand ;
     exprtemp = exprtemp //. c2[a_ + b_,c_]:> c2[a,c] c2[b,c] - s2[a,c] s2[b,c] // Expand ;
             exprtemp = Select[exprtemp,(countlist[#,list2] <= 1)&];
             exprtemp = If[countlist[#,list2]>=1,simplifydelm[# /.c2[a_.q[1],b_]:>delm[b,b]],#]& /@ expr
             exprtemp = FixedPoint[replacec2exact, exprtemp];
             sum = sum + Select[exprtemp,(countlist[#,list2] <= 1)&];
             Print["{i=",i,", length[sum]=", Length[sum],"}"];
             i = i + 100]; sum];
```

```
In[60]:= {Length[%], Variables[%]}
```

In the convergent terms, when the appropriate powers of the external momentum are present in the remaining expression, we can set q to zero:

```
In[61]:= If[countlist[#,list2] == 1 , canonical[#/. {hat2[a_+b_. q[1],c___]:>hat2[a],

 hat2[a_,c__]:>hat2[a], fhat[a_+b_. q[1],c__]:>fhat[a], fhat[a_,c__]:>fhat[a],

 s2[a_+b_. q[1],c_]:>s2[a,c], s2qu[a_+q[1]]:>s2qu[a], s2sq[a_+q[1]]:>s2sq[a]}], #]& /@ %% ;
```

```
In[62]:= {Length[%], Variables[%]}
```

```
In[64]:= rorder[reducerho[%%]];
```

```
In[65]:= Length[%]
```

For completeness, we provide the expressions for the massive propagators:

```
In[66]:= If[countlist[#,list2] < 1, # /. fhat[a_+q[1],m]^i_. :>
fhat[a+q[1],m]^(i-1) (fhat[a]-
(s2[4a+2q[1],rho[20]] s2[2q[1],rho[20]] + m^2 + 4 m r s2sq[a+q[1]]+
  4r^2 s2[2a+q[1],rho[20]] s2[q[1],rho[20]] *(s2sq[a]+s2sq[a+q[1]]) )*fhat[a]*fhat[a+q[1],m]) //
Expand // reducerho, #]& /@ Expand[%%%];

In[67]:= {ByteCount[%],Length[%]}

In[68]:= If[countlist[#,list2] < 1, # /. hat2[a_+q[1],m]^i_. :>
hat2[a+q[1],m]^(i-1) (hat2[a]-
(4s2[q[1],rho[20]]*s2[2a+q[1],rho[20]] + m^2 )*hat2[a]*hat2[a+q[1],m]) //
Expand // reducerho, #]& /@ %% ;

In[69]:= {ByteCount[%],Length[%]}

In[70]:= If[countlist[#,list2] < 1, # /. fhat[a_+q[1]]^i_. :>
fhat[a+q[1]]^(i-1) (fhat[a]-
(s2[4a+2q[1],rho[20]] s2[2q[1],rho[20]] +
  4r^2 s2[2a+q[1],rho[20]] s2[q[1],rho[20]] *(s2sq[a]+s2sq[a+q[1]]) )*fhat[a]*fhat[a+q[1]]) //
Expand // reducerho, #]& /@ %%;

In[71]:= {ByteCount[%],Length[%]}

In[72]:= If[countlist[#,list2] < 1, # /. hat2[a_+q[1]]^i_. :>
hat2[a+q[1]]^(i-1) (hat2[a]-
(4s2[q[1],rho[20]]*s2[2a+q[1],rho[20]])*hat2[a]*hat2[a+q[1]]) //
Expand // reducerho, #]& /@ %% ;

In[73]:= {ByteCount[%],Length[%]}

In[74]:= If[countlist[#,list2] < 1, # /. hat2[a_+q[1]]^i_. :>
hat2[a+q[1]]^(i-1) (hat2[a]-
(4s2[q[1],rho[20]]*s2[2a+q[1],rho[20]])*hat2[a]*hat2[a+q[1]]) //
Expand // reducerho, #]& /@ %% ;

In[75]:= {ByteCount[%],Length[%]}

In[76]:= Variables[%%] // Sort

In[77]:= If[countlist[#,list2] == 1 , canonical[#/.
{hat2[a_+b_. q[1],c___]:>hat2[a],
 hat2[a_,c__]:>hat2[a], fhat[a_+b_. q[1],c__]:>fhat[a],
 fhat[a_,c__]:>fhat[a], s2[a_+b_. q[1],c_]:>s2[a,c],
 hat2[a_+b_. q[1]]:>hat2[a],fhat[a_+b_. q[1]]:>fhat[a],
 s2qu[a_+q[1]]:>s2qu[a], s2sq[a_+q[1]]:>s2sq[a]}], #]& /@ %%% ;

In[78]:= {Length[%], Variables[%]}
```

At any time, we can verify the term count in our expression.

```
In[79]:= {countlist[#, list1], countlist[#, list2]} & /@ List @@ %% // Union
```

Since q is now separate from p, we can use the function 'matchindices'.

