Westfälische
Wilhelms-Universität
Münster

institut für
theoretische physik

Implementation of a Program for QCD and HQET Correlation Functions in the Schrödinger Functional

Diploma Thesis

Submitted by Christian Wittemeier

February 23, 2012

Advisor PD Dr. Jochen Heitger

Contents

1. Introduction

1.1. Methods in QCD

Quantum chromodynamics (QCD) is widely believed to be the theory of strong interactions[Gre04, p. v] and to explain the production of quark and gluon jets in high-energetic collisions as well as the masses of the observed hadron spectrum. It is, however, much more difficult to make predictions in QCD than in quantum electrodynamics (QED) for example. The electromagnetic coupling strength is small enough to justify the application of perturbation theory, and comparisons to experiments have shown that it gives quite accurate results, but in QCD the perturbative approach fails in many situations, because the coupling strength is of the order of one. Quarks and gluons are then confined in hadrons and do not appear as free particles. However, due to the self-interaction of the gluons the renormalized coupling decreases towards higher energies and quarks and gluons are only weakly interacting. Thus, above energy scales of several GeV, they can be treated in a perturbative expansion. This "asymptotic freedom" was found by Gross, Wilczek and Politzer in 1973 [Gro73, Pol73]. But low-energy observables like hadron masses and matrix elements are not accessible by perturbation theory.

Until 1974, all predictions in QCD were based on the perturbative approach [Rot05, p. 2], but in that year, Wilson proposed a formulation of QCD on a lattice [Wil74] (see also [Wil05] for a review), which opened up the possibility of non-perturbative studies. The path integral in a finite and discrete space-time can be evaluated numerically on a computer, and the results for the given lattice are exact up to statistical errors [Lü03, p. 5]. To make physical predictions, the limits of zero lattice spacing and infinite volume must still be performed, but in general the uncertainties associated with them can be brought under control [Lü03, p. 5]. For example, the masses of the light hadrons have been determined in [Aok03].

The Wilson action of lattice QCD is introduced in section 2.3. In particular, we will use the framework of the Schrödinger functional, which is defined in a finite space-time with boundaries (see section 2.2.7). It has several advantages over a fully periodic space-time, e.g. the boundary gauge field can be used to induce a background field, and a renormalized coupling can be defined that runs with the extent L of the space-time. This was done by Lüscher et al. to study the running of the coupling from the non-perturbative region up to energies, where the coupling becomes small enough for perturbation theory [Lü94]. The fermion fields at the boundaries can be used to define correlation functions which result in smaller statistical errors [Hof05, p. 31] [Gua99]. We, too, will employ them as sources for correlation functions (sections 2.3.8 and 3.6). Furthermore, the boundaries of the

Schrödinger-functional space-time induce a gap in the spectrum of the Dirac operator [Fri09, p. 44]. That means that even simulations with massless quarks can be carried out.

1.2. HQET and Matching

However, lattice QCD is not suitable for simulating too heavy quarks, because the lattice spacing must be smaller than the inverse quark mass $1/m_\mathrm{h}$. Especially for the b quark, the necessary lattices are beyond the capabilities of the current computers, but more accurate predictions for the properties of B mesons, which include a b quark and a light u or d quark, are needed to probe the Standard Model. So, instead, one uses effective theories. A possible choice is the heavy-quark effective theory (HQET), which is explained in chapter 3. The lattice-QCD action and the considered observables are expanded in a power series in $1/m_\mathrm{h}$. At the classical level, the coefficients in this series are known (see table 1 in [Blo10]), but they are renormalized in the quantum theory and become unknown parameters. They could be obtained by comparing the results of HQET directly to experimental ones and setting the parameters in a way that both agree, but this would spoil the predictivity of the theory [Hei04b, p. 3]. Instead, one can relate the HQET results to results from the underlying QCD. This procedure is called "matching". Below, it will be described in detail how it can be done in a non-perturbative way. Since a simulation of the b quark in QCD requires very small lattice spacings, the matching must be done in a small physical volume, so that the total number of lattice points does not become too great. However, one would like to know the HQET parameters at larger lattice spacings a, so they can be used in larger volumes. Therefore, in a second step, the "step scaling", the obtained parameters are carried over to larger values of a, as they are used in standard HQET in larger volumes, where physical predictions can be made [Blo10].

In the following paragraphs, I will try to explain the matching procedure in more detail, since we plan to use the SFCF program that is described in this work for this application.

To explain the matching procedure, I will recapitulate the work of Blossier, Della Morte, Garron and Sommer in [Blo10]: They determined the parameters of the HQET expansion of the action and the time component of the axial heavy–light current up to $\mathcal{O}(1/m_\mathrm{h})$. These are five unknown parameters, which we will summarize in a vector $\omega = (\omega_1, \dots, \omega_5)^\mathrm{t}$. Consequently, five observables $\Phi = (\Phi_1, \dots, \Phi_5)^\mathrm{t}$ are needed to determine them. They were defined in the Schrödinger functional on a hyper-cubic space-time with $T = L_1 = L_2 = L_3 \equiv L$:

$$\Phi = \left(L\Gamma^\mathrm{P}, \ln\left(\frac{-f_\mathrm{A}(T/2,\theta_0)}{\sqrt{f_1(T/2,\theta_0)}}\right), R_\mathrm{A}, R_1, \frac{3}{4}\ln\left(\frac{f_1(\theta_0)}{k_1(\theta_0)}\right)\right)^\mathrm{t} \tag{1.1}$$

with

$$\Gamma^{\mathrm{P}} = -\,\widetilde{\partial}_0 \ln\left(-f_{\mathrm{A}}(x_0,\theta_0)\right)\Big|_{x_0=T/2} \tag{1.2}$$

$$R_{\mathrm{A}} = \ln\left(\frac{f_{\mathrm{A}}(T/2,\theta_1)}{f_{\mathrm{A}}(T/2,\theta_2)}\right) \tag{1.3}$$

$$R_1 = \frac{1}{4}\ln\left(\frac{f_1(\theta_1)k_1(\theta_1)^3}{f_1(\theta_2)k_1(\theta_2)^3}\right). \tag{1.4}$$

f_{A}, fo and k_1 are three correlation functions, expectation values of field products, in the Schrödinger functional that will be introduced in section 2.3.8. θ_0, θ_1, θ_2 can be interpreted as additional phases in the space-like boundary conditions of the quark fields (see section 2.3.3).

These observables are chosen in such a way that they have a sensitivity to the parameters ω and their HQET expansion is linear in ω. Using matrix–vector notation, it can be written as

$$\Phi^{\mathrm{HQET}}(L,M,a) = \eta(L,a) + \phi(L,a)\cdot\omega(M,a). \tag{1.5}$$

Φ^{HQET} and η are vectors like ω, ϕ is a matrix. ϕ and η can be computed via the HQET correlation functions defined in section 3.6. The quantities are parametrized by the space size L, the lattice spacing a and the renormalization-group invariant mass of the heavy quark M.

Now, the matching is done in the following way (see also [Som10, p. 52]).

1. On a small lattice with size L_1, the QCD observables are computed and extrapolated to the continuum limit $a \to 0$:

$$\Phi^{\mathrm{QCD}}(L_1,M,0) = \lim_{a\to 0}\Phi^{\mathrm{QCD}}(L_1,M,a). \tag{1.6}$$

2. $\phi(L_1,a)$ and $\eta(L_1,a)$ are computed for different lattice spacings a. The corresponding parameters $\tilde{\omega}(M,a)$ are obtained from the matching condition

$$\Phi^{\mathrm{HQET}}(L_1,M,a) = \Phi^{\mathrm{QCD}}(L_1,M,0) \text{ for all } a \tag{1.7}$$

$$\Longrightarrow \qquad \tilde{\omega}(M,a) = \phi^{-1}(L_1,a)\cdot\left[\Phi^{\mathrm{QCD}}(L_1,M,0) - \eta(L_1,a)\right]. \tag{1.8}$$

3. However, in this way, only lattice spacings $a \lesssim 0.05\,\mathrm{fm}$ are accessible. Therefore a second step, similar to the first two, is performed, the step scaling: With $\tilde{\omega}(M,a)$, the HQET observables Φ^{HQET} are calculated at a greater lattice size $L_2 = s\cdot L_1$ (e.g. $s=2$) and extrapolated to the continuum:

$$\Phi^{\mathrm{HQET}}(L_2,M,0) = \lim_{a\to 0}\left\{\eta(L_2,a) + \phi(L_2,a)\cdot\tilde{\omega}(M,a)\right\}. \tag{1.9}$$

4. The parameters $\omega(M,a)$ for larger a are obtained from

$$\omega(M,a) = \phi^{-1}(L_2,a)\cdot\left[\Phi^{\mathrm{HQET}}(L_2,M,0) - \eta(L_2,a)\right]. \tag{1.10}$$

Thus, one arrives at values up to $a \approx 0.1\,\mathrm{fm}$.

1.3. The Purpose of this Work

The subject of this work is the implementation of the SFCF program, in which I was involved. One of the applications of this program is to compute the QCD and HQET observables for the matching of the vector current V_k. In the previous section, the matching of the axial current A_0 was described. The HQET expansion of the vector current brings new parameters ω_i with it, and accordingly, new observables are necessary for the matching which are sensitive to the vector-current parameters. Potential matching observables have been studied by Samantha Dooling at tree level in perturbation theory [Doo11]. They include correlation functions in the vector channel like k_{V} and k_1 (see section 2.3.8). In some cases the correlations must be evaluated for different θ values of the light and heavy quark. Furthermore, some HQET correlations involve new combinations of spatial derivatives acting on the light- and heavy-quark fields.

The SFCF program is designed to be flexible enough to compute all the necessary QCD and HQET correlation functions. The θ-values can be set individually for each quark flavor and also in each space direction. Generic routines are provided which calculate correlation functions with arbitrary combinations of gamma matrices. The program is still not finished. A main program that is ready to be used is missing yet, but some practical computations have already been successfully performed with a test program. Jochen Heitger has calculated several QCD correlation functions on the PALMA cluster computer at the university of Münster, which will be used by Michael Topp [Top12].

In the chapters 2 and 3, I will introduce the quantum chromodynamics and heavy-quark effective theory, in particular their formulation on a lattice with Schrödinger-functional boundary conditions. The correlation functions that are to be computed will be defined there, and it is shown how to evaluate them. Chapter 4 starts to describe their implementation in the SFCF program. A short introduction to the program and some of its features is given. Chapters 5 and 6 discuss the implementation of the routines which compute the QCD and the HQET correlation functions, which will constitute the main part of my thesis. They also explain how to use them and how to compute the propagators. Chapter 7 provides an outlook towards the main program which does not exist yet. Finally, chapter 8 shows a first application of the program, the study of the large-mass behavior of QCD observables and how they approach the corresponding static HQET observables.

1.4. SFCF Program Version and Source

The SFCF source code is available through a git repository under `pub.ifh.de/afs/ifh.de/group/alpha/bup/alpha/git_repositories/sfcf`. The version that I use as reference in this thesis is labeled by the key `839fe5466d9f4a55475ae25f6d2539dbe8717be2`. Full file and directory names refer to the root directory of this repository. The repository contains a preliminary documentation in `src/doc/corr.pdf`.

2. Quantum Chromodynamics (QCD)

The theory of quantum chromodynamics (QCD) is one of the fundamental theories of particle physics. The others are the theory of electroweak interactions and the theory of general relativity. Both the electroweak theory and QCD are quantum field theories which can be described in a common framework, the Standard Model. The next section will provide a brief overview of it. The later sections will describe quantum chromodynamics in more detail. Section 2.2 treats QCD in continuous space-time, and section 2.3 shows how it can be modelled on a discrete lattice, especially with Schrödinger-functional boundary conditions. In section 2.3.8 the QCD correlation functions that we will compute in chapter 5 are introduced and expressed in terms of propagators.

2.1. The Standard Model

Introductions to the Standard Model of particle physics are e.g. given in [Cot07, Bur07].

In contrast to the theory of general relativity, the Standard Model (SM) of particle physics is a quantum theory. The objects that it describes are fields, whose excited energy eigenstates can be interpreted as assemblies of particles [Cot07, p. 78]. Symmetries play an important role in the SM, especially "gauge symmetries". The fields (or particles) are classified according to their behavior under symmetry transformations, which is characterized by several quantum numbers, e.g. they are said to have a spin of 1, if the field's components transform like a vector under spatial rotations. Other examples of quantum numbers are the mass, the electric charge, the weak isospin, the color charge, baryon number and flavor.

A major distinction for the particles is the one between fermions and bosons. Fermions are particles with a half-integral spin, whereas bosons have an integral spin. The fermions split up into quarks, which carry color charge, and leptons, which do not. There are six kinds of leptons and six kinds of quarks with different masses. They can be grouped into three "generations", each of which contains two quarks with electric charges $+2/3$ and $-1/3$ (e.g. the up and down quark) and two leptons, one with a negative charge and a neutral one (e.g. the electron and the electron neutrino). These generations are very similar to each other, and their members take part in the same interactions. They can only be distinguished by their masses.

The fermions interact with each other by the exchange of gauge bosons. These are the photon, which mediates the electromagnetic force, the $\mathrm{W}^{\pm}$ and Z particles for the weak interaction and the gluons for the strong or color interaction. These bosons are named “gauge bosons”, because they ensure the invariance of the theory under gauge transformations. However, the $\mathrm{W}^{\pm}$ and Z bosons are observed to have masses of nearly 100 GeV [Nak10] (which is the reason why the weak interaction acts only on small distances). If these masses were introduced in the theory by simply adding mass terms like e.g. $-m_{\mathrm{Z}}^2/2 \cdot Z_\mu Z^\mu$ (with the field Z_μ representing the Z boson) to the Lagrangian, the gauge symmetry would be explicitly broken, and the theory would be no longer renormalizable [Cot07, p. 102].

Instead, the masses of the $\mathrm{W}^{\pm}$ and Z bosons can be created dynamically through the interaction with the so-called Higgs field [Cot07, p. 107]. In this Salam–Weinberg model, there is an exact gauge symmetry ($\mathrm{SU}(2)_{\mathrm{L}} \times \mathrm{U}(1)_{\mathrm{Y}}$) at high energies, corresponding to four massless gauge bosons W^+, W^0, W^- and B^0. These bosons interact with the Higgs field, which has a set of degenerate ground states due to its self-interaction. At low energies, the Higgs field randomly assumes one of its ground states. This spontaneous choice of a ground state breaks the electroweak gauge symmetry down to the electromagnetic gauge group $\mathrm{U}(1)_{\mathrm{Q}}$, and, through the interaction with the Higgs field, three gauge bosons acquire an effective mass. The mass eigenstates are the W^+, the W^- and two linear combinations of W^0 and B, which are the Z boson and the massless photon.

The other part of the Standard Model is quantum chromodynamics, which describes the strong interaction between particles with color charge and the associated gauge bosons, the gluons. The elementary particles that carry color charge are the quarks and the gluons themselves, too. That means gluons can also interact with themselves, in contrast to the photon. This self-interaction leads to the phenomenon of “confinement”, which means that no particle with color charge may exist as free particle, especially, quarks can not be observed directly. They only appear as constituents of color-neutral compound particles, e.g. mesons, which contain a quark and an anti-quark, or baryons, which consist of three quarks. It is also the reason for the fact that the strong force is limited to circa 1 fm, although its carrier particle, the gluon, is massless like the photon and thus should have an infinite range.

2.2. QCD in Continuous Space

Quantum chromodynamics (QCD) is the theory of the strong or “color” force. An introduction can be found in [Cot07, p. 153]. QCD resembles quantum electrodynamics (QED) in many ways. Both are built on a gauge symmetry, QED on the Abelian group U(1), QCD on the non-Abelian group SU(3). This means that there are three color charges in QCD, instead of a single electric charge. These charges are sometimes called red, green and blue, but these are only symbolic names. The non-Abelian character of the group SU(3) gives rise to additional interaction terms in QCD, namely the self-interaction of the gluons, which have no analog in QED.

In this section, we will look at the particles or fields that are described by QCD and the Lagrange density which defines their dynamics, and roughly sketch how the classical Lagrange density is used to construct a quantum theory.

2.2.1. Fields

The particles of QCD are quarks and gluons. The quarks are described by a complex field $\psi(x)$ with several components. First, the quarks are spin-1/2 particles that are represented by a Dirac spinor, which has four components. This can be motivated by the fact that they describe particles and antiparticles with two spin states. Hence, the field has a Dirac index $A \in \{1, \dots, 4\}$: $\psi_A(x)$. All Dirac indices in this text will be denoted by capital Latin letters. Furthermore, the quarks carry color charge. They are in the fundamental representation of the group SU(3). That means they are three-dimensional vectors in color space. The gauge group SU(3) is the group of rotations in this color space, i.e. a gauge transformation is represented as a unitary matrix with a determinant of one. Thus, the quark field has also a color index $\alpha \in \{1, 2, 3\}$: $\psi_{A\alpha}(x)$. (Color indices are denoted as lowercase Greek letters.) Finally, there are different quark flavors, characterized by their masses. They are indicated by a third index f: $\psi^{(f)}_{A\alpha}(x)$. In nature, six flavors occur, but often not all of them are taken into account, or flavors with different masses are investigated. I will often omit the flavor index, if it is not necessary to distinguish flavors. Also the Dirac and color indices will often be left out, and $\psi(x)$ will be written in vector notation without explicitly referring to its components.

The gluons are massless spin-1 particles, the gauge field $A(x)$ that represents them is a 4-vector, and thus has an index with four possible values: $A_\mu(x)$. $\mu = 0$ will represent the time component and $\mu = 1, 2, 3$ the space components. Each component is a complex, Hermitian 3×3 matrix with a trace of zero. These matrices act in color space. So, the field $[A_\mu(x)]_{\alpha\beta}$ has two additional color indices.[1]

Mathematically, the $A_\mu(x)$ are elements of the Lie algebra $\mathfrak{su}(3)$ of SU(3). This Lie Algebra is the tangent space of an SU(3) element, which can be visualized as the infinitesimal small environment of that element. In QCD, they characterize the infinitesimal deviations of the coordinate systems in color space at x and $x + \mathrm{d}x$.

2.2.2. Lagrange Density

The Lagrange Density of quantum chromodynamics in Minkowski space-time is

$$\mathcal{L}(x) = \sum_f \bar{\psi}^f(x) \cdot (\mathrm{i}\gamma^\mu D_\mu - m_f)\, \psi^f(x) - \frac{1}{2g^2} \mathrm{Tr}\, [F_{\mu\nu}(x) F^{\mu\nu}(x)] \,. \tag{2.1}$$

In this equation, m_f is the mass of flavor f, g is the coupling strength of gluons to color charge, γ^μ are the Dirac or gamma matrices. They are Hermitian, act in the space of Dirac spinors, and in Minkowski space-time they must fulfill $\{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu}$. The adjoint of a Dirac spinor is $\bar{\psi} = \psi^\dagger \gamma^0$. D_μ is of particular importance. It is the covariant derivative, defined as

$$D_\mu = \partial_\mu + \mathrm{i}A_\mu(x). \tag{2.2}$$

[1] Another notation is to write $A_{\mu a}(x) \cdot T^a$ where the matrices $T^1, \dots, T^8$ constitute a basis for Hermitian, traceless matrices considered as a real vector space and $A_{\mu a}(x)$ is a real number. The T^a are called the generators of the Lie algebra $\mathfrak{su}(3)$.

∂_μ is the usual, coordinate derivative, and $A_\mu(x)$ is the gauge field, which represents the gluons. Mathematically, it defines a "connection" on the vector bundle of color vectors over Minkowski space-time, i.e. it defines how color vectors at different points are to be compared, and thus how a derivative of a color field is calculated. But here it plays a greater role than just tracing different coordinate systems. It is a dynamical quantity itself. Its dynamics are determined by the last term in the Lagrange density, which involves the field-strength tensor $F_{\mu\nu}(x)$. It is defined as

$$F_{\mu\nu}(x) = -\mathrm{i}[D_\mu, D_\nu] = \partial_\mu A_\nu(x) - \partial_\nu A_\mu(x) + \mathrm{i}[A_\mu(x), A_\nu(x)]. \tag{2.3}$$

The last term shows that the gluons interact with themselves (via three- and four-particle vertices), whereas in QED the commutator vanishes and there is no direct photon–photon interaction.

2.2.3. Gauge Symmetry

A gauge transformation rotates the coordinate system for vectors in color space. Equivalently, we can say that we rotate the vectors instead of the coordinate system. Such a rotation is described by a unitary 3×3 matrix Ω with determinant one, i.e. $\Omega \in \mathrm{SU}(3)$. If Ω depends on the space-time point x, it is called a "local" transformation, in contrast to a "global" transformation, which does not depend on x.

The QCD Lagrangian should be invariant under SU(3) gauge transformations, even under local ones. If the color coordinate systems at different points are rotated in different ways, this should be compensated by the gauge field $A_\mu(x)$, because it is its very purpose to record the differences in the coordinate systems.

Under a local gauge transformation $\Omega(x)$, the quark fields transform as

$$\psi(x) = \Omega(x) \cdot \psi'(x) \qquad\qquad \bar{\psi}(x) = \bar{\psi}'(x) \cdot \Omega^\dagger(x). \tag{2.4}$$

The transformation law of the gauge field is

$$A_\mu(x) = \Omega(x) \cdot A'_\mu(x) \cdot \Omega^\dagger(x) - \mathrm{i}\Omega(x) \left(\partial_\mu \Omega^\dagger(x)\right). \tag{2.5}$$

It is chosen in such a way that $D_\mu \psi(x) = \Omega(x) D'_\mu \psi'(x)$:

$$\begin{aligned}
D_\mu \psi(x) &= \left(\partial_\mu + \mathrm{i}A_\mu(x)\right) \psi(x) && (2.6)\\
&= \left(\partial_\mu + \mathrm{i}A_\mu(x)\right) \left(\Omega(x)\psi'(x)\right) && (2.7)\\
&= \Omega(x) \left(\partial_\mu + \Omega^\dagger(x) \left(\partial_\mu \Omega(x)\right) + \mathrm{i}\Omega^\dagger(x) A_\mu(x) \Omega(x)\right) \psi'(x) && (2.8)\\
&= \Omega(x) \left(\partial_\mu + \Omega^\dagger(x) \left(\partial_\mu \Omega(x)\right) + \mathrm{i}A'_\mu(x) + \left(\partial_\mu \Omega^\dagger(x)\right) \Omega(x)\right) \psi'(x) && (2.9)\\
&= \Omega(x) \left(\partial_\mu + \mathrm{i}A'_\mu(x)\right) \psi'(x) = D'_\mu \psi'(x). && (2.10)
\end{aligned}$$

($\Omega^\dagger(\partial_\mu \Omega) + (\partial_\mu \Omega^\dagger)\Omega = \partial_\mu(\Omega^\dagger \Omega) = \partial_\mu(1) = 0$ was used here.)

Under these transformation laws, the first term of the Lagrange density is invariant:

$$\begin{aligned}\bar{\psi}(x)\left(\mathrm{i}\gamma^\mu D_\mu - m\right)\psi(x) &= \bar{\psi}'(x)\Omega^\dagger(x)\left(\mathrm{i}\gamma^\mu D_\mu - m\right)\Omega(x)\psi'(x) && (2.11)\\ &= \bar{\psi}'(x)\Omega^\dagger(x)\Omega(x)\left(\mathrm{i}\gamma^\mu D'_\mu - m\right)\psi'(x) && (2.12)\\ &= \bar{\psi}'(x)\left(\mathrm{i}\gamma^\mu D'_\mu - m\right)\psi'(x). && (2.13)\end{aligned}$$

From the transformation law of $A_\mu(x)$, the transformation law of the field-strength tensor $F_{\mu\nu}(x)$ follows as:

$$F_{\mu\nu}(x) = \Omega(x)\cdot F'_{\mu\nu}(x)\cdot\Omega^\dagger(x). \tag{2.14}$$

Since the trace is taken, the second term of the Lagrange density stays invariant, too, and we arrive at

$$\begin{aligned}\mathcal{L} &= \sum_f \bar{\psi}^f(x)\cdot\left[\mathrm{i}\gamma^\mu D_\mu - m_f\right]\psi^f(x) - \frac{1}{2g^2}\mathrm{Tr}\left[F_{\mu\nu}(x)F^{\mu\nu}(x)\right] && (2.15)\\ &= \sum_f \bar{\psi}'^f(x)\cdot\left[\mathrm{i}\gamma^\mu D'_\mu - m_f\right]\psi'^f(x) - \frac{1}{2g^2}\mathrm{Tr}\left[F'_{\mu\nu}(x)F'^{\mu\nu}(x)\right]. && (2.16)\end{aligned}$$

So, the Lagrange density is invariant under gauge transformations. Moreover, the gauge-transformed fields must not be seen as a different state, but as the same state as before in another coordinate system.

2.2.4. Quantization

Until now, all fields were classical fields, which have a defined value at every space-time point. The transition to a quantum field theory can happen in different ways. The standard approach [Mag05, p. 83]] is to consider the free theory without interactions and to promote the free fields and their conjugate momenta to operators and to impose canonical equal-time commutation relations for bosons and anti-commutation relations for fermions. This is the same principle as in one-particle quantum mechanics, when the classical position and momentum variables are replaced by operators and it is demanded that $[Q_i, P'_i] = \mathrm{i}\delta_{ii'}$ ($\hbar$ is set to one). With these commutation rules, a suitable Hilbert space and the energy eigenstates and eigenvalues of the free Hamiltonian can be constructed. Expectation values of observables are calculated as scalar products $\langle 0|A|0\rangle$, where A is the operator associated with the observable and $|0\rangle$ is the theory's ground state.

However, this only describes free, non-interacting particles. The interaction is regarded as a perturbation of the free theory and observables are expressed as a power series in the coupling. This procedure is very suitable in quantum electrodynamics, where the coupling strength is small.[2] In contrast, the coupling strength in quantum

[2] In Heaviside–Lorentz units the elementary charge is $e \approx 0.303$. Often the fine structure constant α is quoted instead, which is $\alpha = e^2/4\pi \approx 0.00730$ [Moh03].

chromodynamics can be of the order of one or greater. Due to the renormalization process, its exact value depends on the method and the energy scale that is used to define it. At high energies of several GeV or more the coupling becomes small enough to apply perturbation theory ("asymptotic freedom"), but it increases towards lower energies. Thus, low-energy phenomena, like the forming of hadrons and their masses, are not accessible to perturbation theory [Bur07, p. 278].

Another strategy of quantization is the path-integral formalism [Das93, Ram89]. It was found by Richard P. Feynman on the basis of work by Paul A. M. Dirac [Fey65]. Here, the fields do not become operators. The expectation value of an observable is rather calculated via an integral over all possible classical field configurations, e.g.:

$$\langle 0 \,|\mathrm{T}\ [\psi(x)\bar{\psi}(y)]\,|\, 0\rangle = \frac{\int [\mathrm{d}A][\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\ \psi(x)\bar{\psi}(y)\ \mathrm{e}^{\mathrm{i}S[A,\bar{\psi},\psi]}}{\int [\mathrm{d}A][\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\ \mathrm{e}^{\mathrm{i}S[A,\bar{\psi},\psi]}}. \tag{2.17}$$

On the left-hand side the expectation values is expressed in terms of field *operators*, whereas the fields on the right-hand side are no operators, but *classical* fields. For simplicity, they are denoted by the same symbols here. The symbol T means that the product is to be time-ordered: If $y_0 < x_0$, it is to be read as $\psi(x)\bar{\psi}(y)$, else the other way around.

The integral on the right side is infinite-dimensional, since it means to integrate over the fields' values at every point of a continuous space-time. The field product whose expectation value is to be computed is inserted in the integrand and weighted by the exponential function of the action S times i. The denominator, without field insertions, normalizes the integral so that $\langle 0|0\rangle = 1$.

If we consider fermion fields, there is a problem: The field operators that are used to represent fermions on the left side are anti-commuting, i.e. the expectation value acquires a minus sign, if $\psi(x)$ and $\bar{\psi}(y)$ are exchanged. So, the fields in the path integral can not be ordinary complex numbers, since these would commute and give the same result independent of their order. Instead, "Graßmann numbers" are used (see appendix B), which anti-commute: $\psi(x) \cdot \bar{\psi}(y) = -\bar{\psi}(y) \cdot \psi(x)$. Apart from the fact that Graßmann multiplication is anti-commutative, most laws which hold for real and complex numbers are still valid, e.g. the laws of addition (including commutativity), associativity in multiplication and distributivity.

The path integral can be solved analytically for free theories of non-interacting particles, but when the action S includes an interaction term, this is usually not true. As in the standard, operator approach, one can evaluate the path integral perturbatively. However, the path integral formulation can also serve as a starting point for non-perturbative numerical computations. These non-perturbative calculations can provide insights that are not accessible for perturbation theory. This may be compared to the Taylor series (the "perturbative" approach) of $\exp(-1/x^2)$ around $x = 0$, which is constantly zero and does not show that the function is in fact greater than zero everywhere else. For numerical computations it is suitable to work with the field theory in Euclidean space. This is explained in the next chapter.

2.2.5. QCD in Euclidean Space

The integrand of the path integral in usual Minkowski space is strongly oscillatory because of the weight factor $\exp(\mathrm{i}S)$, and thus actually ill-defined. Therefore, it is important to damp or remove these oscillations, especially for numerical calculations. This can be done by adding a damping term to the action, or by continuing the integrals to Euclidean space [Das93, p. 73], as we will do it here. The change to Euclidean space is done by replacing the usual time by an imaginary time:

$$t \longrightarrow t' = -\mathrm{i}\tau \qquad\qquad \tau \in \mathbb{R}. \tag{2.18}$$

If we substitute the Euclidean time τ for the usual time in 4-vectors, the Minkowski metric becomes a Euclidean metric: $x^\mu_\mathrm{M}(x'_\mathrm{M})^\nu g_{\mu\nu} = tt' - \vec{x}\cdot\vec{x}' = -(\tau\tau' + \vec{x}\cdot\vec{x}') = -x^\mu_\mathrm{E}(x'_\mathrm{E})^\nu \delta_{\mu\nu}$. Other quantities are replaced accordingly:

$$\begin{aligned}
\partial_0^{(\mathrm{M})} &\longrightarrow \mathrm{i}\cdot\partial_0^{(\mathrm{E})} & \partial_j^{(\mathrm{M})} &\longrightarrow \partial_j^{(\mathrm{E})}\\
A_0^{(\mathrm{M})} &\longrightarrow \mathrm{i}\cdot A_0^{(\mathrm{E})} & A_j^{(\mathrm{M})} &\longrightarrow A_j^{(\mathrm{E})}\\
\gamma^0_\mathrm{M} &\longrightarrow -\gamma_0^{(\mathrm{E})} & \gamma^j_\mathrm{M} &\longrightarrow \mathrm{i}\cdot\gamma_j^{(\mathrm{E})}\\
\psi_\mathrm{M}(x) &\longrightarrow \gamma_0^{(\mathrm{E})}\gamma_5^{(\mathrm{E})}\cdot\psi_\mathrm{E}(x).
\end{aligned} \tag{2.19}$$

Above, Euclidean and Minkowski variables are marked by the indices E and M for clarity. Later, these indices will be dropped again. The Euclidean gamma matrices $\gamma_\mu^{(\mathrm{E})}$ fulfill the anti-commutation relation $\{\gamma_\mu^{(\mathrm{E})}, \gamma_\nu^{(\mathrm{E})}\} = 2\delta_{\mu\nu}$. As for the Minkowski gamma matrices, there are different representations for them. The one that will be used in later chapters is shown in appendix A.1.

After these replacements, the path integral can be written in the following way (the dots represent the inserted field products) [Gat10, p. 25]:

$$\int [\mathrm{d}A^{(\mathrm{M})}][\mathrm{d}\bar\psi_\mathrm{M}][\mathrm{d}\psi_\mathrm{M}]\ldots \mathrm{e}^{\mathrm{i}S_\mathrm{M}[A^{(\mathrm{M})},\bar\psi_\mathrm{M},\psi_\mathrm{M}]} \tag{2.20}$$

$$\longrightarrow \int [\mathrm{d}A^{(\mathrm{E})}][\mathrm{d}\bar\psi_\mathrm{E}][\mathrm{d}\psi_\mathrm{E}]\ldots \mathrm{e}^{-S_\mathrm{E}[A^{(\mathrm{E})},\bar\psi_\mathrm{E},\psi_\mathrm{E}]} \tag{2.21}$$

where the integration in S_E runs over Euclidean time now (the integral is "Wick-rotated"):

$$S_\mathrm{E} = \int \mathrm{d}\tau\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3\mathcal{L}_\mathrm{E} \tag{2.22}$$

with the Lagrange density

$$\mathcal{L}_\mathrm{E} = \bar\psi_\mathrm{E}(x)\cdot\left(\gamma_\mu^{(\mathrm{E})}D_\mu^{(\mathrm{E})} + m\right)\psi_\mathrm{E}(x) + \frac{1}{2g^2}\mathrm{Tr}\left[F_{\mu\nu}^{(\mathrm{E})}(x)F_{\mu\nu}^{(\mathrm{E})}(x)\right]. \tag{2.23}$$

The definitions of the covariant derivative and the field strength are as before:

$$D_\mu^{(\mathrm{E})} = \partial_\mu^{(\mathrm{E})} + \mathrm{i}A_\mu^{(\mathrm{E})}(x) \qquad\qquad F_{\mu\nu}^{(\mathrm{E})}(x) = -\mathrm{i}[D_\mu^{(\mathrm{E})}, D_\nu^{(\mathrm{E})}]. \tag{2.24}$$

In this way the oscillatory Minkowski path integral assumes the form of a partition function from statistical physics. (Sometimes it is also called the "partition function".) For greater values of S_E, the integrand falls off exponentially now.

The next section describes the evaluation of the Euclidean path integral briefly.

2.2.6. Evaluation of a Path Integral

It is not immediately clear how an integral over continuous fields is to be defined. A possible way is the following taken from [Col84, p. 8]:

1. The Minkowski path integral is replaced by its Euclidean version, as described in the previous section 2.2.5.

2. The infinite, continuous space-time is replaced by a finite lattice, which consists of only a finite number N of points and covers a finite volume V and a finite time T. A frequent choice is a hyper-cubic lattice with a lattice constant a. Also, the Euclidean action S_E must be approximated by a discretized version. Thus, the infinite-dimensional path integral becomes an integral over a finite-dimensional domain:

$$\int [\mathrm{d}\psi] \ldots \longrightarrow \int \mathrm{d}\psi(x_1) \ldots \mathrm{d}\psi(x_N) \ldots \tag{2.25}$$

   where N is the number of lattice points. For fields with several components, the integral runs over each single component.

   For quark fields, the integral is calculated according to the rules for Graßmann numbers. For gauge fields, the Haar measure is employed [Gat10, p. 44].

3. The limit of the path integral for $a \to 0$ is taken. This is called the "continuum limit". In this step, ultraviolet divergences may arise that must be taken care of by renormalization. In a nutshell, this means that we have to adjust the theory's parameters (masses, couplings, wave-function normalizations) as a approaches zero.

   Also, the limit of infinite volume $V \to \infty$ and infinite time $T \to \infty$ are taken.

4. The results are analytically continued back to Minkowski space-time. This step may not be necessary, if the relevant results (e.g. hadron masses) can be obtained directly from Euclidean expectation values.

This definition of the path integral already describes our basic strategy to compute it numerically: We will construct a hyper-cubic lattice on the computer and formulate a discrete version of QCD and the path integral on it. This "lattice QCD" is the subject of section 2.3. Before, in section 2.2.7, the "Schrödinger functional" is described, which uses a finite space-time with special boundary conditions.

2.2.7. The Schrödinger Functional

The Schrödinger functional is defined in a finite hyper-cubic space-time $T \times L_1 \times L_2 \times L_3$. The boundary conditions in the three space directions are periodic, i.e. a field f is treated as $f(x + L_k \hat{k}) = f(x)$ (with $\hat{k}$ being the unit vector in direction $k \in \{1, 2, 3\}$), but in the time direction Dirichlet boundary conditions are imposed. That means f is set to predefined constant values at $x_0 = 0$ and $x_0 = T$. The Schrödinger functional is then the propagation kernel from the "lower" field configuration at $x_0 = 0$ to the "upper" configuration [Lü92, p. 1], which can be expressed via a path integral (see (2.29)). The geometry is often visualized as a cylinder, see figure 2.1, precisely, by the lateral surface of the cylinder. In this picture, the time direction goes upwards, parallel to the axis of the cylinder. One of the space directions goes periodically around the cylinder, the other two are not shown. This illustrates the periodic boundary conditions in space, whereas the fields are set to constant values on the two circles at the top and the bottom of the cylinder. (This means only the one-dimensional boundaries of the circles, not their interior.) In analogy, a space-time with completely periodic boundary conditions could be illustrated by a torus.

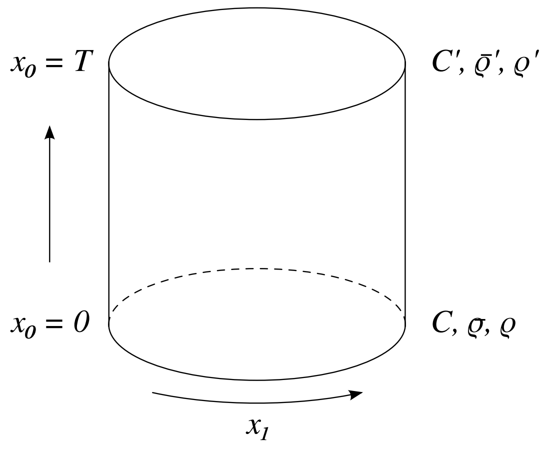

Figure 2.1.: Visualization of space-time in the Schrödinger functional

The existence of the Schrödinger representation in renormalizable quantum field theories was shown in 1981 by K. Symanzik [Sym81]. M. Lüscher et al. adapted it to gauge theories in 1992 [Lü92]. They used it to study the evolution of the renormalized coupling, which they defined by employing the intrinsic length scale of the Schrödinger functional (the size of the space-time box). And in 1993, S. Sint showed how fermions can be included in the Schrödinger functional [Sin94]. Thus, all constituents to describe the complete QCD in the Schrödinger functional were available.

The boundary values of quark fields in the Schrödinger functional are prescribed in the following way:

$$P_+ \cdot \psi(0,\vec{x}) = \rho(\vec{x}) \qquad \bar{\psi}(0,\vec{x}) \cdot P_- = \bar{\rho}(\vec{x}) \tag{2.26}$$

$$P_- \cdot \psi(T,\vec{x}) = \rho'(\vec{x}) \qquad \bar{\psi}(T,\vec{x}) \cdot P_+ = \bar{\rho}'(\vec{x}). \tag{2.27}$$

The projectors $P_\pm = (1 \pm \gamma_0)/2$ project to a two-dimensional sub-space of Dirac-spinor space, so, effectively, we fix only half the components of the fields at each boundary. This is necessary, because the Dirac equation is of first order, and the problem would be over-constrained, if we fixed all components of the quark fields. Usually, the boundary fields ρ, ρ', $\bar{\rho}$, $\bar{\rho}'$ are set to zero.

Similarly for the gauge fields, only the spatial components are fixed:

$$A_k(0,\vec{x}) = C_k(\vec{x}) \qquad A_k(T,\vec{x}) = C'_k(\vec{x}) \tag{2.28}$$

where $C_k(\vec{x})$ and $C'_k(\vec{x})$ must be Hermitian matrices.

Now, the Schrödinger functional can be represented as the "partition function" $\mathcal{Z}$, which depends on these boundary fields:

$$\mathcal{Z}[C, C', \rho, \bar{\rho}, \rho', \bar{\rho}'] = N \cdot \int_{\text{fields}} \mathrm{e}^{-S}. \tag{2.29}$$

The integral runs over the quark and gauge fields in the interior of the Schrödinger-functional cylinder, the boundary values stay fixed. The factor N is the normalization.