```
In[80]:= matchindices[%%];

In[81]:= FreeQ[%, matchindices]

In[82]:= simplifydelm[%%] // reducerho // rorder ;

In[83]:= {Length[%], Variables[%]}
```

Another command, 'makeindependent', can be used to simplify symbolic expressions by isolating terms, identifying common elements, and applying targeted substitutions to ensure that q and p have different Lorentz indices. This command is included in the module, where it can also be combined with some previous commands to further simplify the expression. This task is needed in order to isolate the part to be integrated and to make it independent of the external momentum components.

```
In[84]:= Module[{expr = %%, exprtemp = 0, sum = 0, i=1},
       While[i<=Length[expr],
    temp = Take[expr,{i,Min[i+99,Length[expr]]}];
    temp = ((Expand[makeindependent[temp]] // simplifydelm // Expand // reducerho // rorder ));
    Print["makeindependent done!"];
    temp = temp //. s2[a_,b_]^ii_. c2[a_,b_]^j_. :> s2[a,b]^(ii-1) c2[a,b]^(j-1) 1/2 s2[2a,b];
    temp = makes2sq[temp]//. s2sq[a_]^ii_. hat2[a_]^j_. :> 1/4 s2sq[a]^(ii-1) hat2[a]^(j-1) ;
    temp = rorder[reducerho[makes2sq[temp/.q[1]->p[2]]  /.p[2]->q[1]]] ;
    temp = temp // reducerho // rorder ;
    sum = sum + temp;
    Print["{i=",i,", length[sum]=", Length[sum],"}"];
    i = i + 100];
        sum  ];

In[85]:= Variables[%]
```

Some additional simplifications include:

```
In[86]:= %% /. s2[b_. q[1],a_] :> S2[2q[1],a] b/2  // Expand;

In[87]:= % /. S2 :> s2 ;

In[88]:= % //. im^i_ :> -im^(i-2) ;

In[89]:= {Length[%],Variables[%]}

In[90]:= %% //. productDiracRules4dim //. traceDiracRules4dim;

In[91]:= (Expand[%] // simplifydelm) //reducerho // rorder ;

In[92]:= % // reducegammaproductDirac // reducegammatraceDirac;

In[93]:= lb[%]

In[94]:= Expand[%%] //. productDiracRules4dim //. traceDiracRules4dim;

In[95]:= (Expand[%] // simplifydelm) //reducerho // rorder ;

In[96]:= Variables[%] // Sort

In[97]:= bl[%]

In[98]:= (Expand[%%%] // simplifydelm // reducerho) //.
productDiracRules4dim //. traceDiracRules4dim // simplifydelm // reducerho;

In[99]:= {lb[%], Variables[%] // Sort}
```

At this point, we can prepare the expression for numerical integration by extracting the prefactors and saving them separately. These prefactors consist of the variables that multiply the integrand.

```
In[100]:= out100 = (ExpandAll[%%] // reducerho // rorder // rorderall) /.
r -> 1 /. nDim -> 4;

In[101]:= Select[Variables[out100],(FreeQ[#,p] && FreeQ[#,cprop0] && FreeQ[#,cprop1])&]

In[102]:= ((#/(# /. Table[%[[i]] -> 1,{i,Length[%]}]))& /@ (List @@ out100)) // Union

In[103]:= varlist = Table[var[i] -> %[[i]],{i,Length[%]}]

In[104]:= (* Check *)

In[105]:= Reverse[Table[dummy %%%[[i]]-> var[i],{i,Length[%%%]}]];
```

```
In[106]:= ((dummy #)& /@ out100) /. %;

In[107]:= SameQ[% /. varlist, out100]

In[108]:= intexpr = %%;

In[109]:= % // reducerho // rorder // rorderall;

In[110]:= Variables[%]

In[111]:= Length[%%]

In[112]:= intexpr = Expand[%%%];
```

We perform the explicit summations over rho indices:

```
In[113]:= FreeQ[intexpr,rho[#]]& /@ {1,2,3,4,5,6,7,8,9,10}

In[114]:= (If[FreeQ[#,rho[1]],#,4 #]& /@ (%% + dummy)) /. dummy -> 0 ;

In[115]:= Length[%]
```

For completeness, we also provide the additional summations over rho[i], where the indices for $i > 5$ are treated the same as for $i = 5$.

```
(If[FreeQ[#,rho[2]],#, (# /. {rho[2] -> rho[1]}) + 3 #]& /@ (%% + dummy)) /. dummy -> 0 ;
Length[%]
(If[FreeQ[#,rho[3]],#, (# /. {rho[3] -> rho[1]}) + (# /. {rho[3] -> rho[2]}) + 2 #]& /@
(%% + dummy)) /. dummy -> 0 ;
Length[%]
(If[FreeQ[#,rho[4]],#, (# /. {rho[4] -> rho[1]}) + (# /. {rho[4] -> rho[2]}) +
(# /. {rho[4] -> rho[3]}) + #]& /@ (%% + dummy)) /. dummy -> 0 ;
Length[%]
(If[FreeQ[#,rho[5]],#, (# /. {rho[5] -> rho[1]}) + (# /. {rho[5] -> rho[2]}) +
(# /. {rho[5] -> rho[3]}) + (# /. {rho[5] -> rho[4]}) ]& /@ (%% + dummy)) /. dummy -> 0 ;
Length[%]
```

3.7. Numerical integration over loop momenta

Having reached this point, the only remaining task is the numerical evaluation of loop integrals that do not depend on external momenta.

After performing the summation over rho[i], we use the following commands to reduce further the number of terms. For convenience, these commands are organized into a module, where we save the variables and the integrand.