We can add two more terms including external spinor fields $\eta(x)$ and $\bar{\eta}(x)$ to the action (as in [Som95, p. 1]):

$$\mathcal{Z}[C, C', \rho, \bar{\rho}, \rho', \bar{\rho}', \eta, \bar{\eta}] = N \cdot \int_{\text{fields}} \mathrm{e}^{-S - \int \mathrm{d}^4x (\bar{\psi}(x)\cdot\eta(x) + \bar{\eta}(x)\cdot\psi(x))}. \tag{2.30}$$

This enables us to define correlation functions as functional derivatives of $\mathcal{Z}$ with respect to the external fields [Fri09, p. 46], e.g. the derivatives with respect to η and $\bar{\eta}$ insert the quark fields $\bar{\psi}$ and ψ into the path integral:

$$\langle \psi(x)\bar{\psi}(y) \rangle = \left[-\frac{1}{\mathcal{Z}} \cdot \frac{\delta}{\delta\bar{\eta}(x)} \frac{\delta}{\delta\eta(y)} \mathcal{Z} \right]_{\eta,\bar{\eta},\rho,\bar{\rho},\rho',\bar{\rho}'=0} \tag{2.31}$$

$$= \left[\frac{1}{\mathcal{Z}} \cdot \int_{\text{fields}} \psi(x)\bar{\psi}(y) \cdot \mathrm{e}^{-S} \right]_{\eta,\bar{\eta},\rho,\bar{\rho},\rho',\bar{\rho}'=0}. \tag{2.32}$$

(In writing $\langle\psi(x)\bar{\psi}(y)\rangle$, the initial and final field configurations as well as the time-ordering symbol were omitted.)

This is also possible for the derivatives with respect to the boundary fields ρ, ρ' etc. They are formally identified with the fields ζ, $\bar{\zeta}$, ζ' and $\bar{\zeta}'$:

$$\frac{\delta}{\delta\bar{\eta}(x)} \to \psi(x) \qquad \frac{\delta}{\delta\bar{\rho}(\vec{x})} \to \zeta(\vec{x}) \qquad \frac{\delta}{\delta\bar{\rho}'(\vec{x})} \to \zeta'(\vec{x}) \tag{2.33}$$

$$\frac{\delta}{\delta\eta(x)} \to -\bar{\psi}(x) \qquad \frac{\delta}{\delta\rho(\vec{x})} \to -\bar{\zeta}(\vec{x}) \qquad \frac{\delta}{\delta\rho'(\vec{x})} \to -\bar{\zeta}'(\vec{x}). \tag{2.34}$$

The field products that are inserted by the ζ-fields will be spelled out in section 2.3.3.

2.3. QCD on a Lattice

The theory of quantum chromodynamics in a continuous space-time can not be formulated directly, because it gives divergent results for observable quantities. First, one has to introduce a cut-off in some way which makes the theory's results finite. This procedure is called “regularization” (see [Col84], especially p. 13). There are several methods for regularization. We will replace the continuous space-time by a discrete lattice. In this way, we can compute the path integral non-perturbatively on a computer.

In the following sections, the lattice geometry, the fields living on it and the discretized action that we use are described.

2.3.1. Geometry

We use a four-dimensional, hyper-cubic, Euclidean lattice with Schrödinger-functional boundary conditions for our calculations (see section 2.2.7). For simplicity the lattice spacing a is set to one, i.e. all lengths, times, energies etc. are measured in “lattice units”.

The extents of the lattice in the three space directions are L_1, L_2 and L_3, and time runs from from 0 to T. That means a lattice point x is characterized by four integer coordinates: the time x_0 and the space coordinates x_1, x_2, x_3 with

$$0 \leq x_0 \leq T \qquad\qquad 0 \leq x_k < L_k \text{ for } k = 1, 2, 3. \tag{2.35}$$

Note that $x_0 = T$ is also included.

The unit vectors in the four directions are denoted as the number of the direction with a hat above it: $\hat{0}, \dots, \hat{3}$. Lower Greek indices $(\mu, \nu, \dots)$ stand for all directions and may assume the values 0, 1, 2, 3, whereas lower Latin indices stand only for the space directions 1, 2, 3.

The total number of lattice sites, the volume V, is

$$V = (T + 1) \cdot L_1 \cdot L_2 \cdot L_3, \tag{2.36}$$

the bulk volume V_B is the number of lattice sites which are not part of a boundary, i.e. with $0 < x_0 < T$:

$$V_\mathrm{B} = (T - 1) \cdot L_1 \cdot L_2 \cdot L_3, \tag{2.37}$$

and the three-dimensional volume V_3 of one time slice (i.e. with constant x_0) is

$$V_3 = L_1 \cdot L_2 \cdot L_3. \tag{2.38}$$

2.3.2. Fields

The two important types of fields are the gauge field and the quark fields. Especially the gauge field is represented in a way different from the one in continuous space.

The quark fields The quarks fields $\psi(x)$ are only defined at the discrete points of the lattice, but they keep their Dirac and color structure, i.e. they are still Dirac spinors and carry a spinor index $A \in \{1, \dots, 4\}$, and they are still three-dimensional vectors in color space with a color index $\alpha \in \{1, 2, 3\}$: $\psi(x)_{A\alpha}$. They may have a flavor index, too. A gauge transformation $\Omega(x)$ transforms the quark fields as

$$\psi(x) = \Omega(x) \cdot \psi'(x) \qquad \bar{\psi}(x) = \bar{\psi}'(x) \cdot \Omega^\dagger(x). \tag{2.39}$$

(The gauge transformation must act only on the fields in the bulk, not on the boundary fields at $x_0 = 0, T$, see section 2.3.3.)

The gauge field The gauge field serves as a connection for color vectors at different space-time points (see section 2.2.2). In the continuum, these points are only infinitesimally distant from each other, but on the lattice, the distance between two neighbor points is $a = 1$ and thus finite. Therefore, the continuum gauge field $A_\mu(x)$ is replaced by an integrated version $U_\mu(x)$. Both are connected via (see [Gat10, p. 34])

$$U_\mu(x) = \mathrm{P}\left[\mathrm{e}^{\mathrm{i}\int_x^{x+\hat{\mu}} \mathrm{d}y_\nu A_\nu(y)}\right] \tag{2.40}$$

where the integral runs over a straight path connecting the lattice site x with its neighbor $x + \hat{\mu}$. The symbol P means that the products in the expansion of the exponential function are path-ordered, i.e. fields closer to x are put on the left side.

The field $U_\mu(x)$ is an element from the gauge group SU(3) and not from its Lie algebra like $A_\mu(x)$. But like $A_\mu(x)$ it can be represented by a complex 3×3-matrix, so it has two color indices: $U_\mu(x)_{\alpha\beta}$. It can be thought of as living in between the two lattice sites which are connected by it and is therefore also called a “link”. Under a gauge transformation $\Omega(x)$, it transforms as

$$U_\mu(x) = \Omega(x) \cdot U'_\mu(x) \cdot \Omega^\dagger(x + \hat{\mu}). \tag{2.41}$$

So, gauge-invariant expressions can be formed by connecting spinors at different points by a path of $U_\mu(x)$, e.g. $\bar{\psi}(x) \cdot U_1(x) \cdot U_3(x + \hat{1}) \cdot \psi(x + \hat{1} + \hat{3})$ is gauge-invariant.

2.3.3. Boundary Conditions

Often, completely periodic boundary conditions are applied on the lattice, but we will use the boundary conditions of the Schrödinger functional, which have already been described in section 2.2.7 and are repeated here:

In space-like directions periodic boundary conditions are applied:

$$f(x + \hat{k}) = f(x) \text{ with } k = 1, 2, 3. \tag{2.42}$$

For spinor fields, one may add a phase factor $\mathrm{e}^{\mathrm{i}\theta_k}$ in this equation. This shifts the possible momenta on the lattice by $\vec{\theta}/L$. But here we will include this phase into the Dirac operator instead of the boundary conditions (see section 2.3.4).

In time direction, Dirichlet boundary conditions are used, i.e. the fields have predefined values at $x_0 = 0$ and $x_0 = T$. For the quark fields, we can only fix half the components, because the Dirac equation is a first-order differential equation. This is done via the projectors $P_\pm = (1 \pm \gamma_0)/2$. They fulfill

$$P_+ \cdot P_- = P_- \cdot P_+ = 0 \qquad\qquad P_+ + P_- = 1. \tag{2.43}$$

Then, the boundary conditions for quark fields are

$$P_+ \cdot \psi(0,\vec{x}) = \rho(\vec{x}) \qquad\qquad \bar{\psi}(0,\vec{x}) \cdot P_- = \bar{\rho}(\vec{x}) \tag{2.44}$$
$$P_- \cdot \psi(T,\vec{x}) = \rho'(\vec{x}) \qquad\qquad \bar{\psi}(T,\vec{x}) \cdot P_+ = \bar{\rho}'(\vec{x}). \tag{2.45}$$

In the following, we will set the boundary fields to zero:

$$\rho \equiv \rho' \equiv \bar{\rho} \equiv \bar{\rho}' \equiv 0. \tag{2.46}$$

However, their derivatives (acting on $\exp(-S)$ in the path integral) can still be used to construct observables. In section 2.2.7, they were formally identified with the fields ζ, ζ', $\bar{\zeta}$ and $\bar{\zeta}'$. If we use the action S described in section 2.3.4, these derivatives effectively insert the field products:

$$\zeta(\vec{x}) = -\frac{\delta S}{\delta \bar{\rho}(\vec{x})} = \tilde{c}_\mathrm{t} \cdot P_- \cdot U_0(0,\vec{x}) \cdot \psi(1,\vec{x}) \tag{2.47}$$
$$\zeta'(\vec{x}) = -\frac{\delta S}{\delta \bar{\rho}'(\vec{x})} = \tilde{c}_\mathrm{t} \cdot P_+ \cdot U_0(T-1,\vec{x})^\dagger \cdot \psi(T-1,\vec{x}) \tag{2.48}$$
$$\bar{\zeta}(\vec{x}) = \frac{\delta S}{\delta \rho(\vec{x})} = \tilde{c}_\mathrm{t} \cdot \bar{\psi}(1,\vec{x}) \cdot U_0(0,\vec{x})^\dagger \cdot P_+ \tag{2.49}$$
$$\bar{\zeta}'(\vec{x}) = \frac{\delta S}{\delta \rho'(\vec{x})} = \tilde{c}_\mathrm{t} \cdot \bar{\psi}(T-1,\vec{x}) \cdot U_0(T-1,\vec{x}) \cdot P_-. \tag{2.50}$$

The Dirichlet boundary conditions for the gauge field U can be expressed as

$$U_k(0,\vec{x}) = \exp(\mathrm{i}C_k(\vec{x})) \qquad\qquad U_k(T,\vec{x}) = \exp(\mathrm{i}C_k'(\vec{x})), \tag{2.51}$$

where $C_k(\vec{x})$ and $C_k'(\vec{x})$ are two Hermitian, traceless matrices and $k = 1, 2, 3$.

2.3.4. Action

We use the lattice action as specified in [Som95, Lü97b]. As in the continuum (see (2.23)), the action S decomposes into a pure gauge-field part and a fermionic part, which still depends on the gauge field:

$$S[U,\bar{\psi},\psi] = S_\mathrm{G}[U] + S_\mathrm{F}[U,\bar{\psi},\psi]. \tag{2.52}$$

Gauge action We will not need the gauge action S_G in the following chapters, because we will consider only fermionic expectation values (see sections 2.3.5 and 4), but, for completeness, it is included here:

$$S_\mathrm{G} = \frac{1}{g_0^2} \sum_p w_p \cdot \mathrm{Tr}\,(1 - U_p). \tag{2.53}$$

This expression involves a sum over plaquettes p. A plaquette is a 1×1-loop on the lattice, so it can be characterized by its starting point x and two different directions $\mu, \nu \in \{0, \dots, 3\}$. And p can be seen as a shorthand for $\{x, \mu, \nu\}$. The U_p are defined as:

$$U_p = U_{\{x,\mu,\nu\}} = U_\mu(x) U_\nu(x + \hat{\mu}) U_\mu(x + \hat{\nu})^\dagger U_\nu(x)^\dagger. \tag{2.54}$$

It can be shown that S_G converges to the continuum action $1/2 \cdot \mathrm{Tr}\,(F_{\mu\nu} F_{\mu\nu})$ in the "naive continuum limit" $a \to 0$, see [Gat10, p. 37]. ("Naive" means that the dependence of the coupling g_0 and other quantities on the lattice spacing a is not taken into account.)

For $\mathcal{O}(a)$ improvement a weight factor w_p has to be included in the action. Its values are

$$w_p = \begin{cases} 1 & \text{for plaquettes in the bulk} \\ c_\mathrm{t} & \text{for time-like plaquettes touching a boundary.} \end{cases} \tag{2.55}$$

According to [Bod00, p. 8] we choose

$$c_\mathrm{t} = 1 + g_0^2 \cdot (-0.089 + N_\mathrm{f} \cdot 0.019141) + g_0^4 \cdot (-0.03) \tag{2.56}$$

where N_f is the number of dynamical flavors (see section 2.3.5).

Fermionic action The fermionic action S_F on the lattice is

$$S_\mathrm{F} = \sum_x \bar{\psi}(x)(D + \delta D + m_0)\psi(x) \qquad (1 \le x_0 \le T - 1), \tag{2.57}$$

where D is the unimproved Wilson–Dirac operator, δD contains the $\mathcal{O}(a)$ improvement terms[3] and m_0 is the bare quark mass.

The Wilson–Dirac operator is defined as:

$$D = \gamma_\mu \widetilde{\nabla}_\mu - \frac{1}{2} \cdot \nabla_\mu^* \nabla_\mu. \tag{2.58}$$

with the forward lattice derivative

$$\nabla_\mu \psi(x) = \lambda_\mu U_\mu(x) \psi(x + \hat{\mu}) - \psi(x), \tag{2.59}$$

[3]These "improve" the convergence to the continuum limit $a \to 0$.

the backward derivative:

$$\nabla^*_\mu \psi(x) = \psi(x) - \lambda^*_\mu U_\mu(x-\hat{\mu})^\dagger \psi(x-\hat{\mu}) \tag{2.60}$$

and the symmetric derivative:

$$\begin{aligned}\widetilde{\nabla}_\mu \psi(x) &= \frac{1}{2}\Big(\lambda_\mu U_\mu(x)\psi(x+\hat{\mu}) - \lambda^*_\mu U_\mu(x-\hat{\mu})^\dagger \psi(x-\hat{\mu})\Big) \qquad (2.61)\\ &= \frac{1}{2}\left(\nabla_\mu + \nabla^*_\mu\right)\psi(x). \qquad (2.62)\end{aligned}$$

These are three ways to discretize the continuum covariant derivative D_μ, but we have included additional phases λ_μ, which can be expressed as

$$\lambda_0 = 1 \qquad \text{and} \qquad \lambda_j = \mathrm{e}^{i\theta_j/L_j} \text{ for } j = 1,2,3. \tag{2.63}$$

They effectively shift the possible momenta on the lattice by θ_j/L_j. Instead of the derivatives, they could have been included in the boundary conditions for quark fields, too (see section 2.3.3). Often, isotropic θ-angles are chosen, i.e. $\theta_1 = \theta_2 = \theta_3$, but here also anisotropic θ-angles are explicitly allowed. They may depend on the quark flavor, too.

Coming back to the Wilson–Dirac operator (2.58), the first term $\gamma_\mu \widetilde{\nabla}_\mu$ is a straight-forward discretization of the continuum expression $\gamma_\mu D_\mu$ using the symmetric lattice derivative. The second term $-1/2 \cdot \nabla^*_\mu \nabla_\mu$ is the "Wilson term". It is included to remove unphysical solutions of the Dirac equation that would occur on the lattice otherwise, the so-called "doublers" [Gat10, p. 110]. If the lattice spacing a is introduced again, it can be seen that the Wilson term is proportional to a times a second-order derivative, and thus it vanishes in the naive continuum limit for $a \to 0$.

The improvement operator δD can be split up into a volume and a boundary term:

$$\delta D = \delta D_\mathrm{V} + \delta D_\mathrm{B}. \tag{2.64}$$

The volume term (also referred to as Sheikholeslami–Wohlert or clover term) is

$$\delta D_\mathrm{V} \psi(x) = c_\mathrm{sw} \frac{i}{4} \sigma_{\mu\nu} F_{\mu\nu}(x)\psi(x). \tag{2.65}$$

According to [Lü97a, p. 16] and [Jan98, p. 7], we set the coefficient c_sw to the value

$$c_\mathrm{sw} = \frac{1.0 - 0.656 \cdot g_0^2 + 0.152 \cdot g_0^4 + 0.054 \cdot g_0^6}{1.0 - 0.922 \cdot g_0^2} \qquad \text{for } N_\mathrm{f} = 0 \tag{2.66}$$

and

$$c_\mathrm{sw} = \frac{1 - 0.454 \cdot g_0^2 - 0.175 \cdot g_0^4 + 0.012 \cdot g_0^6 + 0.045 \cdot g_0^8}{1.0 - 0.720 \cdot g_0^2} \qquad \text{for } N_\mathrm{f} = 2. \tag{2.67}$$

$F_{\mu\nu}(x)$ is the lattice field strength tensor, which is given by

$$F_{\mu\nu}(x) = \frac{Q_{\mu\nu}(x) - (Q_{\mu\nu}(x))^\dagger}{8} \tag{2.68}$$

with

$$\begin{aligned} Q_{\mu\nu}(x) = {} & U_\mu(x)U_\nu(x+\hat\mu)U_\mu(x+\hat\nu)^\dagger U_\nu(x)^\dagger \\ & + U_\nu(x-\hat\nu)^\dagger U_\mu(x-\hat\nu)U_\nu(x+\hat\mu-\hat\nu)U_\mu(x)^\dagger \\ & + U_\nu(x)U_\mu(x-\hat\mu+\hat\nu)^\dagger U_\nu(x-\hat\mu)^\dagger U_\mu(x-\hat\mu) \\ & + U_\mu(x-\hat\mu)^\dagger U_\nu(x-\hat\mu-\hat\nu)^\dagger U_\mu(x-\hat\mu-\hat\nu)U_\nu(x-\hat\nu). \end{aligned} \tag{2.69}$$

(Note that this definition of $F_{\mu\nu}$ does not coincide with the definition of the continuum field-strength tensor in section 2.2.2. They differ by a factor i, so the components $F_{\mu\nu}$ of the tensor are anti-Hermitian here.)