```
In[116]:= Module[{expr = Expand[%%%], exprtemp = 0, sum = 0, i=1},
       While[i<=Length[expr],
    exprtemp = Take[expr,{i,Min[i+1999,Length[expr]]}];
    exprtemp = Expand[exprtemp] // reducerho // rorder ;
    sum = sum + exprtemp;
    Print["{i=",i,", length[sum]=", Length[sum],"}"];
    i = i + 2000];
       sum  ];

In[117]:= intexpr = % ;

In[118]:= Save["MyFirstDiagram.m",varlist, intexpr]
```

The required numerical integrations of the algebraic expressions for the loop integrands are performed by highly optimized Fortran programs; these are generated by our Mathematica ‘integrator1’ routine. Each integral is expressed as a sum over the discrete Brillouin zone of finite lattices, with varying size L ($4^4 \leq L^4 \leq 128^4$ for one-loop integrals and $4^4 \leq L^4 \leq 32^4$ for two-loop integrals), and evaluated for given values of the Symanzik coefficients of the gluon action.

```
In[119]:= Module[{i=1,temp,filename,expr=intexpr},
While[i<=Length[expr], Print[i]; temp=Take[expr,{i,Min[i+39999,Length[expr]]}];
      filename = StringJoin["fortranfiles/namefile_part",ToString[(i+39999)/40000],".f"];
      integrator1[temp,1,filename];
      Close[filename]; i = i + 40000]]
```

After generating the Fortran files, they can be executed, and the output will provide the integral values for each lattice size. These values must then be extrapolated to an infinite lattice and multiplied by the appropriate variable.

```
>> gfortran @ofastG namefile_part1.f -o namefile_part1.x
>> ./namefile_part1.x < input1loop > ./outfiles/namefile_part1.out
```

3.8. Extrapolation to infinite lattice size

The final part of the evaluation of Feynman diagrams is the extrapolation of the numerical results calculated in the previous sections (which apply to finite lattice size) to an infinite lattice size. This process introduces a systematic error, which is reliably estimated using a sophisticated inference technique. Drawing from our previous experience, we aim for a fractional error smaller than 10^{-8} for one-loop quantities and smaller than 10^{-4} for two-loop quantities.

For the extrapolation, we create a command 'extrapolate', which uses the variables that multiply each integral and the output file containing the results from the executed Fortran files.

When the files *namefile_parti.out* are ready, you can extrapolate each integral as follows (in our case $i = 1$):

```
In[120]:= extrapolate[Table[StringJoin["outfiles/namefile_part",ToString[i],".out"],{i,1}],varlist]

In[121]:= % /. result[{{{a__},{b__}}}] :> Around[b] ;

In[122]:= Variables[%] // Sort

In[123]:= %% /. Around[a_,b_] :> a // Expand ;

In[124]:= Variables[%] // Sort

In[125]:= CONVpart = %% ;

In[126]:= Save["MyFirstDiagram.m", CONVpart]
```

Lastly, we add the divergent parts to the extrapolated convergent results to obtain the total expression of the diagram. We also include the combinatorial factor for each diagram (calculated by hand), multiplying it with the appropriate result.

```
In[127]:= TotalExpression = Combinatorial * (DIVpart + CONVpart) /. Combinatorial -> 1;

In[128]:=  Save["MyFirstDiagram.m", TotalExpression]

In[129]:= Quit
```

Appendix A: Evaluation of Continuum Integrals

There are three main commands:'reducediamond', 'ChetyrkinFormula' and 'FourierTransformIntegral', which can be use to calculate loop integrals in the continuum. We use dimensional regularization in order to calculate the integrals in the continuum, in $D = 4 - 2\epsilon$ dimensions.

Firstly, the 'reducediamond' is designed to transform diamond-topology to simpler topologies. Its primary purpose is to process "diamond"-type diagrams, which commonly appear in higher loop calculations.

```
reducediamond[expression]
```

This function makes use of the recursion relation, given in Chetyrkin et al., N.P. B.192, 159, for scalar two-loop integrals of the "diamond" type, i.e., with propagators of the form:

$$(\widehat{p_1}^2)^a(-\widehat{p_1+q_1}^2)^b(-\widehat{p_1+p_2}^2)^l(\widehat{p_1}^2)^m(-\widehat{p_2+q_1}^2)^n \tag{A1}$$

The function can work also for tensor two-loop integrals, using a generalized recursion relation given in arXiv: 2010.02062 (Phys. Rev. D 103, 014515 (2021)).

Note that before applying 'reducediamond' in an expression, the latter must be expanded, or the part of the expression that depends on both denominators and p_1 must be collected. Below are the commands that you can use to evaluate such two-loop integrals:

```
In[1]:= << /home/username/FEDILA-main/input.m

In[2]:= = hat2[p[1]] hat2[-p[1]+q[1]]^2 hat2[-p[1]+p[2]]^2 hat2[p[2]] hat2[-p[2]+q[1]]

In[3]:= makedenoms[%]

In[4]:= reducediamond[%]

In[5]:= unmakedenoms[%]
```

Secondly, the 'ChetyrkinFormula' function automates the application of a standard one-loop formula for Feynman integrals, a cornerstone in the renormalization of operators in QFT, given in Chetyrkin et al., N.P. B.192, 159. It is particularly suited for dimensional regularization techniques and higher-loop corrections, enabling symbolic and numerical evaluations of Feynman integrals. This function performs integration over the momentum p[i], using the one-loop formula.

```
ChetyrkinFormula[expression, p[i_]]
```

For example:

```
In[6]:= hat2[p[1]] hat2[p[1]+q[1]]^2 s2[2 p[1],rho[1]] hat2[p[2]] hat2[p[2]+q[1]]

In[7]:= ChetyrkinFormula[%, p[1]]

In[8]:= ChetyrkinFormula[%, p[2]]

In[9]:= % /. grule
```

Lastly, the 'FourierTransformIntegrals' function handles the symbolic evaluation of Fourier transform integrals, a crucial task in converting position-space quantities into momentum-space representations in QFT. In particular, it calculates one-loop integrals of p[1] having the the form:

$$\exp[i\,z\,p]/(p^2)^a(p_{\mu_1}\cdots p_{\mu_n}) \tag{A2}$$

```
FourierTransformIntegral[expression]

In[10]:= exp[im z p[1]] hat2[p[1]] s2[2p[1],rho[1]]

In[11]:= FourierTransformIntegral[%] /.  oneminusx -> (1-x) /. hold[a_] :> a ;

In[12]:= FullSimplify[%]

In[13]:= Quit
```

Appendix B: Evaluation of a primitively divergent integral

Divergent integrals which appear in calculations up to $\mathcal{O}(a^1)$ may be evaluated using the standard procedure of Kawai et al. [31], in which one subtracts and adds to the original integrand its naive Taylor expansion, to the appropriate order with respect to a, in $D \to 4^+$ dimensions: The subtracted integrand, being UV convergent, is calculated in the continuum limit $a \to 0$, using the methods of Ref. [32], while the Taylor expansion terms are recast in terms of Bessel functions and are evaluated in the limit $\epsilon \to 0$ ($\epsilon \equiv (4-D)/2$).

In contrast to the above, some of the integrals in the section, given that they must be evaluated to $\mathcal{O}(a^2)$, have Taylor expansions which remain IR divergent all the way up to $D \leq 6$ dimensions. A related difficulty regards Kawai's procedure: Subtracting from the original integral its Taylor expansion in D-dimensions to the appropriate order, the UV-convergent subtracted expression at which one arrives can no longer be evaluated in the continuum limit by naively setting $a \to 0$, because there will be $\mathcal{O}(a^2)$ corrections which must not be neglected. These novel difficulties plague integrals C1, C2, C3, of Appendix B. Using a combination of momentum shifts, integration by parts and trigonometric identities, one may express C2 and C3 in terms of C1 and other less divergent integrals. Thus, it suffices to address the evaluation of C1

$$A1(p) = \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{1}{\hat{k}^2\, \widehat{k+a\,p}^{\,2}} \tag{B1}$$

This is a prototype case of an integral which is IR divergent in $D \leq 6$ dimensions; in fact, all other integrals encountered in the present calculation may be expressed in terms of $A1(p)$ plus other integrals which are IR convergent at $D > 4$ (and are thus amenable to a more standard treatment).

First we split the original integrand I into two parts

$$I \equiv \frac{1}{\hat{k}^2\, \widehat{k+a\,p}^{\,2}} = I_1 + I_2 \tag{B2}$$

where I_2 is obtained from I by a series expansion, with respect to the arguments of all trigonometric functions, to subleading order; I_1 is simply the remainder $I - I_2$

$$\begin{aligned} I_1 \;=\;& \frac{k^2 - \frac{k^4}{12} - \hat{k}^2}{k^2\, \hat{k}^2\, \widehat{k+a\,p}^{\,2}} + \frac{k^4 \left(k^2 - \hat{k}^2\right)}{12\, \left(k^2\right)^2\, \hat{k}^2\, \widehat{k+a\,p}^{\,2}} + \frac{k^4 \left((k+a\,p)^2 - \widehat{k+a\,p}^{\,2}\right)}{12\, \left(k^2\right)^2\, (k+a\,p)^2\, \widehat{k+a\,p}^{\,2}} \\ &+ \frac{(k+a\,p)^2 - \frac{(k+a\,p)^4}{12} - \widehat{k+a\,p}^{\,2}}{k^2\, (k+a\,p)^2\, \widehat{k+a\,p}^{\,2}} + \frac{(k+a\,p)^4 \left((k+a\,p)^2 - \widehat{k+a\,p}^{\,2}\right)}{12\, k^2\, \left((k+a\,p)^2\right)^2\, \widehat{k+a\,p}^{\,2}} \end{aligned} \tag{B3}$$

$$I_2 \;=\; \frac{1}{k^2\, (k+a\,p)^2} + \left[\frac{(k+a\,p)^4}{12\, k^2\, \left((k+a\,p)^2\right)^2} + \frac{k^4}{12\, \left(k^2\right)^2\, (k+a\,p)^2}\right] \tag{B4}$$

($q^4 \equiv \sum_\mu q_\mu^4$). I_2 is free of trigonometric functions, while I_1 is naively Taylor expandable to $\mathcal{O}(a^2)$; its integral equals

$$\int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} I_1 = 0.004210419649(1) + a^2\, p^2\, 0.0002770631001(3) + \mathcal{O}(a^4, a^4 \ln a) \tag{B5}$$

The errors appearing in the above equation come from extrapolations to infinite lattice size.

To evaluate the integral of I_2 we split the hypercubic integration region into a sphere of arbitrary radius μ about the origin ($\mu \leq \pi$) plus the rest

$$\int_{-\pi}^{\pi} = \int_{|k| \leq \mu} + \left(\int_{-\pi}^{\pi} - \int_{|k| \leq \mu}\right) \tag{B6}$$

The integral outside the sphere is free of IR divergences and is thus Taylor expandable to any order, giving[3] (for $\mu = 3.14155$)

$$\left(\int_{-\pi}^{\pi} - \int_{|k| \leq \mu}\right) \frac{d^4k}{(2\pi)^4} I_2 = 6.42919(3)\, 10^{-3} + a^2\, p^2\, 6.2034(1)\, 10^{-5} + \mathcal{O}(a^4) \tag{B7}$$

We are now left with the integral of I_2 over a sphere. The most infrared divergent part of I_2 is $1/(k^2\,(k+a\,p)^2)$, with IR degree of divergence -4, and can be integrated *exactly*, giving