The boundary term δD_B effectively modifies the weight of the derivatives at $x_0 = 1$ and $x_0 = T-1$. In analogy to the gauge-action improvement it is characterized by a coefficient $\tilde c_\mathrm{t}$:

$$\begin{aligned} \delta D_\mathrm{B}\psi(x) = {} & (\tilde c_\mathrm{t} - 1)\cdot\Big[\delta_{x_0,1}\Big(\psi(x) - U_0(x-\hat 0)^\dagger P_+\psi(x-\hat 0)\Big) \\ & + \delta_{x_0,T-1}\left(\psi(x) - U_0(x)P_-\psi(x+\hat 0)\right)\Big]. \end{aligned} \tag{2.70}$$

For vanishing boundary fields, this reduces to

$$\delta D_\mathrm{B}\psi(x) = (\tilde c_\mathrm{t} - 1)\cdot(\delta_{x_0,1} + \delta_{x_0,T-1})\cdot\psi(x). \tag{2.71}$$

The improvement coefficient $\tilde c_\mathrm{t}$ is chosen after [Lü96a, p. 26] as

$$\tilde c_\mathrm{t} = 1 - 0.018\cdot g_0^2. \tag{2.72}$$

The operator D is not Hermitian. Instead, it obeys the relation $\gamma_5 D\gamma_5 = D^\dagger$. This property is called "γ_5-Hermiticity". However, often it is useful to work with a Hermitian operator Q (not to be confused with $Q_{\mu\nu}(x)$ from (2.69)), e.g. the standard conjugate-gradient method relies on an Hermitian operator (see section 4.4.1). Q is defined as

$$Q = \gamma_5(D + \delta D + m_0). \tag{2.73}$$

Explicitly, this means (for vanishing boundary fields)

$$\begin{aligned} Q\psi(x) = \gamma_5\cdot\Bigg\{ & \frac{1}{2\kappa}\psi(x) - \sum_{\mu=0}^{3}\Big[\lambda_\mu U_\mu(x)\frac{1-\gamma_\mu}{2}\psi(x+\hat\mu) \\ & + \lambda_\mu^* U_\mu(x-\hat\mu)^\dagger(1+\gamma_\mu)\psi(x-\hat\mu)\Big] \\ & + c_\mathrm{sw}\frac{i}{4}\sigma_{\mu\nu}F_{\mu\nu}(x)\psi(x) + (\tilde c_\mathrm{t} - 1)\cdot(\delta_{x_0,1} + \delta_{x_0,T-1})\cdot\psi(x)\Bigg\}. \end{aligned} \tag{2.74}$$

In the above equation, the parameter κ has been substituted for the mass m_0. Their relation is

$$\kappa = \frac{1}{2(4+m_0)}. \tag{2.75}$$

2.3.5. Splitting into Fermionic and Gauge Expectation Value

With the lattice action $S[U,\bar{\psi},\psi]$, we can now calculate the expectation value of an observable $\mathcal{O}$ as described in section 2.2.4:

$$\langle \mathcal{O} \rangle = \frac{\int [\mathrm{d}U][\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\, \mathcal{O}\, \mathrm{e}^{-S_{\mathrm{G}}[U]-S_{\mathrm{F}}[U,\bar{\psi},\psi]}}{\int [\mathrm{d}U][\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\, \mathrm{e}^{-S_{\mathrm{G}}[U]-S_{\mathrm{F}}[U,\bar{\psi},\psi]}} . \tag{2.76}$$

The integrals over the gauge field and over the fermionic fields will be evaluated by different means. Therefore, we will split the expectation value into a gauge and a fermionic expectation value. The fermionic integrals can be computed via the Wick theorem (see appendix B and [Gat10, p. 109]). The result for the integral from the denominator is

$$\int [\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\, \mathrm{e}^{-S_{\mathrm{F}}[U,\bar{\psi},\psi]} = \det\left(D + \delta D + m_0\right). \tag{2.77}$$

So, we can write (see also [Fri09, p. 46])

$$\langle \mathcal{O} \rangle = \big\langle \langle \mathcal{O} \rangle_{\mathrm{F}} \big\rangle_{\mathrm{G}} \tag{2.78}$$

with

$$\langle \mathcal{O} \rangle_{\mathrm{F}} = \frac{\int [\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\, \mathcal{O}\, \mathrm{e}^{-S_{\mathrm{F}}[U,\bar{\psi},\psi]}}{\int [\mathrm{d}\bar{\psi}][\mathrm{d}\psi]\, \mathrm{e}^{-S_{\mathrm{F}}[U,\bar{\psi},\psi]}} \tag{2.79}$$

$$\big\langle \langle \mathcal{O} \rangle_{\mathrm{F}} \big\rangle_{\mathrm{G}} = \frac{\int [\mathrm{d}U]\, \langle \mathcal{O} \rangle_{\mathrm{F}}\, \mathrm{e}^{-S_{\mathrm{G}}[U]+\ln\det(D+\delta D+m_0)}}{\int [\mathrm{d}U]\, \mathrm{e}^{-S_{\mathrm{G}}[U]+\ln\det(D+\delta D+m_0)}} . \tag{2.80}$$

We will treat only the computation of the fermionic expectation value for a given gauge field here. The integration over the gauge field can be done by Monte-Carlo algorithms. A summary of them can be found in [Rot05, p. 284] and [Gat10, p. 74, 190]. These algorithms create gauge-field configurations according to the probability distribution $\exp(-S_{\mathrm{G}}[U] + \ln\det(D + \delta D + m_0))$. Especially the computation of the determinant of the Dirac operator is very laborious. In the "quenched" approximation, the determinant is assumed to be one and omitted. This corresponds to not including dynamical sea quarks, i.e. the number of dynamical flavors is $N_{\mathrm{f}} = 0$. In an $N_{\mathrm{f}} = 2$ simulation, the determinant is computed for the Dirac operator of the two lightest quarks, up and down.

2.3.6. Propagators

As it was pointed out in the last section, the integration over the fermionic, Graßmann-valued fields is performed by means of Wick's theorem, more precisely, it allows us to write a fermionic expectation value in terms of the inverse Dirac operator $(D + \delta D + m_0)^{-1}$. This quantity is called the propagator $S(x, y)$. It yields the expectation value $\langle \psi(x)\bar{\psi}(y)\rangle_{\mathrm{F}}$, from which all other expectation values can be deduced.

Below we will also define propagators for more specific field combinations, e.g. $\bar{S}(x)$. The definitions are based on [Som95] and [Hof05, p. 107] with different prefactors. Section 2.3.7 will summarize which fermionic expectation values the different propagators correspond to, and Section 2.3.8 will show how to express correlation functions in terms of these propagators.

The quark propagator $S(x,y)$ is the inverse of the Dirac operator $(D+\delta D+m_0)^{-1}$. Thus, it is a matrix in Dirac-spinor space, color space and space-time (i.e. it depends on two space-time arguments x and y) like the Dirac operator itself. (It also has a flavor index, since it depends on the mass m_0, but this index will be left out here.) $S(x,y)$ can be obtained as the solution of the Dirac equation with the constant gauge field U and a point-like source in the bulk at y:

$$(D+\delta D+m_0)S(x,y)=\delta_{x,y}. \tag{2.81}$$

Furthermore S has to respect the boundary conditions for fermionic fields. It is treated as periodic in the three space directions. At the time-like boundaries it must fulfill:

$$P_+S(x,y)|_{x_0=0}=\rho(\vec{x})=0 \qquad\qquad P_-S(x,y)|_{x_0=T}=\rho'(\vec{x})=0. \tag{2.82}$$

(2.81) can also be written in terms of the Hermitian matrix Q:

$$QS(x,y)=\gamma_5\cdot\delta_{x,y}. \tag{2.83}$$

Next, we define the propagator $\bar{S}(x)$ from the lower boundary to the point x via

$$Q\bar{S}(x)=\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}}\cdot\delta_{x_0,1}\cdot U_0(0,\vec{x})^\dagger\cdot\gamma_5P_+. \tag{2.84}$$

It can also be expressed in terms of the previously defined "point-to-point" propagator $S(x,y)$. We sum equation (2.83) over all points y with $y_0=1$ and multiply it by the appropriate prefactors and matrices, so that the right-hand side becomes identical to the right-hand side of (2.84):

$$\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}}\cdot\sum_{\vec{y}}QS(x,(1,\vec{y}))\cdot U_0(0,\vec{y})^\dagger\cdot P_+=\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}}\cdot\sum_{\vec{y}}\gamma_5\cdot\delta_{x,(1,\vec{y})}\cdot U_0(0,\vec{y})^\dagger\cdot P_+, \tag{2.85}$$

using the linearity of Q and the δ-factor on the right-hand side:

$$Q\cdot\left[\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}}\cdot\sum_{\vec{y}}S(x,(1,\vec{y}))\cdot U_0(0,\vec{y})^\dagger\cdot P_+\right]=\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}}\cdot\delta_{x_0,1}\cdot U_0(0,\vec{x})^\dagger\cdot\gamma_5P_+. \tag{2.86}$$

We can conclude that the boundary-to-bulk propagator can be written as

$$\bar{S}(x)=\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}}\cdot\sum_{\vec{y}}S(x,(1,\vec{y}))\cdot U_0(0,\vec{y})^\dagger\cdot P_+. \tag{2.87}$$

Likewise, we define the propagator $\bar{R}(x)$ from the upper boundary to the point x by:

$$Q\bar{R}(x) = \frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}} \cdot \delta_{x_0,T-1} \cdot U_0(T-1,\vec{x}) \cdot \gamma_5 P_- \tag{2.88}$$

$$\Longrightarrow \qquad \bar{R}(x) = \frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}} \cdot \sum_{\vec{y}} S(x,(T-1,\vec{y})) \cdot U_0(T-1,\vec{y}) \cdot P_- \tag{2.89}$$

and the boundary-to-boundary propagator $\bar{S}_T$:

$$\bar{S}_T = -\frac{\tilde{c}_\mathrm{t}}{\sqrt{V_3}} \sum_{\vec{x}} U_0(T-1,\vec{x})^\dagger \cdot P_+ \cdot \bar{S}(T-1,\vec{x}). \tag{2.90}$$

($\bar{S}_T$ can also be expressed via $\bar{R}(x)$ and $S(x,y)$.)

2.3.7. Summary of Contractions

The propagators from the last section can be used to express the basic correlations of two field operators via the Wick theorem [Gat10, p. 109], e.g.

$$\langle \psi(x)\bar{\psi}(y)\rangle_\mathrm{F} = (D + \delta D + m_0)^{-1}(x,y) = S(x,y). \tag{2.91}$$

In particular, we will need correlations with a flat-wave-function[4] source like $1/\sqrt{V_3} \cdot \sum_{\vec{x}} \bar{\zeta}(\vec{x})$. (The sum over $\vec{x}$ projects to states with momenta $\vec{\theta}/L$.) They are summarized in the following equations:

$$\langle \psi(x)\bar{\psi}(y)\rangle_\mathrm{F} = S(x,y) \tag{2.92}$$

$$\frac{1}{\sqrt{V_3}} \sum_{\vec{y}} \langle \psi(x)\bar{\zeta}(\vec{y})\rangle_\mathrm{F} = \bar{S}(x) \tag{2.93}$$

$$\frac{1}{\sqrt{V_3}} \sum_{\vec{y}} \langle \psi(x)\bar{\zeta}'(\vec{y})\rangle_\mathrm{F} = \bar{R}(x) \tag{2.94}$$

$$\frac{1}{\sqrt{V_3}} \sum_{\vec{x}} \langle \zeta(\vec{x})\bar{\psi}(y)\rangle_\mathrm{F} = \gamma_5 \bar{S}^\dagger(y)\gamma_5 \tag{2.95}$$

$$\frac{1}{\sqrt{V_3}} \sum_{\vec{x}} \langle \zeta'(\vec{x})\bar{\psi}(y)\rangle_\mathrm{F} = \gamma_5 \bar{R}^\dagger(y)\gamma_5 \tag{2.96}$$

$$\frac{1}{V_3} \sum_{\vec{x}\vec{y}} \langle \zeta(\vec{x})\bar{\zeta}'(\vec{y})\rangle_\mathrm{F} = -\gamma_5 \bar{S}_T^\dagger \gamma_5 \tag{2.97}$$

$$\frac{1}{V_3} \sum_{\vec{x}\vec{y}} \langle \zeta'(\vec{x})\bar{\zeta}(\vec{y})\rangle_\mathrm{F} = -\bar{S}_T. \tag{2.98}$$

[4] "Flat wave function" means that we will simply sum over the boundary fields in correlation functions, e.g. $\sum_{\vec{y}\vec{z}} \bar{\zeta}(\vec{y})\gamma_5\zeta(\vec{z})$ (see equation 2.103). Instead, one could also include a non-trivial wave function ω, as in $\sum_{\vec{y}\vec{z}} \bar{\zeta}(\vec{y})\gamma_5\zeta(\vec{z}) \cdot \omega(\vec{y}-\vec{z})$. This can improve the overlap with the energy state under investigation. A flat wave function corresponds to choosing $\omega(x) = \text{const}$.

2.3.8. Correlation Functions

With the help of these relations, we can write the correlation functions in which we are interested in terms of the propagators. These correlation functions are expectation values of different combinations of the following field products:

$$\text{the vector current:} \quad V_\mu^{ji}(x) = \bar{\psi}^j(x)\gamma_\mu\psi^i(x) \tag{2.99}$$

$$\text{the axial current:} \quad A_\mu^{ji}(x) = \bar{\psi}^j(x)\gamma_\mu\gamma_5\psi^i(x) \tag{2.100}$$

$$\text{the pseudo-scalar density:} \quad P^{ji}(x) = \bar{\psi}^j(x)\gamma_5\psi^i(x) \tag{2.101}$$

$$\text{the tensor:} \quad T_{\mu\nu}^{ji}(x) = \bar{\psi}^j(x)i\sigma_{\mu\nu}\psi^i(x) \tag{2.102}$$

and sources at the boundaries:

$$\text{lower pseudo-scalar source:} \quad \mathcal{O}^{ij} = \frac{1}{V_3}\sum_{\vec{y}\vec{z}}\bar{\zeta}^i(\vec{y})\gamma_5\zeta^j(\vec{z}) \tag{2.103}$$

$$\text{upper pseudo-scalar source:} \quad \mathcal{O}'^{ij} = \frac{1}{V_3}\sum_{\vec{y}\vec{z}}\bar{\zeta}'^i(\vec{y})\gamma_5\zeta'^j(\vec{z}) \tag{2.104}$$

$$\text{lower vector source:} \quad \mathcal{V}_k^{ij} = \frac{1}{V_3}\sum_{\vec{y}\vec{z}}\bar{\zeta}^i(\vec{y})\gamma_k\zeta^j(\vec{z}) \tag{2.105}$$

$$\text{upper vector source:} \quad \mathcal{V}_k'^{ij} = \frac{1}{V_3}\sum_{\vec{y}\vec{z}}\bar{\zeta}'^i(\vec{y})\gamma_k\zeta'^j(\vec{z}). \tag{2.106}$$

(The upper right indices are flavor indices, which will be given explicitly now.)

Now we can build correlation functions from these field products. As an example, we consider

$$\begin{aligned} f_{\mathrm{A}}^{ij}(x_0) &= -\frac{1}{2V_3}\sum_{\vec{x}}\left\langle A_0^{ji}(x)\cdot\mathcal{O}^{ij}\right\rangle & (2.107)\\ &= -\frac{1}{2V_3}\sum_{\vec{x}\vec{y}\vec{z}}\left\langle\bar{\psi}^j(x)\gamma_0\gamma_5\psi^i(x)\bar{\zeta}^i(\vec{y})\gamma_5\zeta^j(\vec{z})\right\rangle. & (2.108)\end{aligned}$$

To compute the fermionic part of the expectation value, we apply the Wick theorem and contract the fields $\psi^i(x)$ and $\bar{\zeta}^i(\vec{y})$, as well as $\bar{\psi}^j(x)$ and $\zeta^j(\vec{z})$:

$$f_{\mathrm{A}}^{ij}(x_0) = \frac{1}{2V_3}\sum_{\vec{x}\vec{y}\vec{z}}\left\langle\mathrm{Tr}\left(\left\langle\zeta^j(\vec{z})\bar{\psi}^j(x)\right\rangle_{\mathrm{F}}\cdot\gamma_0\gamma_5\cdot\left\langle\psi^i(x)\bar{\zeta}^i(\vec{y})\right\rangle_{\mathrm{F}}\cdot\gamma_5\right)\right\rangle_{\mathrm{G}}. \tag{2.109}$$

The minus sign disappears because of the anti-commutation of the Graßmann fields. With the relations from section 2.3.7, the two-field contractions can be expressed in terms of propagators:

$$f_{\mathrm{A}}^{ij}(x_0) = \frac{1}{2}\sum_{\vec{x}}\left\langle\mathrm{Tr}\left(\gamma_5\bar{S}^j(x)^\dagger\gamma_5\cdot\gamma_0\gamma_5\cdot\bar{S}^i(x)\cdot\gamma_5\right)\right\rangle_{\mathrm{G}}. \tag{2.110}$$

Using the cyclic property of the trace and the properties of the gamma matrices, we arrive at

$$f_{\mathrm{A}}^{ij}(x_0) = -\frac{1}{2}\sum_{\vec{x}} \left\langle \mathrm{Tr}\,\left(\bar{S}^j(x)^\dagger \gamma_0 \bar{S}^i(x)\right)\right\rangle_{\mathrm{G}} . \tag{2.111}$$

This is the expression that we are to evaluate on the computer (see chapter 2.3.8, in particular section 5.3).

The following equations contain the definitions of all QCD correlation functions that we consider. For each correlation, the first line shows its definition in terms of field products, which is rewritten in terms of the propagators in the second line. For the vector correlations k_{V}, k_{T} etc. also the expansion in Dirac components of the propagators is shown, because two thirds of the terms cancel, when the sum over the index $k \in \{1,2,3\}$ of the gamma matrices is performed. So, the explicit computation of the sum as shown in the second line would need circa three times as many operations. Therefore, the routines `k_v_rel` etc. which are described in section 5.3 employ the expression from the third and following lines with explicit reference to the propagators' Dirac components. Also, the relations $\bar{S}\cdot P_- = \bar{R}\cdot P_+ = 0$ have been used (see section 5.3.2).

$$f_{\mathrm{A}}^{ij}(x_0) = -\frac{1}{2V_3}\sum_{\vec{x}} \left\langle A_0^{ji}(x)\cdot \mathcal{O}^{ij}\right\rangle \tag{2.112}$$

$$= -\frac{1}{2}\sum_{\vec{x}} \left\langle \mathrm{Tr}\,\left(\bar{S}^j(x)^\dagger \gamma_0 \bar{S}^i(x)\right)\right\rangle_{\mathrm{G}} \tag{2.113}$$

$$g_{\mathrm{A}}^{ij}(x_0) = \frac{1}{2V_3}\sum_{\vec{x}} \left\langle A_0^{ji}(x)\cdot \mathcal{O}'^{ij}\right\rangle \tag{2.114}$$

$$= \frac{1}{2}\sum_{\vec{x}} \left\langle \mathrm{Tr}\,\left(\bar{R}^j(x)^\dagger \gamma_0 \bar{R}^i(x)\right)\right\rangle_{\mathrm{G}} \tag{2.115}$$

$$f_{\mathrm{P}}^{ij}(x_0) = -\frac{1}{2V_3}\sum_{\vec{x}} \left\langle P^{ji}(x)\cdot \mathcal{O}^{ij}\right\rangle \tag{2.116}$$

$$= \frac{1}{2}\sum_{\vec{x}} \left\langle \mathrm{Tr}\,\left(\bar{S}^j(x)^\dagger \bar{S}^i(x)\right)\right\rangle_{\mathrm{G}} \tag{2.117}$$

$$g_{\mathrm{P}}^{ij}(x_0) = -\frac{1}{2V_3}\sum_{\vec{x}} \left\langle P^{ji}(x)\cdot \mathcal{O}'^{ij}\right\rangle \tag{2.118}$$

$$= \frac{1}{2}\sum_{\vec{x}} \left\langle \mathrm{Tr}\,\left(\bar{R}^j(x)^\dagger \bar{R}^i(x)\right)\right\rangle_{\mathrm{G}} \tag{2.119}$$

$$f_1^{ij} = -\frac{1}{2V_3^2}\cdot \left\langle \mathcal{O}'^{ji}\cdot \mathcal{O}^{ij}\right\rangle \tag{2.120}$$

$$= \frac{1}{2}\left\langle \mathrm{Tr}\,\left(\bar{S}_{\mathrm{T}}^{j\dagger}\bar{S}_{\mathrm{T}}^i\right)\right\rangle_{\mathrm{G}} \tag{2.121}$$

$$k_{\mathrm{V}}^{ij}(x_0) = -\frac{1}{6V_3}\sum_{\vec{x}} \left\langle V_k^{ji}(x)\cdot \mathcal{V}_k^{ij}\right\rangle \tag{2.122}$$

$$= \frac{1}{6}\sum_{\vec{x}} \Big\langle \mathrm{Tr}\ \Big(\gamma_5 \bar{S}^j(x)^\dagger \gamma_5 \gamma_k \bar{S}^i(x)\gamma_k\Big)\Big\rangle_\mathrm{G} \tag{2.123}$$

$$\begin{aligned} = \frac{1}{3}\sum_{\vec{x}} \Big\langle \mathrm{Tr}_\mathrm{col}\Big[&\bar{S}^j(x)^\dagger_{11}\big(-2\bar{S}^i(x)_{42}-\bar{S}^i(x)_{31}\big)+\bar{S}^j(x)^\dagger_{12}\bar{S}^i(x)_{32}\\ &+\bar{S}^j(x)^\dagger_{22}\big(-2\bar{S}^i(x)_{31}-\bar{S}^i(x)_{42}\big)+\bar{S}^j(x)^\dagger_{21}\bar{S}^i(x)_{41}\\ &+\bar{S}^j(x)^\dagger_{31}\big(-2\bar{S}^i(x)_{22}-\bar{S}^i(x)_{11}\big)+\bar{S}^j(x)^\dagger_{32}\bar{S}^i(x)_{12}\\ &+\bar{S}^j(x)^\dagger_{42}\big(-2\bar{S}^i(x)_{11}-\bar{S}^i(x)_{22}\big)+\bar{S}^j(x)^\dagger_{41}\bar{S}^i(x)_{21}\Big]\Big\rangle_\mathrm{G} \end{aligned} \tag{2.124}$$