$$\int_{|k| \leq \mu} \frac{d^4k}{(2\pi)^4} \frac{1}{k^2\, (k+a\,p)^2} = \frac{1}{16\pi^2} \left(1 - \ln(\frac{a^2\, p^2}{\mu^2})\right) \tag{B8}$$

[3] Due to its peculiar domain, this integral has been evaluated by a Monte Carlo routine, rather than as a sum over lattice points. The errors in Eq. (B7) are thus Monte Carlo errors.

The remaining two terms comprising I_2 have IR degree of divergence -2, thus their calculation to $\mathcal{O}(a^2)$ can be performed in D-dimensions, with D slightly greater that 4. Let us illustrate the procedure with one of these terms: $k^4/(\left(k^2\right)^2\,(k+a\,p)^4)$. By appropriate substitutions of

$$\frac{1}{(k+\bar{p})^2} = \frac{1}{k^2} + \frac{-2(k\cdot\bar{p})-\bar{p}^2}{k^2\,(k+\bar{p})^2} \qquad (\bar{p}\equiv a\,p) \tag{B9}$$

we split this term as follows

$$\begin{aligned}\frac{k^4}{\left(k^2\right)^2\,(k+\bar{p})^2} &= \left[\frac{k^4}{\left(k^2\right)^3} + \frac{k^4\left(-2(k\cdot\bar{p})-\bar{p}^2\right)}{\left(k^2\right)^4} + \frac{4\,k^4(k\cdot\bar{p})^2}{\left(k^2\right)^5}\right] \\ &+ \left(\frac{k^4\left(4(k\cdot\bar{p})\bar{p}^2+(\bar{p}^2)^2\right)}{\left(k^2\right)^4\,(k+\bar{p})^2} + \frac{4\,k^4(k\cdot\bar{p})^2\left(-2(k\cdot\bar{p})-\bar{p}^2\right)}{\left(k^2\right)^5\,(k+\bar{p})^2}\right)\end{aligned} \tag{B10}$$

The part in square brackets is polynomial in a and can be integrated easily, using D-dimensional spherical coordinates. The remaining part is UV-convergent; thus the integration domain can now be recast in the form

$$\int_{|k|\leq\mu} = \int_{|k|<\infty} - \int_{\mu\leq|k|<\infty} \tag{B11}$$

The integral over the whole space can be performed using the methods of Ref. [32], whereas the integral outside the sphere of radius μ is $\mathcal{O}(a^3)$ and may be safely dropped. The same procedure is applied to the last term of I_2. Adding the contributions from all the steps described above, we check that the result is independent of μ.

Appendix C: A basis of divergent integrals

The most challenging aspect of this calculation, which demands careful attention, is isolating the dependence on the external momentum p and the lattice spacing a from the divergent terms. The singularities can be systematically extracted, and below we provide a list of primitively divergent integrals that arise in our algebraic expressions.

In the following integrals we define

$$\begin{aligned}\hat{k}_\mu &= 2\,\sin(\frac{k_\mu}{2}) \\ \hat{k}^2 &= 4\sum_\mu \sin^2(\frac{k_\mu}{2}) \\ \overset{\circ}{k}_\mu &= \sin(k_\mu)\end{aligned}$$

In addition, $(\ )_S$ means sum over inequivalent permutations. No summation over the indices μ, ν, ρ, σ is implied, unless otherwise stated.

$$\bullet \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{1}{\hat{k}^2\,\widehat{k+a\,p}^{\,2}} = 0.036678329075 - \frac{\ln(a^2p^2)}{16\pi^2} + 0.0000752406(3)\,a^2\,p^2 + a^2\,\frac{\sum_\mu p_\mu^4}{384\pi^2\,p^2} + \mathcal{O}(a^4\,p^4) \tag{C1}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{\overset{\circ}{k}_\mu}{\hat{k}^2\,\widehat{k+a\,p}^{\,2}} = a\,p_\mu\Big[&-0.008655827648 + \frac{\ln(a^2p^2)}{32\pi^2} \\ &- 0.0005107825(2)\,a^2\,p^2 + 0.001171329715\,a^2\,p_\mu^2 \\ &- a^2\,\frac{\sum_\mu p_\mu^4}{768\pi^2\,p^2} + a^2\,\frac{\ln(a^2p^2)}{384\pi^2}\left(\frac{p^2}{2}-p_\mu^2\right)\Big] \\ &+\mathcal{O}(a^5\,p^5)\end{aligned} \tag{C2}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\overset{\circ}{k}_\mu \, \overset{\circ}{k}_\nu}{(\hat{k}^2)^2 \, \widehat{k+a\,p}^{\,2}} = {} & \delta_{\mu\nu}\Big[0.004327913824 - \frac{\ln(a^2p^2)}{64\pi^2} \\ & + 0.00025539124(8)\,a^2\,p^2 - 0.000135654113\,a^2\,p_\mu^2 \\ & + a^2\,\frac{\sum_\mu p_\mu^4}{1536\pi^2\,p^2} + a^2\,\frac{\ln(a^2p^2)}{768\pi^2}\left(p_\mu^2 - \frac{p^2}{2}\right)\Big] \\ + {} & a^2\,p_\mu\,p_\nu\Big[\frac{1}{32\pi^2\,a^2\,p^2} - 0.0003788538(2) + \frac{\sum_\mu p_\mu^4}{768\pi^2\,(p^2)^2} \\ & - \frac{(p_\mu^2+p_\nu^2)}{384\pi^2\,p^2} + \frac{\ln(a^2p^2)}{768\pi^2}\Big] + \mathcal{O}(a^4\,p^4)\end{aligned} \tag{C3}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\overset{\circ}{k}_\mu \, \overset{\circ}{k}_\nu}{\hat{k}^2 \, \widehat{k+a\,p}^{\,2}} = {} & \delta_{\mu\nu}\Big[0.014966695116 - 0.001256484446\,a^2\,p^2 \\ & - 0.001027789631\,a^2\,p_\mu^2 + \frac{a^2\,p^2\,\ln(a^2p^2)}{192\pi^2}\Big] \\ + {} & a^2\,p_\mu\,p_\nu\Big[0.003970508789 - \frac{\ln(a^2p^2)}{48\pi^2}\Big] + \mathcal{O}(a^4\,p^4)\end{aligned} \tag{C4}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{(\overset{\circ}{k}_\mu)^3}{\hat{k}^2 \, \widehat{k+a\,p}^{\,2}} = {} & a\,p_\mu\Big[-0.006184131744 + 0.001102333439\,a^2\,p^2 \\ & - 0.000174224479\,a^2\,p_\mu^2 + a^2\,\frac{\ln(a^2p^2)}{64\pi^2}\left(p_\mu^2 - \frac{p^2}{2}\right)\Big] \\ + {} & \mathcal{O}(a^5p^5)\end{aligned} \tag{C5}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\overset{\circ}{k}_\mu \, \overset{\circ}{k}_\nu \, \overset{\circ}{k}_\rho}{(\hat{k}^2)^2 \, \widehat{k+a\,p}^{\,2}} = {} & (\delta_{\nu\rho}\,a\,p_\mu)_S\Big[-0.000728769948 + \frac{\ln(a^2p^2)}{192\pi^2}\Big] \\ + {} & 0.001027789631\delta_{\mu\nu\rho}\,a\,p_\mu - a\,\frac{p_\mu\,p_\nu\,p_\rho}{48\pi^2\,p^2} + \mathcal{O}(a^3\,p^3)\end{aligned} \tag{C6}$$

$$\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\sum_\mu \hat{k}_\mu^4}{16(\hat{k}^2)^2\, \widehat{k+a\,p}^{\,2}} = 0.004050096698 - 0.000107954163\, a^2\, p^2 + a^2\, \frac{\sum_\mu p_\mu^4}{1024\pi^2\, p^2} + \mathcal{O}(a^4\, p^4) \tag{C7}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\mathring{k}_\mu\, \mathring{k}_\nu\, \mathring{k}_\rho\, \mathring{k}_\sigma}{(\hat{k}^2)^2\, \widehat{k+a\,p}^{\,2}} =\ & 0.001589337971\, (\delta_{\mu\nu}\, \delta_{\rho\sigma})_S - 0.001675948042\, \delta_{\mu\nu\rho\sigma} \\ & - 0.000372782983(\delta_{\mu\nu\rho}\, a^2\, p_\mu\, p_\sigma)_S - 0.000062130497(\delta_{\mu\nu}\, \delta_{\rho\sigma}\, a^2\, p_\mu^2)_S \\ & + \delta_{\mu\nu\rho\sigma}\left(0.000186391491\, a^2\, p^2 + 0.000410290033\, a^2\, p_\mu^2\right) \\ & + (\delta_{\mu\nu}\, a^2\, p_\rho\, p_\sigma)_S \left(0.000227848225 - \frac{\ln(a^2p^2)}{384\pi^2}\right) \\ & + (\delta_{\mu\nu}\, \delta_{\rho\sigma})_S\, a^2\, p^2 \left(-0.000245852737 + \frac{\ln(a^2p^2)}{768\pi^2}\right) \\ & + a^2\, \frac{p_\mu\, p_\nu\, p_\rho\, p_\sigma}{64\pi^2\, p^2} + \mathcal{O}(a^4\, p^4)\end{aligned} \tag{C8}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\mathring{k}_\nu \sum_\mu \hat{k}_\mu^4}{16(\hat{k}^2)^2\, \widehat{k+a\,p}^{\,2}} = a\, p_\nu \Big[& -0.000800034900 + 0.000069705553\, a^2\, p^2 \\ & + 0.000107082394\, a^2\, p_\nu^2 - a^2\, \frac{\sum_\rho p_\rho^4}{1280\pi^2\, p^2} \\ & - a^2\, \frac{\ln(a^2p^2)}{2560\pi^2} \left(\frac{p^2}{2} - p_\nu^2\right)\Big] + \mathcal{O}(a^5\, p^5)\end{aligned} \tag{C9}$$

$$\begin{aligned}\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{\mathring{k}_\nu\, \mathring{k}_\rho \sum_\mu \widehat{k_\mu{+}a\,p_\mu}^{\,4}}{16(\hat{k}^2)^2\, (\widehat{k{+}a\,p}^{\,2})^2} =\ & \delta_{\nu\rho}\Big[0.000400017450 - 0.000034852777\, a^2\, p^2 + a^2\, \frac{\sum_\mu p_\mu^4}{2560\pi^2\, p^2} \\ & + 0.000105349447\, a^2\, p_\nu^2 + a^2\, \frac{\ln(a^2p^2)}{5120\pi^2}(\frac{p^2}{2} - 3p_\nu^2)\Big] \\ & + a^2\, p_\nu\, p_\rho\Big[0.000006643045 - \frac{p_\nu^2 + p_\rho^2}{2560\pi^2\, p^2} + \frac{\sum_\mu p_\mu^4}{5120\pi^2\, (p^2)^2} \\ & + \frac{\ln(a^2p^2)}{5120\pi^2}\Big] + \mathcal{O}(a^4\, p^4)\end{aligned} \tag{C10}$$

Appendix D: Well-Known One-Loop Lattice Integrals P_1, P_2

Some one-loop and two-loop integrals can be found in Ref [34]. We list below the most well-known of them. Eqs. (D1)-(D5) refer to one-loop integrals with zero external momentum.

$$\bullet \int_{-\pi}^{\pi} \frac{d^4k}{(2\pi)^4} \frac{1}{\hat{k}^2} = P_1 \tag{D1}$$

with the numerical value: $P_1 = 0.15493339023106021(1)$.

$$\bullet P_2 = \lim_{m\to 0}\left(\frac{1}{(4\pi)^2}\ln(m^2) + \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{1}{(\hat{k}^2+m^2)^2}\right) \tag{D2}$$

with the numerical value: $P_2 = 0.02401318111946489(1)$.

$$\bullet \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{\sum_\mu \hat{k}_\mu^4}{(\hat{k}^2)^2} = 1 - 4P_1 \tag{D3}$$

$$\bullet \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{\sum_\mu \hat{k}_\mu^4}{(\hat{k}^2)^3} = \frac{1}{2}P_1 - \frac{1}{8\pi^2} \tag{D4}$$

$$\bullet \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{\sum_\mu \hat{k}_\mu^6}{(\hat{k}^2)^3} = 1 - 5P_1 - \frac{1}{2\pi^2} \tag{D5}$$

$$\bullet \int_{-\pi}^{\pi}\frac{d^4k}{(2\pi)^4}\frac{1}{\hat{k}^2\widehat{k+ap}^{\,2}} = A(ap), \tag{D6}$$

where $A(p) = \frac{1}{(4\pi)^2}\left(-\ln(a^2p^2)+2\right) + P_2 + O(a^2p^2)$ and Eq. (C1) gives also the $O(a^2p^2)$ contribution.

Acknowledgments

The project (CONCEPT/0823/0052) is implemented under the programme of social cohesion "THALIA 2021-2027" co-funded by the European Union through the Research and Innovation Foundation (RIF).

[1] C. Morningstar and M. J. Peardon, "Analytic smearing of SU(3) link variables in lattice QCD," Phys. Rev. D **69** (2004), 054501 doi:10.1103/PhysRevD.69.054501 [arXiv:hep-lat/0311018 [hep-lat]].
[2] R. Horsley, H. Perlt, P. E. L. Rakow, G. Schierholz and A. Schiller, "Perturbative determination of c(SW) for plaquette and Symanzik gauge action and stout link clover fermions," Phys. Rev. D **78** (2008), 054504 doi:10.1103/PhysRevD.78.054504 [arXiv:0807.0345 [hep-lat]].
[3] R. K. Ellis and G. Martinelli, "Two Loop Corrections to the Λ Parameters of One-Plaquette Actions," Nucl. Phys. B **235** (1984), 93-114.
[4] M. Luscher and P. Weisz, "Background field technique and renormalization in lattice gauge theory," Nucl. Phys. B **452** (1995), 213-233.
[5] P. Cvitanovic, Phys. Rev. **D14** (1976) 1536
[6] I. Gradshteyn, I. M. Ryzhik, "Tables of integrals, series, and products," Elsevier, 2007, Equation 9.132.1.
[7] K. Symanzik, Nucl. Phys. **B226** (1983) 187; Nucl. Phys. **B226** (1983) 205
[8] R. Frezzotti, P. Grassi, S. Sint, P. Weisz, JHEP **08** (2001) 058 [hep-lat/0101001].
[9] R. Frezzotti, G. Rossi, JHEP **08** (2004) 007 [hep-lat/0306014].
[10] G. Martinelli and Y. Zhang, Phys. Lett. **123B** (1983) 433.
[11] S. Aoki, K. Nagai, Y. Taniguchi and A. Ukawa, Phys. Rev. **D58** (1998) 074505 [hep-lat/9802034].
[12] S. Capitani et al., Nucl. Phys. **B593** (2001) 183 [hep-lat/0007004].
[13] C. Alexandrou, E Follana, H. Panagopoulos and E. Vicari, Nucl.Phys. **B580** (2000) 394 [hep-lat/0002010].
[14] S. Capitani and L. Giusti, Phys. Rev. **D62** (2000) 114506 [hep-lat/0007011].
[15] R. Horsley et al., Nucl. Phys. **B693** (2004) 3; Erratum-ibid. **B713** (2005) 601 [hep-lat/0404007].
[16] M. Ioannou and H. Panagopoulos, Phys. Rev. **D73** (2006) 054507 [hep-lat/0601020].
[17] S. Aoki, K. Nagai, Y. Taniguchi, A. Ukawa, Phys. Rev. **D58** (1998) 074505 [hep-lat/9802034].
[18] A. Skouroupathis, H. Panagopoulos, Phys. Rev. **D76** (2007) 094514 [arXiv:0707.2906].
[19] A. Skouroupathis, H. Panagopoulos, Two-loop renormalization of vector, axial-vector and tensor fermion bilinears on the lattice, Phys. Rev. **D79** (2009) 094508 [arXiv:0811.4264].
[20] F. Di Renzo, V. Miccio, L. Scorzato and C. Torrero, PoS LAT2006 (2006) 156 [hep-lat/0609077].
[21] F. Di Renzo, V. Miccio, L. Scorzato and C. Torrero, Eur. Phys. J. **C51** (2007) 645 [hep-lat/0611013].

[22] F. Di Renzo, L. Scorzato and C. Torrero, PoS LAT2007 (2007) 240 [arXiv:0710.0552].
[23] Y. Iwasaki, Univ. of Tsukuba Report UTHEP-118 (1983).
[24] T. Takaishi, Phys. Rev. D54 (1996) 1050.
[25] M. Lüscher and P. Weisz, Commun. Math. Phys. 97 (1985) 59; Erratum-ibid. 98 (1985) 433.
[26] M. G. Alford, W. Dimm, G. P. Lepage, G. Hockney and P. B. Mackenzie, Phys. Lett. B 361 (1995) 87 [hep-lat/9507010].
[27] K. Osterwalder and E. Seiler, Ann. Phys. (NY) 110 (1978) 440.
[28] P. Dimopoulos et al., PoS LAT2007 (2007) 241 [arXiv:0710.0975]; ETM Collaboration, in preparation.
[29] G. Martinelli, C. Pittori, C. T. Sachrajda, M. Testa and A. Vladikas, Nucl. Phys. B **445** (1995) 81 [hep-lat/9411010].
[30] H. Panagopoulos, E. Vicari, Nucl. Phys. **B332** (1990) 261.
[31] H. Kawai, R. Nakayama, K. Seo, Nucl. Phys. **B189** (1981) 40.
[32] K.G. Chetyrkin and F.V. Tkachov, Nucl. Phys. **B192** (1981) 159.
[33] K. G. Chetyrkin and A. Retey, Nucl. Phys. B **583** (2000) 3 [arXiv:hep-ph/9910332].
[34] M. Luscher and P. Weisz, Nucl. Phys. B **452** (1995), 234-260 doi:10.1016/0550-3213(95)00338-S [arXiv:hep-lat/9505011 [hep-lat]].