$$l_\mathrm{V}^{ij}(x_0) = -\frac{1}{6V_3}\sum_{\vec{x}}\Big\langle V_k^{ji}(x)\cdot \mathcal{V}_k'^{ij}\Big\rangle \tag{2.125}$$

$$= \frac{1}{6}\sum_{\vec{x}} \Big\langle \mathrm{Tr}\ \Big(\gamma_5 \bar{R}^j(x)^\dagger \gamma_5 \gamma_k \bar{R}^i(x)\gamma_k\Big)\Big\rangle_\mathrm{G} \tag{2.126}$$

$$\begin{aligned} = -\frac{1}{3}\sum_{\vec{x}} \Big\langle \mathrm{Tr}_\mathrm{col}\Big[&\bar{R}^j(x)^\dagger_{11}\big(-2\bar{R}^i(x)_{42}-\bar{R}^i(x)_{31}\big)+\bar{R}^j(x)^\dagger_{12}\bar{R}^i(x)_{32}\\ &+\bar{R}^j(x)^\dagger_{22}\big(-2\bar{R}^i(x)_{31}-\bar{R}^i(x)_{42}\big)+\bar{R}^j(x)^\dagger_{21}\bar{R}^i(x)_{41}\\ &+\bar{R}^j(x)^\dagger_{31}\big(-2\bar{R}^i(x)_{22}-\bar{R}^i(x)_{11}\big)+\bar{R}^j(x)^\dagger_{32}\bar{R}^i(x)_{12}\\ &+\bar{R}^j(x)^\dagger_{42}\big(-2\bar{R}^i(x)_{11}-\bar{R}^i(x)_{22}\big)+\bar{R}^j(x)^\dagger_{41}\bar{R}^i(x)_{21}\Big]\Big\rangle_\mathrm{G} \end{aligned} \tag{2.127}$$

$$k_\mathrm{T}^{ij}(x_0) = -\frac{1}{6V_3}\sum_{\vec{x}}\Big\langle T_{k0}^{ji}(x)\cdot \mathcal{V}_k^{ij}\Big\rangle \tag{2.128}$$

$$= \frac{1}{6}\sum_{\vec{x}} \Big\langle \mathrm{Tr}\ \Big(\gamma_5 \bar{S}^j(x)^\dagger \gamma_5 i\sigma_{k0} \bar{S}^i(x)\gamma_k\Big)\Big\rangle_\mathrm{G} \tag{2.129}$$

$$\begin{aligned} = \frac{1}{3}\sum_{\vec{x}} \Big\langle \mathrm{Tr}_\mathrm{col}\Big[&\bar{S}^j(x)^\dagger_{11}\big(-2\bar{S}^i(x)_{22}-\bar{S}^i(x)_{11}\big)+\bar{S}^j(x)^\dagger_{12}\bar{S}^i(x)_{12}\\ &+\bar{S}^j(x)^\dagger_{22}\big(-2\bar{S}^i(x)_{11}-\bar{S}^i(x)_{22}\big)+\bar{S}^j(x)^\dagger_{21}\bar{S}^i(x)_{21}\\ &+\bar{S}^j(x)^\dagger_{31}\big(-2\bar{S}^i(x)_{42}-\bar{S}^i(x)_{31}\big)+\bar{S}^j(x)^\dagger_{32}\bar{S}^i(x)_{32}\\ &+\bar{S}^j(x)^\dagger_{42}\big(-2\bar{S}^i(x)_{31}-\bar{S}^i(x)_{42}\big)+\bar{S}^j(x)^\dagger_{41}\bar{S}^i(x)_{41}\Big]\Big\rangle_\mathrm{G} \end{aligned} \tag{2.130}$$

$$l_\mathrm{T}^{ij}(x_0) = \frac{1}{6V_3}\sum_{\vec{x}}\Big\langle T_{k0}^{ji}(x)\cdot \mathcal{V}_k'^{ij}\Big\rangle \tag{2.131}$$

$$= -\frac{1}{6}\sum_{\vec{x}} \Big\langle \mathrm{Tr}\ \Big(\gamma_5 \bar{R}^j(x)^\dagger \gamma_5 i\sigma_{k0} \bar{R}^i(x)\gamma_k\Big)\Big\rangle_\mathrm{G} \tag{2.132}$$

$$\begin{aligned} = \frac{1}{3}\sum_{\vec{x}} \Big\langle \mathrm{Tr}_\mathrm{col}\Big[&\bar{R}^j(x)^\dagger_{11}\big(-2\bar{R}^i(x)_{22}-\bar{R}^i(x)_{11}\big)+\bar{R}^j(x)^\dagger_{12}\bar{R}^i(x)_{12}\\ &+\bar{R}^j(x)^\dagger_{22}\big(-2\bar{R}^i(x)_{11}-\bar{R}^i(x)_{22}\big)+\bar{R}^j(x)^\dagger_{21}\bar{R}^i(x)_{21}\\ &+\bar{R}^j(x)^\dagger_{31}\big(-2\bar{R}^i(x)_{42}-\bar{R}^i(x)_{31}\big)+\bar{R}^j(x)^\dagger_{32}\bar{R}^i(x)_{32} \end{aligned}$$

$$+ \bar{R}^j(x)^\dagger_{42}\big(- 2\bar{R}^i(x)_{31} - \bar{R}^i(x)_{42}\big) + \bar{R}^j(x)^\dagger_{41}\bar{R}^i(x)_{41}\Big]\Big\rangle_{\mathrm{G}} \tag{2.133}$$

$$k_1^{ij} = -\frac{1}{6V_3^2} \cdot \left\langle \mathcal{V}'^{ji}_k \cdot \mathcal{V}^{ij}_k \right\rangle \tag{2.134}$$

$$= \frac{1}{6}\left\langle \mathrm{Tr}\, \left(\gamma_5 \bar{S}^{j\dagger}_{\mathrm{T}} \gamma_5 \gamma_k \bar{S}^i_{\mathrm{T}} \gamma_k \right) \right\rangle_{\mathrm{G}} \tag{2.135}$$

$$= \frac{2}{3}\Big\langle \mathrm{Tr}_{\mathrm{col}}\, \Big[\bar{S}^j_{\mathrm{T}}(x)^\dagger_{11}\big(2\bar{S}^i_{\mathrm{T}}(x)_{22} + \bar{S}^i_{\mathrm{T}}(x)_{11}\big) - \bar{S}^j_{\mathrm{T}}(x)^\dagger_{12}\bar{S}^i_{\mathrm{T}}(x)_{12}$$

$$+ \bar{S}^j_{\mathrm{T}}(x)^\dagger_{22}\big(2\bar{S}^i_{\mathrm{T}}(x)_{11} + \bar{S}^i_{\mathrm{T}}(x)_{22}\big) - \bar{S}^j_{\mathrm{T}}(x)^\dagger_{21}\bar{S}^i_{\mathrm{T}}(x)_{21}\Big]\Big\rangle_{\mathrm{G}} \tag{2.136}$$

In the above equations, Tr is the trace in Dirac and color space, whereas $\mathrm{Tr}_{\mathrm{col}}$ is the trace in color space only. Repeated indices are summed over.

The normalization of the correlation functions that are defined above is chosen so that it agrees with the normalization conventions in the APE program (see also the introduction of chapter 4). The convention on the APE was that at tree level, zero mass (or $\kappa = 1/8$) and $\vec{\theta} = \vec{0}$ the correlations are to fulfill

$$f_{\mathrm{A}}(x_0) = g_{\mathrm{A}}(x_0) = k_{\mathrm{T}}(x_0) = l_{\mathrm{T}}(x_0) = -3 \tag{2.137}$$

$$f_{\mathrm{P}}(x_0) = g_{\mathrm{P}}(x_0) = k_{\mathrm{V}}(x_0) = l_{\mathrm{V}}(x_0) = +3 \tag{2.138}$$

$$f_1 = k_1 = +3 \tag{2.139}$$

for all $x_0 \in \{1, \ldots, T-1\}$. (The value 3 stems from the number of colors.)

3. Heavy-Quark Effective Theory (HQET)

The procedures described in the previous chapter for calculating correlation functions non-perturbatively on a lattice can well be applied to light quarks (up, down, strange), but the treatment of heavy quarks, especially the beauty quark, poses a problem, because they would require a very fine lattice. Typical length scales that must be covered in a simulation are given by the masses of the appropriate hadrons, e.g. the lightest hadron containing up and down quarks is the pion: $m_\pi \approx 140\,\mathrm{MeV}$, whereas the B meson, which contains a beauty quark and one of the light quarks, has a mass of $m_\mathrm{B} \approx 5\,\mathrm{GeV}$ [Nak10]. Therefore, the infrared cutoff, i.e. the physical lattice length L, must be larger than $1/m_\pi$, so that a whole pion “fits” into the lattice, and the ultraviolet cutoff, i.e. the lattice constant a, must be smaller than $1/m_\mathrm{B}$:

$$a \ll \frac{1}{m_\mathrm{B}}, \dots, \frac{1}{m_\pi} \ll L. \tag{3.1}$$

Adequate lattices would have to include 240 to 480 points in each direction [Som10, p. 1]. This is beyond the capabilities of today's computers. Instead of QCD, an effective theory for the beauty quark must be used. One way is heavy-quark effective theory (HQET). It can be used to describe systems containing one heavy quark and one light quark (it is also possible to treat systems with more light quarks, like the Λ_b baryon, but we will consider only systems with one light quark here) [Som10, p. 2]. Its strategy is to expand the Lagrange density and the observables (heavy–light correlation functions) in powers of the inverse mass m_h^{-1} of the heavy quark.

3.1. HQET Lagrange Density in the Continuum

This derivation of the HQET Lagrange density is based on [Som10].

The full-QCD Lagrange density for a heavy-quark field $\psi(x)$ in continuous Euclidean space is

$$\mathcal{L}(x) = \bar{\psi}(x)(\gamma_\mu D_\mu + m_\mathrm{h})\psi(x). \tag{3.2}$$

Other quark fields and the gauge field's kinetic term are omitted here.

In HQET, we now suppose the heavy quark to be nearly at rest and to have no components with large spatial momenta compared to its mass. Then we expect that the Lagrange density can be written in terms of a quark and an anti-quark field, ψ_h and $\psi_{\bar{\mathrm{h}}}$, which are only weakly coupled. This corresponds to the non-relativistic limit of the Dirac equation, where a spinor wave function is separated into its two "large" and its two "small" components. Here, we will rewrite the Lagrange density via two Foldy–Wouthuysen–Tani transformations and expand the resulting terms in powers of the inverse mass m_{h}^{-1} of the heavy quark.

First, let us consider the order of the field's derivatives compared to m_{h}. We assume that the heavy quark's spatial momentum is much smaller than its mass, and so, the same must hold for the spatial derivatives, whereas the time derivative of the heavy-quark field is of the same order as m_{h}:

$$D_j\psi \in \mathcal{O}(1) \qquad\qquad D_0\psi \in \mathcal{O}(m_{\mathrm{h}}). \tag{3.3}$$

The field strength tensor $F_{\mu\nu}$ is assumed to be of order one.

The large and small, or quark and anti-quark components of ψ are defined as

$$\psi_{\mathrm{h}} = P_+\psi \qquad\qquad \bar\psi_{\mathrm{h}} = \bar\psi P_+ \tag{3.4}$$

$$\psi_{\bar{\mathrm{h}}} = P_-\psi \qquad\qquad \bar\psi_{\bar{\mathrm{h}}} = \bar\psi P_- \tag{3.5}$$

with the projectors P_+ and P_-:

$$P_+ = \frac{1}{2}(1+\gamma_0) \qquad\qquad P_- = \frac{1}{2}(1-\gamma_0). \tag{3.6}$$

In the QCD Lagrangian the large and small components are coupled via the spatial gamma matrices in $\gamma_j D_j$. To decouple them, we perform a Foldy–Wouthuysen–Tani transformation (see [Kö91]) and replace ψ by ψ':

$$\psi = \mathrm{e}^{-\frac{1}{2m_{\mathrm{h}}}\gamma_j D_j}\psi' \qquad\qquad \bar\psi = \bar\psi'\mathrm{e}^{+\frac{1}{2m_{\mathrm{h}}}\gamma_j \overleftarrow{D}_j}. \tag{3.7}$$

Then, the Lagrange density can be written as

$$\mathcal{L}(x) = \bar\psi'(x)\mathrm{e}^{+\frac{1}{2m_{\mathrm{h}}}\gamma_j \overleftarrow{D}_j}(\gamma_\mu D_\mu + m_{\mathrm{h}})\mathrm{e}^{-\frac{1}{2m_{\mathrm{h}}}\gamma_j D_j}\psi'. \tag{3.8}$$

Via partial integration, the operators $\overleftarrow{D}_j$ can be transformed into $-D_j$:

$$\mathcal{L}(x) = \bar\psi'(x)\mathrm{e}^{-\frac{1}{2m_{\mathrm{h}}}\gamma_j D_j}(\gamma_\mu D_\mu + m_{\mathrm{h}})\mathrm{e}^{-\frac{1}{2m_{\mathrm{h}}}\gamma_j D_j}\psi'. \tag{3.9}$$

To evaluate this expression, we use the relation

$$\mathrm{e}^B A \mathrm{e}^B = A + \{A,B\} + \frac{1}{2}\{\{A,B\},B\} + \frac{1}{6}\{\{\{A,B\},B\},B\} + \dots \tag{3.10}$$

$$= \sum_{n=0}^{\infty}\frac{1}{n!}\{\{\dots\{A,\underbrace{B\}\dots\},B\}}_{n\text{ times}} \tag{3.11}$$

with

$$A = \gamma_\mu D_\mu + m_{\mathrm{h}} \qquad\qquad B = -\frac{1}{2m_{\mathrm{h}}}\gamma_j D_j. \tag{3.12}$$

The first terms in this series are

$$\begin{aligned} A &= \gamma_0 D_0 + m_\mathrm{h} + \gamma_j D_j && (3.13) \\ \{A, B\} &= -\gamma_j D_j - \frac{1}{m_\mathrm{h}} D_j D_j - \frac{1}{m_\mathrm{h}} \sigma_j B_j(x) - \frac{1}{2m_\mathrm{h}} \gamma_0 \gamma_j F_{0j}(x) && (3.14) \\ \frac{1}{2}\{\{A, B\}, B\} &= \frac{1}{2m_\mathrm{h}} D_j D_j + \frac{1}{2m_\mathrm{h}} \sigma_j B_j(x) + \mathcal{O}(m_\mathrm{h}^{-2}), && (3.15) \end{aligned}$$

with the field strength

$$F_{\mu\nu}(x) = [D_\mu, D_\nu] = -\mathrm{i}g(\partial_\mu A_\nu(x) - \partial_\nu A_\mu(x)) - g^2[A_\mu(x), A_\nu(x)], \tag{3.16}$$

the magnetic field

$$B_j(x) = \frac{\mathrm{i}}{2} \varepsilon_{jkl} F_{kl} \tag{3.17}$$

and the Pauli matrices σ_1, σ_2, σ_3. In the gamma representation used here (see appendix A.1), they are to be understood as a shorthand for the following 4×4-matrices acting in Dirac space:

$$\sigma_j \otimes I_2 = \begin{pmatrix} \sigma_j & 0 \\ 0 & \sigma_j \end{pmatrix}. \tag{3.18}$$

This results in the following Lagrangian:

$$\begin{aligned} \mathcal{L}(x) = \bar{\psi}'(x) \left(\gamma_0 D_0 + m_\mathrm{h} - \frac{1}{2m_\mathrm{h}} D_j D_j - \frac{1}{2m_\mathrm{h}} \sigma_j B_j - \frac{1}{2m_\mathrm{h}} \gamma_0 \gamma_j F_{0j} \right) \psi'(x) \\ + \mathcal{O}(m_\mathrm{h}^{-2}). \end{aligned} \tag{3.19}$$

The term $\gamma_j D_j$ has vanished, but the quark and anti-quark components $P_\pm \psi'$ are still coupled at order m_h^{-1} via $-1/(2m_\mathrm{h})\gamma_0\gamma_j F_{0j}$. We can make this term disappear by a second FWT transformation:

$$\psi' = \mathrm{e}^{\frac{1}{4m_\mathrm{h}^2}\gamma_0\gamma_j F_{0j}} \psi'' \qquad \bar{\psi}' = \bar{\psi}'' \mathrm{e}^{\frac{1}{4m_\mathrm{h}^2}\gamma_0\gamma_j F_{0j}}. \tag{3.20}$$

It creates only one additional term that is of order m_h^{-1}, which comes from

$$\left\{ \frac{1}{4m_\mathrm{h}^2} \gamma_0 \gamma_j F_{0j}, m_\mathrm{h} \right\} = \frac{1}{2m_\mathrm{h}} \gamma_0 \gamma_j F_{0j}, \tag{3.21}$$

and exactly cancels $-1/(2m_\mathrm{h})\gamma_0\gamma_j F_{0j}$.

So, after the second transformation, the Lagrange density is

$$\mathcal{L}(x) = \bar{\psi}''(x) \left(\gamma_0 D_0 + m_\mathrm{h} - \frac{1}{2m_\mathrm{h}} D_j D_j - \frac{1}{2m_\mathrm{h}} \sigma_j B_j \right) \psi''(x) + \mathcal{O}(m_\mathrm{h}^{-2}). \tag{3.22}$$

Now, we can split ψ'' into its quark and anti-quark components by inserting $\psi'' = \psi''_{\mathrm{h}} + \psi''_{\bar{\mathrm{h}}} = P_+\psi''_{\mathrm{h}} + P_-\psi''_{\bar{\mathrm{h}}}$:

$$\begin{aligned}\mathcal{L}(x) = (\bar{\psi}''_{\mathrm{h}}(x)P_+ + \bar{\psi}''_{\bar{\mathrm{h}}}(x)P_-)\left(\gamma_0 D_0 + m_{\mathrm{h}} - \frac{1}{2m_{\mathrm{h}}}D_jD_j - \frac{1}{2m_{\mathrm{h}}}\sigma_j B_j\right)\\ \times (P_+\psi''_{\mathrm{h}}(x) + P_-\psi''_{\bar{\mathrm{h}}}(x)) + \mathcal{O}(m_{\mathrm{h}}^{-2}).\end{aligned} \tag{3.23}$$

We expand this expression and use the fact that the projectors commute with the operators $\gamma_0 D_0$, m_{h} etc. between them. Furthermore, we can use $\gamma_0 P_+ = P_+$ and $\gamma_0 P_- = -P_-$ to cancel the gamma matrix in $\gamma_0 D_0$. The resulting terms are

$$\begin{aligned}\mathcal{L}(x) = \bar{\psi}''_{\mathrm{h}}(x)P_+P_+\left(D_0 + m_{\mathrm{h}} - \frac{1}{2m_{\mathrm{h}}}D_jD_j - \frac{1}{2m_{\mathrm{h}}}\sigma_j B_j\right)\psi''_{\mathrm{h}}(x)\\ + \bar{\psi}''_{\bar{\mathrm{h}}}(x)P_-P_-\left(-D_0 + m_{\mathrm{h}} - \frac{1}{2m_{\mathrm{h}}}D_jD_j - \frac{1}{2m_{\mathrm{h}}}\sigma_j B_j\right)\psi''_{\bar{\mathrm{h}}}(x)\\ + \bar{\psi}''_{\mathrm{h}}(x)P_+P_-(\dots)\psi''_{\bar{\mathrm{h}}}(x) + \bar{\psi}''_{\bar{\mathrm{h}}}(x)P_-P_+(\dots)\psi''_{\mathrm{h}}(x) + \mathcal{O}(m_{\mathrm{h}}^{-2}).\end{aligned} \tag{3.24}$$

Since $P_+P_- = 0$, the last two terms are zero. For simplicity, we will relabel $\psi''_{\mathrm{h}} \to \psi_{\mathrm{h}}$ from now on and do the same for the anti-quark field and the adjoints. If we drop the $\mathcal{O}(m_{\mathrm{h}}^{-2})$ terms, we are left with the Lagrangian

$$\mathcal{L}(x) = \mathcal{L}_{\mathrm{h}}^{\mathrm{stat}}(x) + \mathcal{L}_{\bar{\mathrm{h}}}^{\mathrm{stat}}(x) + \frac{1}{2m_{\mathrm{h}}}\mathcal{L}_{\mathrm{h}}^{(1)}(x) + \frac{1}{2m_{\mathrm{h}}}\mathcal{L}_{\bar{\mathrm{h}}}^{(1)}(x), \tag{3.25}$$

where the static and sub-leading parts are defined as

$$\mathcal{L}_{\mathrm{h}}^{\mathrm{stat}}(x) = \bar{\psi}_{\mathrm{h}}(x)(D_0 + m_{\mathrm{h}})\psi_{\mathrm{h}}(x) \qquad \mathcal{L}_{\bar{\mathrm{h}}}^{\mathrm{stat}}(x) = \bar{\psi}_{\bar{\mathrm{h}}}(x)(-D_0 + m_{\mathrm{h}})\psi_{\bar{\mathrm{h}}}(x) \tag{3.26}$$

$$\mathcal{L}_{\mathrm{h}}^{(1)}(x) = -\left(\mathcal{O}_{\mathrm{kin}}(x) + \mathcal{O}_{\mathrm{spin}}(x)\right) \qquad \mathcal{L}_{\bar{\mathrm{h}}}^{(1)}(x) = -\left(\bar{\mathcal{O}}_{\mathrm{kin}}(x) + \bar{\mathcal{O}}_{\mathrm{spin}}(x)\right) \tag{3.27}$$

with the kinetic and spin insertions

$$\mathcal{O}_{\mathrm{kin}}(x) = \bar{\psi}_{\mathrm{h}}(x)\cdot\vec{D}^2\psi_{\mathrm{h}}(x) \qquad \bar{\mathcal{O}}_{\mathrm{kin}}(x) = \bar{\psi}_{\bar{\mathrm{h}}}(x)\cdot\vec{D}^2\psi_{\bar{\mathrm{h}}}(x) \tag{3.28}$$

$$\mathcal{O}_{\mathrm{spin}}(x) = \bar{\psi}_{\mathrm{h}}(x)\cdot(\vec{\sigma}\cdot\vec{B})\psi_{\mathrm{h}}(x) \qquad \bar{\mathcal{O}}_{\mathrm{spin}}(x) = \bar{\psi}_{\bar{\mathrm{h}}}(x)\cdot(\vec{\sigma}\cdot\vec{B})\psi_{\bar{\mathrm{h}}}(x). \tag{3.29}$$

Usually we will only need the quark terms in the following chapters and the anti-quark terms will be dropped.

We can do a last transformation to cancel the mass term m_{h}: We replace

$$\psi_{\mathrm{h}}(x) \longrightarrow \mathrm{e}^{-m_{\mathrm{h}}x_0}\psi_{\mathrm{h}}(x) \qquad \psi_{\bar{\mathrm{h}}}(x) \longrightarrow \mathrm{e}^{+m_{\mathrm{h}}x_0}\psi_{\bar{\mathrm{h}}}(x) \tag{3.30}$$

$$\bar{\psi}_{\mathrm{h}}(x) \longrightarrow \mathrm{e}^{+m_{\mathrm{h}}x_0}\bar{\psi}_{\mathrm{h}}(x) \qquad \bar{\psi}_{\bar{\mathrm{h}}}(x) \longrightarrow \mathrm{e}^{-m_{\mathrm{h}}x_0}\bar{\psi}_{\bar{\mathrm{h}}}(x). \tag{3.31}$$

Then, the derivative D_0 of the exponential creates an additional term which compensates the mass term in $\mathcal{L}_{\mathrm{h}/\bar{\mathrm{h}}}^{\mathrm{stat}}$:

$$\mathcal{L}_{\mathrm{h}}^{\mathrm{stat}}(x) = \bar{\psi}_{\mathrm{h}}(x)D_0\psi_{\mathrm{h}}(x) \qquad \mathcal{L}_{\bar{\mathrm{h}}}^{\mathrm{stat}}(x) = -\bar{\psi}_{\bar{\mathrm{h}}}(x)D_0\psi_{\bar{\mathrm{h}}}(x). \tag{3.32}$$

3.2. Evaluation of Expectation Values

To compute the expectation value of observables, we would now insert the HQET Lagrangian (3.25) into the exponential of the path integral. For an observable $\mathcal{O}$ this would be

$$\langle \mathcal{O} \rangle = \int_{\text{fields}} \mathcal{O} \cdot W_{\text{NRQCD}}, \tag{3.33}$$

where the integral goes over the gauge and all quark fields and the weight W_{NRQCD} is

$$W_{\text{NRQCD}} = \exp\left[- \int \mathrm{d}^4 x \left(\mathcal{L}_{\text{light}}(x) + \mathcal{L}_{\text{h}}^{\text{stat}}(x) + \mathcal{L}_{\text{h}}^{(1)}(x) \right) \right]. \tag{3.34}$$

Here, we have supposed that the heavy anti-quark fields can be dropped for the computation of $\langle \mathcal{O} \rangle$.

However, with the weight W_{NRQCD}, the theory is not renormalizable. We will not treat renormalization in detail here, because in the following chapters we will only be concerned with correlation functions at finite lattice spacing, but it is necessary to understand the way how observables are evaluated in HQET.

An established criterion for a theory to be renormalizable is that the Lagrange density may contain only local field products with a mass dimension that is less than or equal to the space-time dimension d, in this case $d = 4$ (see [Som10, p. 9], [Mag05, p. 140]). Then, UV divergences can be cured by adding counterterms containing local field products of mass dimension $\leq d$. If we only consider the static Lagrange density $\mathcal{L}_{\text{h}}^{\text{stat}}$ (or $\mathcal{L}_{\bar{\text{h}}}^{\text{stat}}$), we can see that it is renormalizable: The counterterms have to share the same symmetries as $\mathcal{L}_{\text{h}}^{\text{stat}}$, and so, only a wave-function and mass renormalization is needed.

However, the sub-leading $\mathcal{O}_{\text{kin}}(x) = \bar{\psi}_{\text{h}}(x)\vec{D}^2\psi_{\text{h}}(x)$ and $\mathcal{O}_{\text{spin}}(x) = \bar{\psi}_{\text{h}}(x)\vec{B}\psi_{\text{h}}(x)$ have mass dimension 5 and so are not renormalizable. That means, in order to make observables finite, we would have to add infinitely many counterterms to the Lagrangian. The first ones of these counterterms have mass dimension 5 and are just proportional to $\mathcal{O}_{\text{kin}}$ and $\mathcal{O}_{\text{spin}}$. This can be show by using the theory's symmetries and the equation of motion again. So, only the weights of $\mathcal{O}_{\text{kin}}$ and $\mathcal{O}_{\text{spin}}$ are modified, and the sub-leading terms become:

$$\mathcal{L}_{\text{h}}^{(1)}(x) = -\left(\omega_{\text{kin}} \cdot \mathcal{O}_{\text{kin}}(x) + \omega_{\text{spin}} \cdot \mathcal{O}_{\text{spin}}(x) \right) \tag{3.35}$$

with $\omega_{\text{kin}}, \omega_{\text{spin}} \in \mathcal{O}(m_{\text{h}}^{-1})$. But in higher orders infinitely many more counterterms are to be added, which introduce infinitely many free parameters. Therefore, the theory would loose its predictive power. In HQET, this problem is circumvented by applying the following prescription: Instead of the weight W_{NRQCD}, the weight W_{HQET} is used in the path integral. In W_{HQET} the exponential is expanded up to the first order in the sub-leading terms:

$$W_{\text{HQET}} = \left[1 - \int \mathrm{d}^4x \mathcal{L}_{\text{h}}^{(1)}(x)\right] \cdot \exp\left[-\int \mathrm{d}^4x \left(\mathcal{L}_{\text{light}}(x) + \mathcal{L}_{\text{h}}^{\text{stat}}(x)\right)\right] \tag{3.36}$$

$$= W_{\text{NRQCD}} + \mathcal{O}(m_{\text{h}}^{-2}). \tag{3.37}$$

This can be done, since we are only interested in corrections up to the order of m_{h}^{-1} and the difference of W_{HQET} and W_{NRQCD} is $\mathcal{O}(m_{\text{h}}^{-2})$. Only the renormalizable terms of the light and the static action are left in the exponential, and the sub-leading terms become part of the observables as additional insertions in their expectation values.

So, the HQET expectation value of an observable $\mathcal{O}$ is calculated as

$$\langle\mathcal{O}\rangle = \langle\mathcal{O}\rangle_{\text{stat}} + \omega_{\text{kin}}\langle\mathcal{O}\rangle_{\text{kin}} + \omega_{\text{spin}}\langle\mathcal{O}\rangle_{\text{spin}} \tag{3.38}$$

with the static expectation value:

$$\langle\mathcal{O}\rangle_{\text{stat}} = N \cdot \int_{\text{fields}} \mathcal{O}\mathrm{e}^{-\int \mathrm{d}^4x\mathcal{L}_{\text{light}}(x)+\mathcal{L}_{\text{h}}^{\text{stat}}(x)(x)} \tag{3.39}$$

and the expectation values with an insertion of the kinetic or spin part of $\mathcal{L}_{\text{h}}^{(1)}$:

$$\langle\mathcal{O}\rangle_{\text{kin}} = \int \mathrm{d}^4x\langle\mathcal{O}\mathcal{O}_{\text{kin}}(x)\rangle_{\text{stat}} \qquad \mathcal{O}_{\text{kin}}(x) = \bar{\psi}_{\text{h}}(x)\vec{D}^2\psi_{\text{h}}(x) \tag{3.40}$$

$$\langle\mathcal{O}\rangle_{\text{spin}} = \int \mathrm{d}^4x\langle\mathcal{O}\mathcal{O}_{\text{spin}}(x)\rangle_{\text{stat}} \qquad \mathcal{O}_{\text{spin}}(x) = \bar{\psi}_{\text{h}}(x)(\vec{\sigma}\cdot\vec{B})\psi_{\text{h}}(x). \tag{3.41}$$

The normalization N in front of the path integral is such that $\langle 1\rangle_{\text{stat}} = 1$.

3.3. Discretization

The static Lagrangian $\mathcal{L}_{\text{h}}^{\text{stat}}$ and the kinetic and spin insertions $\mathcal{O}_{\text{kin}}$ and $\mathcal{O}_{\text{spin}}$ are discretized according to [Mor05, p.37] and [Som10, pp. 13, 37].

In the Lagrange density of the light quarks $\mathcal{L}_{\text{light}}$, we have replaced the term $\gamma_0 D_0$ by a symmetric lattice derivative times γ_0 and we have added the Wilson term to avoid doubler modes (see section 2.3.4):

$$\gamma_\mu D_\mu \longrightarrow \frac{1}{2}\gamma_\mu\left(\nabla_\mu + \nabla_\mu^*\right) - \frac{1}{2}\nabla_\mu^*\nabla_\mu. \tag{3.42}$$

We will do the same in the static Lagrangian $\mathcal{L}_{\text{h}}^{\text{stat}}$, i.e. we replace

$$D_0 \longrightarrow \frac{1}{2}\left(\nabla_0 + \nabla_0^*\right) - \frac{1}{2}\gamma_0\nabla_0^*\nabla_0 \tag{3.43}$$

and obtain

$$\mathcal{L}_{\mathrm{h}}^{\mathrm{stat}}(x) = \bar{\psi}_{\mathrm{h}}(x) \cdot \left[\frac{1}{2}\left(\nabla_0 + \nabla_0^*\right) - \frac{1}{2}\gamma_0 \nabla_0^* \nabla_0\right] \cdot \psi_{\mathrm{h}}(x) \tag{3.44}$$

$$\begin{aligned} &= \bar{\psi}_{\mathrm{h}}(x) \cdot \left[\frac{1}{2}\Big(U_0(x)\psi_{\mathrm{h}}(x+\hat{0}) - U_0(x-\hat{0})^\dagger \psi_{\mathrm{h}}(x-\hat{0})\Big)\right. \\ &\qquad \left. - \frac{1}{2}\gamma_0 \Big(U_0(x)\psi_{\mathrm{h}}(x+\hat{0}) - 2\psi_{\mathrm{h}}(x) + U_0(x-\hat{0})^\dagger \psi_{\mathrm{h}}(x-\hat{0})\Big)\right]. \end{aligned} \tag{3.45}$$

Since $\gamma_0 \psi_{\mathrm{h}} = \psi_{\mathrm{h}}$, which follows from its definition, the above formula can be simplified to

$$\begin{aligned} \mathcal{L}_{\mathrm{h}}^{\mathrm{stat}}(x) = \bar{\psi}_{\mathrm{h}}(x) \cdot &\left[\frac{1}{2}\Big(U_0(x)\psi_{\mathrm{h}}(x+\hat{0}) - U_0(x-\hat{0})^\dagger \psi_{\mathrm{h}}(x-\hat{0})\Big)\right. \\ &\left. - \frac{1}{2}\Big(U_0(x)\psi_{\mathrm{h}}(x+\hat{0}) - 2\psi_{\mathrm{h}}(x) + U_0(x-\hat{0})^\dagger \psi_{\mathrm{h}}(x-\hat{0})\Big)\right] \end{aligned} \tag{3.46}$$

$$= \bar{\psi}_{\mathrm{h}}(x) \cdot \Big[\psi_{\mathrm{h}}(x) - U_0(x-\hat{0})^\dagger \psi_{\mathrm{h}}(x-\hat{0})\Big] \tag{3.47}$$

$$= \bar{\psi}_{\mathrm{h}}(x) \cdot \nabla_0^* \psi_{\mathrm{h}}(x). \tag{3.48}$$

In this expression, one often inserts a "smeared" gauge field $V_\mu(x)$ instead of the original gauge field $U_\mu(x)$. There are different ways to define this smeared field (e.g. HYP smearing [Has01, DM05]). In any case, $V_\mu(x)$ is a gauge-invariant combination of the original $U_\mu(x)$ and links from its neighborhood. Since sums of SU(3)-matrices are generally not in SU(3) again, this combination is projected back to an SU(3)-matrix, so that $V_\mu(x) \in \mathrm{SU}(3)$ is guaranteed.

The expressions for the anti-quark Lagrangian $\mathcal{L}_{\bar{\mathrm{h}}}^{\mathrm{stat}}$ are very similar. The difference is that $\gamma_0 \psi_{\bar{\mathrm{h}}} = -\psi_{\bar{\mathrm{h}}}$. Hence, the discretized Lagrangian contains the forward instead of the backward derivative:

$$\mathcal{L}_{\bar{\mathrm{h}}}(x) = \bar{\psi}_{\bar{\mathrm{h}}}(x) \cdot \nabla_0 \psi_{\bar{\mathrm{h}}}(x). \tag{3.49}$$

Some of the correlation functions we will look at also involve spatial derivatives. Since they are not part of the action (i.e. the argument of the exponential function in (3.36)), we do not have to worry about doublers and do not need to add Wilson terms. The continuum derivatives are simply replaced by symmetric lattice derivatives:

$$D_k \longrightarrow \frac{1}{2}\left(\nabla_k + \nabla_k^*\right). \tag{3.50}$$

The kinetic term $\mathcal{O}_{\mathrm{kin}}$, too, includes spatial derivatives in the operator $\vec{D}^2$. Here, they are of second order. These second-order derivatives are not replaced by the square of the above expression, but by $\nabla_k^* \nabla_k = \nabla_k \nabla_k^*$. Both expressions reproduce the second derivative up to errors of $\mathcal{O}(a^2)$. Thus, the discretized version of $\mathcal{O}_{\mathrm{kin}}$

becomes

$$\mathcal{O}_{\text{kin}}(x) = \bar{\psi}_{\text{h}}(x) \cdot \nabla_k^* \nabla_k \psi_{\text{h}}(x) \tag{3.51}$$

$$= \sum_k \bar{\psi}_{\text{h}}(x) \cdot \left[\lambda_k U_k(x) \psi_{\text{h}}(x+\hat{k}) - 2\psi_{\text{h}}(x) + \lambda_k^* U_k(x-\hat{k})^\dagger \psi_{\text{h}}(x-\hat{k})\right]. \tag{3.52}$$

(The phase factors λ_k are the same as in (2.59) and (2.60).)

The spin insertion $\mathcal{O}_{\text{spin}}$ involves the chromomagnetic field $\vec{B}$, which has been defined as

$$B_j = \frac{\mathrm{i}}{2} \varepsilon_{jkl} F_{kl} \tag{3.53}$$

with the continuum field strength tensor $F_{\mu\nu}$. This is replaced by the discretized version that has already appeared in the Sheikoleslami–Wohlert term (see (2.68)) and consists of the sum over a clover-like set of plaquettes.

3.4. HQET on a Lattice with Schrödinger-Functional Boundary Conditions

We will use the discretized HQET on a lattice where the fields obey Schrödinger-functional boundary conditions. Then, the static HQET action for quarks is

$$S_{\text{h}} = \sum_{x_0=1}^{T} \sum_{\vec{x}} \bar{\psi}_{\text{h}}(x) \cdot \nabla_0^* \psi_{\text{h}}(x). \tag{3.54}$$

In space directions the heavy-quark fields are to obey periodic boundary conditions

$$\psi_{\text{h}}(x + L\hat{k}) = \psi_{\text{h}}(x) \qquad \bar{\psi}_{\text{h}}(x + L\hat{k}) = \bar{\psi}_{\text{h}}(x) \tag{3.55}$$

(for $k = 1, 2, 3$). And at the time boundaries they are to fulfill

$$\psi_{\text{h}}(0, \vec{x}) = \rho_{\text{h}}(\vec{x}) \qquad \bar{\psi}_{\text{h}}(T, \vec{x}) = \bar{\rho}'_{\text{h}}(\vec{x}). \tag{3.56}$$

As for the relativistic quarks, we will set the boundary fields ρ_{h} and $\bar{\rho}'_{\text{h}}$ to zero, so we can not use them to build correlation functions. Instead, we can use derivatives of the Schrödinger functional $\mathcal{Z}[\rho_{\text{h}}, \bar{\rho}'_{\text{h}}, \dots]$ with respect to them. These derivatives are formally identified with the fields

$$\bar{\zeta}_{\text{h}}(\vec{x}) = -\frac{\delta}{\delta \rho_{\text{h}}(\vec{x})} \qquad \zeta'_{\text{h}}(\vec{x}) = \frac{\delta}{\delta \bar{\rho}'_{\text{h}}(\vec{x})}. \tag{3.57}$$

If we use the static action as shown in (3.54), these fields effectively insert the products

$$\bar{\zeta}_{\text{h}}(\vec{x}) = \bar{\psi}_{\text{h}}(1, \vec{x}) \cdot V_0(0, \vec{x})^\dagger \qquad \zeta'_{\text{h}}(\vec{x}) = V_0(T-1, \vec{x})^\dagger \cdot \psi_{\text{h}}(T-1, \vec{x}). \tag{3.58}$$

3.5. Heavy-Quark Propagators

To evaluate correlation functions that involve a heavy quark by means of the Wick theorem, we have to calculate the propagator of the heavy quark first. The basic propagator is the "static propagator". It only takes into account the static part $\mathcal{L}_\mathrm{h}^\mathrm{stat}$ of the heavy-quark Lagrange density and can be used to compute the leading, $\mathcal{O}(m_\mathrm{h}^0)$ term of observables. For the sub-leading, $\mathcal{O}(m_\mathrm{h}^{-1})$ terms, two more propagators are defined, the kinetic and spin propagators W^kin and W^spin, which can be expressed by means of the static propagator and an $\mathcal{O}_\mathrm{kin}$ or $\mathcal{O}_\mathrm{spin}$ insertion.

3.5.1. Static Propagator

The static propagator $S_\mathrm{h}(x,y)$ describes the propagation of a quark under the static action $\mathcal{L}_\mathrm{h}^\mathrm{stat}$. It can be obtained as the solution of the equation of motion for a point-like quark source at y. Such a source is $P_+\delta_{xy}$. (Likewise, an anti-quark source would be $P_-\delta_{xy}$.) Then, the equation of motion for $S_\mathrm{h}(x,y)$ is

$$\nabla_0^* S_\mathrm{h}(x,y) = P_+\delta_{xy} \tag{3.59}$$

$$\iff \quad S_\mathrm{h}(x,y) = P_+\delta_{xy} + V_0(x-\hat{0})^\dagger S_\mathrm{h}(x-\hat{0},y). \tag{3.60}$$

Furthermore, it must not contain anti-quark components, and it must respect the boundary conditions:

$$P_- S_\mathrm{h}(x,y) = 0 \qquad\qquad P_+ S_\mathrm{h}((0,\vec{x}),y) = 0. \tag{3.61}$$

The solution can be obtained iteratively via (3.60). The result is:

$$S_\mathrm{h}(x,y) = \theta(x_0-y_0)\cdot P_+ \cdot W(x,y) \tag{3.62}$$

with

$$W(x,y) = \delta_{\vec{x}\vec{y}} \cdot V_0(x_0-1,\vec{x})^\dagger \dots V_0(y_0,\vec{x})^\dagger. \tag{3.63}$$

This means that in the static limit, the heavy quark is at rest and keeps its position in space (expressed by $\delta_{\vec{x}\vec{y}}$), and it propagates only forward in time. Often the Wilson line $W(x,y)$, which acts only in color space, will also be called the static propagator instead of S_h.

In the following, it will be convenient to express $W(x,y)$ in terms of the Wilson line $W(x)$ which connects $(0,\vec{x})$ at the lower boundary with x:

$$W(x) = V_0(x_0-1,\vec{x})^\dagger \dots V_0(0,\vec{x})^\dagger \tag{3.64}$$

$$\iff \quad W(x,y) = W(x)W(y)^\dagger \cdot \delta_{\vec{x}\vec{y}} \tag{3.65}$$

$$S_\mathrm{h}(x,y) = \theta(x_0-y_0)\delta_{\vec{x}\vec{y}} \cdot P_+ \cdot W(x)W(y)^\dagger. \tag{3.66}$$

($\theta(x_0-y_0)$ is the discrete Heaviside step function. It is one for $x_0 \geq y_0$ and zero for $x_0 < y_0$.) $W(x)$ will also be the quantity that we use in the implementation of our program (see section 6.1.1).

The static propagator is the result of the contraction of two heavy-quark fields $\psi_{\rm h}(x)$ and $\bar{\psi}_{\rm h}(y)$ in the static approximation:

$$S_{\rm h}(x,y) = \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(y)\rangle_{\rm F}. \tag{3.67}$$

We will need, in particular, the contraction of $\psi_{\rm h}(x)$ with a flat-wave-function source at the bottom (see footnote 4 on page 29 for the explanation of a "flat-wave-function source") and define

$$\bar{S}_{\rm h}(x) = \sum_{\vec{z}} \langle \psi_{\rm h}(x)\bar{\zeta}_{\rm h}(\vec{z})\rangle_{\rm F}. \tag{3.68}$$

By (3.58) we obtain

$$\bar{S}_{\rm h}(x) = \sum_{\vec{z}} \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(1,\vec{z})\rangle_{\rm F} \cdot V_0(0,\vec{z})^\dagger \tag{3.69}$$

$$= \sum_{\vec{z}} \theta(x_0-1)\delta_{\vec{x}\vec{z}} \cdot P_+ \cdot W(x)W(1,\vec{z})^\dagger \cdot V_0(0,\vec{z})^\dagger. \tag{3.70}$$

Since x is in the bulk, x_0 is greater than or equal to one and $\theta(x_0-1)=1$. Furthermore, the sum can be performed by using the δ-function, and $W(1,\vec{z})^\dagger = V_0(0,\vec{z})$ cancels the term $V_0(0,\vec{z})^\dagger$, so we are left with:

$$\bar{S}_{\rm h}(x) = P_+ \cdot W(x). \tag{3.71}$$

3.5.2. Kinetic Propagator

To compute correlation functions in sub-leading order, we need to evaluate expectation values of field products with an insertion of the kinetic term $\sum_z \mathcal{O}_{\rm kin}(z) = \sum_z \bar{\psi}(z)\cdot \nabla_k^*\nabla_k\psi(z)$. For this purpose, we define the kinetic propagator:

$$S_{\rm h}^{\rm kin}(x,y) = \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(y)\rangle_{\rm kin,F} = \sum_z \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(z)\cdot \nabla_k^*\nabla_k\psi_{\rm h}(z)\bar{\psi}_{\rm h}(y)\rangle_{\rm F}. \tag{3.72}$$

We contract $\psi_{\rm h}(x)$ with $\bar{\psi}_{\rm h}(z)$ and $\psi_{\rm h}(z)$ with $\bar{\psi}_{\rm h}(y)$:

$$S_{\rm h}^{\rm kin}(x,y) = \sum_z S_{\rm h}(x,z)\cdot \nabla_k^*\nabla_k S_{\rm h}(z,y) \tag{3.73}$$

$$= \theta(x_0-y_0)\delta_{\vec{x}\vec{y}}\cdot P_+ \cdot \sum_{z_0=y_0}^{x_0} W(x)W(z_0,\vec{x})^\dagger \cdot \nabla_k^*\nabla_k W(z_0,\vec{x})W(y_0,\vec{x})^\dagger. \tag{3.74}$$

For a source on the bottom lid, we can define the kinetic boundary-to-bulk propagator $\bar{S}_{\rm h}^{\rm kin}(x)$, which is evaluated in analogy to $S_{\rm h}^{\rm kin}$ and $\bar{S}_{\rm h}$:

$$\bar{S}_{\rm h}^{\rm kin}(x) = \sum_{\vec{y}} \langle \psi_{\rm h}(x)\bar{\zeta}_{\rm h}(\vec{y})\rangle_{\rm kin,F}, \tag{3.75}$$

using (3.58):

$$\bar{S}_{\rm h}^{\rm kin}(x) = \sum_{\vec{y}z} \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(z) \cdot \nabla_k^* \nabla_k \psi_{\rm h}(z)\bar{\psi}_{\rm h}(1,\vec{y})\rangle_{\rm F} \cdot V_0(0,\vec{y})^\dagger \tag{3.76}$$

$$= \sum_{\vec{y}z} S_{\rm h}(x,z) \cdot \nabla_k^* \nabla_k S_{\rm h}(z,(1,\vec{y})) \cdot V_0(0,\vec{y})^\dagger \tag{3.77}$$

$$= P_+ \cdot \sum_{z_0=1}^{x_0} W(x) W(z_0,\vec{x})^\dagger \cdot \nabla_k^* \nabla_k W(z_0,\vec{x}) \tag{3.78}$$

$$\bar{S}_{\rm h}^{\rm kin}(x) = P_+ \cdot W^{\rm kin}(x) \tag{3.79}$$

with

$$W^{\rm kin}(x) = \sum_{z_0=1}^{x_0} W(x) W(z_0,\vec{x})^\dagger \cdot \nabla_k^* \nabla_k W(z_0,\vec{x}). \tag{3.80}$$

As a computational advantage, note that $W^{\rm kin}$ can be computed recursively (like W itself) via the formula

$$W^{\rm kin}(x) = \nabla_k^* \nabla_k W(x_0,\vec{x}) + V_0(x-\hat{0})^\dagger \cdot W^{\rm kin}(x-\hat{0}) \tag{3.81}$$

with the starting point

$$W^{\rm kin}(0,\vec{x}) = 0. \tag{3.82}$$

3.5.3. Spin Propagator

In a similar way, we define the spin propagator, which contains an insertion of the operator $\sum_z \mathcal{O}_{\rm spin}(z) = \sum_z \bar{\psi}_{\rm h}(z) \cdot (\vec{\sigma} \cdot \vec{B})\psi_{\rm h}(z)$, as

$$S_{\rm h}^{\rm spin}(x,y) = \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(y)\rangle_{\rm spin,F} = \sum_z \langle \psi_{\rm h}(x)\bar{\psi}_{\rm h}(z) \cdot (\vec{\sigma}\cdot\vec{B})\psi_{\rm h}(z)\bar{\psi}_{\rm h}(y)\rangle_{\rm F} \tag{3.83}$$

$$= \theta(x_0-y_0)\delta_{\vec{x}\vec{y}} \cdot P_+ \cdot \sum_{z_0=y_0}^{x_0} W(x)W(z_0,\vec{x})^\dagger \cdot (\vec{\sigma}\cdot\vec{B}) W(z_0,\vec{x}) W(y_0,\vec{x})^\dagger \tag{3.84}$$

and the boundary-to-bulk propagator for a source at the bottom boundary:

$$\bar{S}_{\rm h}^{\rm spin}(x) = \sum_{\vec{y}} S_{\rm h}^{\rm spin}(x,(0,\vec{y})) = P_+ \cdot \vec{\sigma} \cdot \vec{W}^{\rm spin}(x) \tag{3.85}$$

with

$$W_j^{\rm spin}(x) = \sum_{z_0=1}^{x_0} W(x)W(z_0,\vec{x})^\dagger \cdot B_j W(z_0,\vec{x}). \tag{3.86}$$

Here, we have not included $\vec{\sigma}$ into the definition of $\vec{W}^{\mathrm{spin}}$, because in this way, each vector component $W_j^{\mathrm{spin}}(x)$ is only a matrix in color space, not in Dirac space. Thus, it can be handled in the same way as the static and kinetic propagator (see also section 6.2.1).

$\vec{W}^{\mathrm{spin}}$ can be computed in a recursive fashion, too:

$$W^{\mathrm{spin}}(x) = (\vec{\sigma}\cdot\vec{B})W(x) + V_0(x-\hat{0})^\dagger \cdot W^{\mathrm{spin}}(x-\hat{0}) \tag{3.87}$$

$$W^{\mathrm{spin}}(0,\vec{x}) = 0. \tag{3.88}$$

3.5.4. Boundary-to-Boundary Propagators

We will also need the expectation value of a field at the upper and a field at the lower boundary. This can be expressed in terms of propagators as

$$\sum_{\vec{x}\vec{y}} \left\langle \zeta_{\mathrm{h}}'(\vec{x})\bar{\zeta}_{\mathrm{h}}(\vec{y})\right\rangle_{\mathrm{F}} = \sum_{\vec{x}\vec{y}} U_0(T-1,\vec{x})^\dagger \cdot \left\langle \psi_{\mathrm{h}}(T-1,\vec{x})\bar{\psi}_{\mathrm{h}}(1,\vec{y})\right\rangle_{\mathrm{F}} \cdot U_0(0,\vec{y})^\dagger \tag{3.89}$$

$$= \sum_{\vec{x}} P_+ \cdot W(T,\vec{x}). \tag{3.90}$$

Likewise, the kinetic and spin expectation values can be calculated. For these expectation values, we define the boundary-to-boundary propagators:

$$\bar{S}_{\mathrm{hT}} = P_+ \cdot W_{\mathrm{T}} \qquad \bar{S}_{\mathrm{hT}}^{\mathrm{kin}} = P_+ \cdot W_{\mathrm{T}}^{\mathrm{kin}} \qquad \bar{S}_{\mathrm{hT}}^{\mathrm{spin}} = P_+ \cdot \vec{\sigma} \cdot \vec{W}_{\mathrm{T}}^{\mathrm{spin}} \tag{3.91}$$

with

$$W_{\mathrm{T}} = \sum_{\vec{x}} W(T,\vec{x}) \quad W_{\mathrm{T}}^{\mathrm{kin}} = \sum_{\vec{x}} W^{\mathrm{kin}}(T,\vec{x}) \quad \vec{W}_{\mathrm{T}}^{\mathrm{spin}} = \sum_{\vec{x}} \vec{W}^{\mathrm{spin}}(T,\vec{x}). \tag{3.92}$$

3.5.5. Summary of Contractions for a Source at the Bottom

The following equations summarize how field contractions with a flat-wave-function source at the bottom can be expressed in terms of the propagators that have been defined in the previous sections:

$$\sum_{\vec{y}} \left\langle \psi_{\mathrm{h}}(x)\bar{\zeta}_{\mathrm{h}}(\vec{y})\right\rangle_{\mathrm{F}} = \bar{S}_{\mathrm{h}}(x) = P_+ \cdot W(x) \tag{3.93}$$

$$\sum_{\vec{y}z} \left\langle \psi_{\mathrm{h}}(x)\cdot \mathcal{O}_{\mathrm{kin}}(z)\cdot \bar{\zeta}_{\mathrm{h}}(\vec{y})\right\rangle_{\mathrm{F}} = \bar{S}_{\mathrm{h}}^{\mathrm{kin}}(x) = P_+ \cdot W^{\mathrm{kin}}(x) \tag{3.94}$$

$$\sum_{\vec{y}z} \left\langle \psi_{\mathrm{h}}(x)\cdot \mathcal{O}_{\mathrm{spin}}(z)\cdot \bar{\zeta}_{\mathrm{h}}(\vec{y})\right\rangle_{\mathrm{F}} = \bar{S}_{\mathrm{h}}^{\mathrm{spin}}(x) = P_+ \cdot \vec{\sigma}\cdot \vec{W}^{\mathrm{spin}}(x) \tag{3.95}$$

$$\sum_{\vec{x}\vec{y}} \left\langle \zeta_{\mathrm{h}}'(\vec{x})\bar{\zeta}_{\mathrm{h}}(\vec{y})\right\rangle_{\mathrm{F}} = \bar{S}_{\mathrm{hT}} = P_+ \cdot W_{\mathrm{T}} \tag{3.96}$$

$$\sum_{\vec{x}\vec{y}z} \left\langle \zeta_{\mathrm{h}}'(\vec{x})\cdot \mathcal{O}_{\mathrm{kin}}(z)\cdot \bar{\zeta}_{\mathrm{h}}(\vec{y})\right\rangle_{\mathrm{F}} = \bar{S}_{\mathrm{hT}}^{\mathrm{kin}} = P_+ \cdot W_{\mathrm{T}}^{\mathrm{kin}} \tag{3.97}$$

$$\sum_{\vec{x}\vec{y}z} \left\langle \zeta_{\rm h}'(\vec{x}) \cdot \mathcal{O}_{\rm spin}(z) \cdot \bar{\zeta}_{\rm h}(\vec{y}) \right\rangle_{\rm F} = \bar{S}_{\rm hT}^{\rm spin} = P_+ \cdot \vec{\sigma} \cdot \vec{W}_{\rm T}^{\rm spin}. \tag{3.98}$$

3.6. Correlation Functions in HQET

The observables in the heavy-quark effective theory that we will compute are two-point correlation functions that involve one heavy quark. For the sub-leading (i.e. $\mathcal{O}(m_{\rm h}^{-1})$) terms, we will also need the same correlation functions with an insertion of $\mathcal{O}_{\rm kin}$ or $\mathcal{O}_{\rm spin}$.

A typical example of a correlation function is $f_{\rm A}^{\rm stat}(x_0)$:

$$f_{\rm A}^{\rm stat}(x_0) = -\frac{1}{2} \sum_{\vec{x}\vec{y}\vec{z}} \langle \bar{\psi}(x)\gamma_0\gamma_5\psi_{\rm h}(x)\bar{\zeta}_{\rm h}(\vec{y})\gamma_5\zeta(\vec{z})\rangle. \tag{3.99}$$

Here, the expectation value is computed with respect to the static action of the heavy quark only. To distinguish the correlation function from $f_{\rm A}$ with fully relativistic quarks, it carries the label "stat". Fields without an index "h" are to denote light quarks. Their flavor index is not written explicitly in the following sections, but e.g. ζ and $\bar{\psi}$ in the above formula are intended to have the same flavor.

The corresponding sub-leading correlation functions are by labeled $f_{\rm A}^{\rm kin}(x_0)$ and $f_{\rm A}^{\rm spin}(x_0)$. The first one has an additional $\mathcal{O}_{\rm kin}$ insertion, the second one an $\mathcal{O}_{\rm spin}$ insertion:

$$f_{\rm A}^{\rm kin}(x_0) = -\frac{1}{2} \sum_{\vec{x}\vec{y}\vec{z}u} \langle \bar{\psi}(x)\gamma_0\gamma_5\psi_{\rm h}(x)\bar{\psi}_{\rm h}(u) \cdot \nabla_k^*\nabla_k\psi_{\rm h}(u)\bar{\zeta}_{\rm h}(\vec{y})\gamma_5\zeta(\vec{z})\rangle \tag{3.100}$$

$$f_{\rm A}^{\rm spin}(x_0) = -\frac{1}{2} \sum_{\vec{x}\vec{y}\vec{z}u} \langle \bar{\psi}(x)\gamma_0\gamma_5\psi_{\rm h}(x)\bar{\psi}_{\rm h}(u) \cdot (\vec{\sigma}\cdot\vec{B})\psi_{\rm h}(u)\bar{\zeta}_{\rm h}(\vec{y})\gamma_5\zeta(\vec{z})\rangle. \tag{3.101}$$

Strictly speaking, we should call $f_{\rm A}^{\rm kin}$ and $f_{\rm A}^{\rm spin}$ three-point correlation functions, since they involve fields at the point x, u and the bottom boundary.

As a new feature, the program shall also be used to compute vector correlations in HQET, e.g. the static version $k_{\rm V}^{\rm stat}$ of $k_{\rm V}$:

$$k_{\rm V}^{\rm stat}(x_0) = -\frac{1}{6} \sum_{\vec{x}\vec{y}\vec{z}k} \left\langle \bar{\psi}(x) \cdot \gamma_k \cdot \psi_{\rm h}(x)\bar{\zeta}_{\rm h}(\vec{y}) \cdot \gamma_k \cdot \zeta(\vec{z}) \right\rangle \tag{3.102}$$

Here, an additional summation over the index k of the gamma matrices is to be done.

Another type of correlation function that we have to compute contains spatial derivatives of the quark fields. Examples are $f_{\delta{\rm A}}^{\rm stat}(x_0)$ and $f_{A_k^{(4)}}^{\rm stat}$ (the field products that are involved are defined in [DM] e.g.):

$$f_{\delta\mathrm{A}}^{\mathrm{stat}}(x_0) = -\frac{1}{2}\sum_{\vec{x}\vec{y}\vec{z}}\langle\bar{\psi}(x)\overleftarrow{\nabla}_k \cdot \gamma_k\gamma_5\psi_\mathrm{h}(x)\bar{\zeta}_\mathrm{h}(\vec{y})\gamma_5\zeta(\vec{z})\rangle \tag{3.103}$$

$$f_{A_k^{(4)}}^{\mathrm{stat}}(x_0) = \frac{\mathrm{i}}{6}\sum_{\vec{x}\vec{y}\vec{z}k}\langle\bar{\psi}(x)\frac{1}{2}(\widetilde{\nabla}_k - \overleftarrow{\nabla}_k)\gamma_5\psi_\mathrm{h}(x)\bar{\zeta}_\mathrm{h}(\vec{y})\gamma_5\zeta(\vec{z})\rangle. \tag{3.104}$$

These are needed for the improvement of $f_\mathrm{A}^{\mathrm{stat}}$, i.e. to improve its convergence to the continuum limit.

There are also corresponding boundary-to-boundary correlation functions: f_1^{stat}, k_1^{stat} and the correlations with a kinetic or spin insertion. For example, f_1^{stat} is defined as

$$f_1^{\mathrm{stat}} = -\frac{1}{2\sqrt{V_3}}\sum_{\vec{u}\vec{v}\vec{x}\vec{y}}\left\langle\bar{\zeta}'(\vec{u})\cdot\gamma_5\cdot\zeta_\mathrm{h}'(\vec{v})\bar{\zeta}_\mathrm{h}(\vec{x})\cdot\gamma_5\cdot\zeta(\vec{y})\right\rangle. \tag{3.105}$$

The normalization of the correlation functions defined above match the conventions in the APE program (see the introduction of chapter 4). There, the correlations are normalized in such a way that at tree level, zero mass of the light quark (or $\kappa = 1/8$) and $\vec{\theta} = \vec{0}$ the static correlation functions are

$$f_\mathrm{A}^{\mathrm{stat}}(x_0) = -3\cdot V_3 \qquad \text{for all } x_0 \in \{1,\ldots,T-1\} \tag{3.106}$$

$$f_1^{\mathrm{stat}} = +3\cdot V_3^{3/2}. \tag{3.107}$$

In the next sections, we will discuss how to compute correlation functions like $f_\mathrm{A}^{\mathrm{stat}}$ from the light and heavy propagators. For most cases, the other types of correlation functions can be reduced to the form of $f_\mathrm{A}^{\mathrm{stat}}$ with additional summations or modified propagators and also the boundary-to-boundary correlations can be computed in a similar fashion.

3.6.1. Simple Static Correlation Functions

Instead of specific CF like $f_\mathrm{A}^{\mathrm{stat}}$, we consider a general correlation function of the form

$$c^{\mathrm{stat}}(x_0) = -\frac{1}{\sqrt{V_3}}\sum_{\vec{x}\vec{y}\vec{z}}\langle\bar{\psi}(x)\cdot\mathcal{A}\cdot\psi_\mathrm{h}(x)\bar{\zeta}_\mathrm{h}(\vec{y})\cdot\mathcal{B}\cdot\zeta(\vec{z})\rangle_\mathrm{F} \tag{3.108}$$

with arbitrary 4×4 matrices $\mathcal{A}$ and $\mathcal{B}$ acting in Dirac space. In contrast to the previous definition of $f_\mathrm{A}^{\mathrm{stat}}$ in (3.99), we leave out the integration over the gauge field, because this is not part of our program's task.

Equation (3.108) does not cover such correlations as $k_\mathrm{V}^{\mathrm{stat}}$ (equation 3.102), which involve a summation over the matrices $\mathcal{A}$ and $\mathcal{B}$. Instead, every summand has to be evaluated separately and the results must be summed afterwards (see section 6.2.1).

We can now use the Wick theorem to express $c^{\mathrm{stat}}(x_0)$ in terms of the propagators that have been introduced in the previous section. We contract the two heavy-quark fields with each other and the two light-quark fields (the rules for replacing these contractions by propagators are summarized in sections 2.3.7 and 3.5.5):

$$c^{\mathrm{stat}}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\gamma_5 \bar{S}(x)^\dagger \gamma_5 \cdot \mathcal{A} \cdot \bar{S}_{\mathrm{h}}(x) \cdot \mathcal{B}\right] \tag{3.109}$$

$$= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}(x)^\dagger \cdot \gamma_5 \mathcal{A} \cdot P_+ W(x) \cdot \mathcal{B}\gamma_5\right]. \tag{3.110}$$

The minus sign has disappeared, because to apply the contraction rules, the fields have to be rearranged, e.g. ζ may be taken to the left, passing three other Graßmann fields, it picks up a minus sign that compensates for the previous one.

3.6.2. Simple Kinetic and Spin Correlation Functions

The sub-leading correlation functions that correspond to the general c^{stat} are

$$c^{\mathrm{kin}}(x_0) = -\frac{1}{\sqrt{V_3}} \sum_{u\vec{x}\vec{y}\vec{z}} \langle \bar{\psi}(x) \cdot \mathcal{A} \cdot \psi_{\mathrm{h}}(x) \cdot \mathcal{O}_{\mathrm{kin}}(u) \cdot \bar{\zeta}_{\mathrm{h}}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z})\rangle_{\mathrm{F}} \tag{3.111}$$

$$= -\frac{1}{\sqrt{V_3}} \sum_{\vec{x}\vec{y}\vec{z}u} \langle \bar{\psi}(x) \cdot \mathcal{A} \cdot \psi_{\mathrm{h}}(x)\bar{\psi}_{\mathrm{h}}(u) \cdot \nabla_k^*\nabla_k \psi_{\mathrm{h}}(u)\bar{\zeta}_{\mathrm{h}}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z})\rangle_{\mathrm{F}} \tag{3.112}$$

$$c^{\mathrm{spin}}(x_0) = -\frac{1}{\sqrt{V_3}} \sum_{u\vec{x}\vec{y}\vec{z}} \langle \bar{\psi}(x) \cdot \mathcal{A} \cdot \psi_{\mathrm{h}}(x) \cdot \mathcal{O}_{\mathrm{spin}}(u) \cdot \bar{\zeta}_{\mathrm{h}}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z})\rangle_{\mathrm{F}} \tag{3.113}$$

$$= -\frac{1}{\sqrt{V_3}} \sum_{\vec{x}\vec{y}\vec{z}u} \langle \bar{\psi}(x) \cdot \mathcal{A} \cdot \psi_{\mathrm{h}}(x)\bar{\psi}_{\mathrm{h}}(u) \cdot (\vec{\sigma}\cdot\vec{B})\psi_{\mathrm{h}}(u)\bar{\zeta}_{\mathrm{h}}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z})\rangle_{\mathrm{F}}. \tag{3.114}$$

As before, Wick's theorem can be applied. We contract $\zeta_{\mathrm{h}}(\vec{z})$ and $\bar{\psi}_{\mathrm{h}}(x)$, $\psi_{\mathrm{h}}(x)$ and $\bar{\psi}_{\mathrm{h}}(u)$, and $\psi_{\mathrm{h}}(u)$ and $\bar{\zeta}_{\mathrm{h}}(\vec{y})$. The last two contractions result just in the kinetic or spin propagator, $\bar{S}_{\mathrm{h}}^{\mathrm{kin}}$ or $\bar{S}_{\mathrm{h}}^{\mathrm{spin}}$. Thus, the kinetic and spin correlation functions become

$$c^{\mathrm{kin}}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\gamma_5 \bar{S}(x)^\dagger \gamma_5 \cdot \mathcal{A} \cdot \bar{S}_{\mathrm{h}}^{\mathrm{kin}}(x) \cdot \mathcal{B}\right] \tag{3.115}$$

$$= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}(x)^\dagger \cdot \gamma_5 \mathcal{A} P_+ W^{\mathrm{kin}}(x) \mathcal{B}\gamma_5\right] \tag{3.116}$$

$$c^{\mathrm{spin}}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\gamma_5 \bar{S}(x)^\dagger \gamma_5 \cdot \mathcal{A} \cdot \bar{S}_{\mathrm{h}}^{\mathrm{spin}} \cdot \mathcal{B}\right] \tag{3.117}$$

$$= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}(x)^\dagger \cdot \gamma_5 \mathcal{A} P_+ \left(\vec{\sigma}\cdot\vec{W}^{\mathrm{spin}}(x)\right) \mathcal{B}\gamma_5\right]. \tag{3.118}$$

3.6.3. Static Correlation Functions Including a Derivative

Here, we will look at correlation functions similar to $f^{\rm stat}_{\delta \rm A}(x_0)$ and $f^{\rm stat}_{A_k^4}(x_0)$ (see (3.103) and (3.104)), more precisely, we will evaluate a CF of the general form

$$c^{\rm stat}_\delta(x_0)_k = -\frac{1}{\sqrt{V_3}} \sum_{\vec{x}\vec{y}\vec{z}} \langle \bar{\psi}(x)(d \cdot \overleftarrow{\widetilde{\nabla}}_k + d_{\rm h} \cdot \widetilde{\nabla}_k) \cdot \mathcal{A} \cdot \psi_{\rm h}(x) \bar{\zeta}_{\rm h}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z}) \rangle_{\rm F}. \quad (3.119)$$

In contrast to e.g. $f^{\rm stat}_{\delta \rm A}$, a summation over the index k of the derivative is not included in this definition. Such a summation must be done by calculating $c^{\rm stat}_\delta(x_0)_k$ three times for $k = 1, 2, 3$ and summing the results.

Despite the additional derivatives, the application of Wick's theorem works in the same way as for $c^{\rm stat}$ and the result is

$$c^{\rm stat}_\delta(x_0)_k = \sum_{\vec{x}} {\rm Tr} \left[\bar{S}(x)^\dagger (d \cdot \overleftarrow{\widetilde{\nabla}}_k + d_{\rm h} \cdot \widetilde{\nabla}_k) \cdot \gamma_5 \mathcal{A} \cdot P_+ W(x) \cdot \mathcal{B} \gamma_5 \right]. \quad (3.120)$$

However, the sum of backward and forward derivative can be simplified by a discrete version of integration by parts. First, note that $W(x)$ can be put left of $\gamma_5 \mathcal{A} P_+$, since these are matrices in Dirac space, whereas $W(x)$ acts only in color space and space-time. The backward derivative acts on $\bar{S}(x)^\dagger$ and the forward derivative on $W(x)$. All the other factors are constant. So, we can concentrate on the part of the whole expression that includes only $\bar{S}(x)^\dagger$, the derivative operators and $W(x)$:

$$\begin{aligned} &\sum_{\vec{x}} \bar{S}(x)^\dagger (d \cdot \overleftarrow{\widetilde{\nabla}}_k + d_{\rm h} \cdot \widetilde{\nabla}_k) W(x) \\ &\quad = \sum_{\vec{x}} \frac{d}{2} \left[\bar{S}(x+\hat{k})^\dagger U_k(x)^\dagger \cdot \lambda_k^* - \bar{S}(x-\hat{k})^\dagger U_k(x-\hat{k}) \cdot \lambda_k \right] W(x) \\ &\qquad + \sum_{\vec{x}} \frac{d_{\rm h}}{2} \bar{S}(x)^\dagger \left[(\lambda_{\rm h})_k \cdot U_k(x) W(x+\hat{k}) - (\lambda_{\rm h}^*)_k \cdot U_k(x-\hat{k})^\dagger W(x-\hat{k}) \right]. \end{aligned} \quad (3.121)$$

In the last lines, we have written out the derivatives. The phase factors λ for the light and the heavy quark may be different. To distinguish them, the heavy quark's phase factor $\lambda_{\rm h}$ carries the index "h".

Both sums include similar terms. In each sum, the relativistic propagator $\bar{S}$ is once multiplied by the heavy propagator W in positive k-direction and once by W in negative k-direction. Because we sum over all $\vec{x}$ and apply periodic boundary conditions in all space directions, we can combine both terms via summation shifts, either in a backward derivative, acting on the light quark, or in a forward derivative, acting on the heavy quark. We will take the last option, because the heavy propagator $W(x)$ has fewer components than the relativistic propagator $\bar{S}(x)$, since it has no Dirac structure, and thus, a computation of a derivative of $W(x)$ is presumably faster.

We expand the first sum of (3.121),

$$\begin{aligned}
&\sum_{\vec{x}} \bar{S}(x)^\dagger (d \cdot \overleftarrow{\nabla}_k + d_\mathrm{h} \cdot \widetilde{\nabla}_k) W(x) \\
&\quad = \sum_{\vec{x}} \frac{d\lambda_k^*}{2} \bar{S}(x+\hat{k})^\dagger U_k(x)^\dagger W(x) - \sum_{\vec{x}} \frac{d\lambda_k}{2} \bar{S}(x-\hat{k})^\dagger U_k(x-\hat{k}) W(x) \\
&\qquad + \sum_{\vec{x}} \frac{d_\mathrm{h}}{2} \bar{S}(x)^\dagger \left[(\lambda_\mathrm{h})_k \cdot U_k(x) W(x+\hat{k}) - (\lambda_\mathrm{h}^*)_k \cdot U_k(x-\hat{k})^\dagger W(x-\hat{k}) \right]
\end{aligned} \tag{3.122}$$

substitute $x_\mathrm{old} = x_\mathrm{new} - \hat{k}$ in the sum proportional to $d\lambda_k^*$, and $x_\mathrm{old} = x_\mathrm{new} + \hat{k}$ in the sum proportional to $d\lambda_k$

$$\begin{aligned}
&= \sum_{\vec{x}} \frac{d\lambda_k^*}{2} \bar{S}(x)^\dagger U_k(x-\hat{k})^\dagger W(x-\hat{k}) - \sum_{\vec{x}} \frac{d\lambda_k}{2} \bar{S}(x)^\dagger U_k(x) W(x+\hat{k}) \\
&\quad + \sum_{\vec{x}} \frac{d_\mathrm{h}}{2} \bar{S}(x)^\dagger \left[(\lambda_\mathrm{h})_k \cdot U_k(x) W(x+\hat{k}) - (\lambda_\mathrm{h}^*)_k \cdot U_k(x-\hat{k})^\dagger W(x-\hat{k}) \right]
\end{aligned} \tag{3.123}$$

and combine the obtained terms with the last sum:

$$= \sum_{\vec{x}} \frac{1}{2} \bar{S}(x)^\dagger \left[\mu_k \cdot U_k(x) W(x+\hat{k}) - \mu_k^* \cdot U_k(x-\hat{k})^\dagger W(x-\hat{k}) \right], \tag{3.124}$$

where

$$\mu_k = d_\mathrm{h} (\lambda_\mathrm{h})_k - d\lambda_k. \tag{3.125}$$

The term that we get has still the form of a derivative. Only the phase factor μ_k is modified and is now a combination of the light and heavy phase factors, λ_k and $(\lambda_\mathrm{h})_k$.

Finally, we can insert the expression that we have found into (3.120) again:

$$\begin{aligned}
c_\delta^\mathrm{stat}(x_0)_k = \sum_{\vec{x}} \mathrm{Tr} \Bigg[& \bar{S}(x)^\dagger \cdot \frac{\mu_k U_k(x) W(x+\hat{k}) - \mu_k^* U_k(x-\hat{k})^\dagger W(x-\hat{k})}{2} \\
& \times \gamma_5 \mathcal{A} P_+ \mathcal{B} \gamma_5 \Bigg].
\end{aligned} \tag{3.126}$$

This is the way correlation functions with derivatives will be computed in our program (see the appropriate parts of section 6.2.1).

3.6.4. Kinetic and Spin Correlation Functions Including a Derivative

These correlation functions look like $c_\delta^{\mathrm{stat}}(x_0)$, but they include an additional $\mathcal{O}_{\mathrm{kin}}$ or $\mathcal{O}_{\mathrm{spin}}$ insertion:

$$\begin{aligned} c_\delta^{\mathrm{kin}}(x_0)_k = -\frac{1}{\sqrt{V_3}} \sum_{\vec{x}\vec{y}\vec{z}} \langle \bar{\psi}(x)(d \cdot \overleftarrow{\widetilde{\nabla}}_k + d_{\mathrm{h}} \cdot \widetilde{\nabla}_k) \cdot \mathcal{A} \cdot \psi_{\mathrm{h}}(x) \\ \times \bar{\psi}_{\mathrm{h}}(u) \cdot \nabla_k^* \nabla_k \psi_{\mathrm{h}}(u) \bar{\zeta}_{\mathrm{h}}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z}) \rangle_{\mathrm{F}} \end{aligned} \tag{3.127}$$

$$\begin{aligned} c_\delta^{\mathrm{spin}}(x_0)_k = -\frac{1}{\sqrt{V_3}} \sum_{\vec{x}\vec{y}\vec{z}} \langle \bar{\psi}(x)(d \cdot \overleftarrow{\widetilde{\nabla}}_k + d_{\mathrm{h}} \cdot \widetilde{\nabla}_k) \cdot \mathcal{A} \cdot \psi_{\mathrm{h}}(x) \\ \times \bar{\psi}_{\mathrm{h}}(u) \cdot (\vec{\sigma} \cdot \vec{B}) \psi_{\mathrm{h}}(u) \bar{\zeta}_{\mathrm{h}}(\vec{y}) \cdot \mathcal{B} \cdot \zeta(\vec{z}) \rangle_{\mathrm{F}}. \end{aligned} \tag{3.128}$$

The application of the Wick theorem will not be shown in detail here. As for $c^{\mathrm{kin}}(x_0)$ and $c^{\mathrm{spin}}(x_0)$, the static propagator $W(x)$ in (3.126) is simply replaced by the kinetic or spin propagator:

$$\begin{aligned} c_\delta^{\mathrm{kin}}(x_0)_k = \sum_{\vec{x}} \mathrm{Tr} \Bigg[\bar{S}(x)^\dagger \cdot \frac{\mu_k U_k(x) W^{\mathrm{kin}}(x+\hat{k}) - \mu_k^* U_k(x-\hat{k})^\dagger W^{\mathrm{kin}}(x-\hat{k})}{2} \\ \times \gamma_5 \mathcal{A} P_+ \mathcal{B} \gamma_5 \Bigg] \end{aligned} \tag{3.129}$$

$$\begin{aligned} c_\delta^{\mathrm{spin}}(x_0)_k = \sum_{\vec{x}} \mathrm{Tr} \Bigg[\bar{S}(x)^\dagger \cdot \sigma_j \cdot \frac{\mu_k U_k(x) W_j^{\mathrm{spin}}(x+\hat{k}) - \mu_k^* U_k(x-\hat{k})^\dagger W_j^{\mathrm{spin}}(x-\hat{k})}{2} \\ \times \gamma_5 \mathcal{A} P_+ \mathcal{B} \gamma_5 \Bigg]. \end{aligned} \tag{3.130}$$

Correlations of this kind will probably not be needed, because they provide corrections of the order $\mathcal{O}(m_{\mathrm{h}}^{-2})$, but it is comparatively cheap to compute them, because all the necessary ingredients are already present (i.e. the light propagator $\bar{S}(x)$, the kinetic propagator $W^{\mathrm{kin}}(x)$ and the spin propagator $W^{\mathrm{spin}}(x)$), and they have been calculated on the APE, too. Therefore, they are described here just for completeness.

3.6.5. Static Boundary-to-Boundary Correlation Functions

The type of boundary-to-boundary correlations that we will consider here looks like

$$c_1^{\mathrm{stat}} = \frac{1}{V_3} \sum_{\vec{u}\vec{v}\vec{x}\vec{y}} \left\langle \bar{\zeta}'(\vec{u}) \cdot \mathcal{A} \cdot \zeta'_{\mathrm{h}}(\vec{v}) \bar{\zeta}_{\mathrm{h}}(\vec{x}) \cdot \mathcal{B} \cdot \zeta(\vec{y}) \right\rangle_{\mathrm{F}}, \tag{3.131}$$

where $\mathcal{A}$ and $\mathcal{B}$ are arbitrary 4×4-matrices that act in Dirac space. The only possible contraction is

$$= -\frac{1}{V_3} \mathrm{Tr} \left\{ \sum_{\vec{u}\vec{y}} \left[\zeta(\vec{y}) \bar{\zeta}'(\vec{u}) \right]_{\mathrm{F}} \cdot \mathcal{A} \cdot \sum_{\vec{v}\vec{x}} \left[\zeta'_{\mathrm{h}}(\vec{v}) \bar{\zeta}_{\mathrm{h}}(\vec{x}) \right]_{\mathrm{F}} \cdot \mathcal{B} \right\}. \tag{3.132}$$

With the relations given in sections 2.3.7 and 3.5.5, this can be written in terms of the boundary-to-boundary propagators (note that $\bar{S}_{\mathrm{T}}$ entails another minus sign):

$$
\begin{aligned}
c_1^{\mathrm{stat}} &= \mathrm{Tr}\left\{\gamma_5 \bar{S}_{\mathrm{T}}^{\dagger} \gamma_5 \cdot \mathcal{A} \cdot \bar{S}_{\mathrm{hT}} \cdot \mathcal{B}\right\} && (3.133)\\
&= \mathrm{Tr}\left\{\gamma_5 \bar{S}_{\mathrm{T}}^{\dagger} \gamma_5 \cdot \mathcal{A} \cdot P_+ W_{\mathrm{T}} \cdot \mathcal{B}\right\}. && (3.134)
\end{aligned}
$$

3.6.6. Kinetic and Spin Boundary-to-Boundary Correlation Functions

Boundary-to-boundary correlations with a kinetic or a spin insertion look like

$$
\begin{aligned}
c_1^{\mathrm{kin}} &= \frac{1}{V_3} \sum_{\vec{u}\vec{v}w\vec{x}\vec{y}} \left\langle \bar{\zeta}'(\vec{u}) \cdot \mathcal{A} \cdot \zeta_{\mathrm{h}}'(\vec{v}) \bar{\psi}_{\mathrm{h}}(w) \cdot \nabla_k^* \nabla_k \psi_{\mathrm{h}}(w) \bar{\zeta}_{\mathrm{h}}(\vec{x}) \cdot \mathcal{B} \cdot \zeta(\vec{y}) \right\rangle_{\mathrm{F}} && (3.135)\\
c_1^{\mathrm{spin}} &= \frac{1}{V_3} \sum_{\vec{u}\vec{v}w\vec{x}\vec{y}} \left\langle \bar{\zeta}'(\vec{u}) \cdot \mathcal{A} \cdot \zeta_{\mathrm{h}}'(\vec{v}) \bar{\psi}_{\mathrm{h}}(w) \cdot (\vec{\sigma} \cdot \vec{B}) \psi_{\mathrm{h}}(w) \bar{\zeta}_{\mathrm{h}}(\vec{x}) \cdot \mathcal{B} \cdot \zeta(\vec{y}) \right\rangle_{\mathrm{F}}. && (3.136)
\end{aligned}
$$

As for the boundary-to-bulk correlations, the heavy-quark fields $\zeta_{\mathrm{h}}'(\vec{v})$ and $\bar{\psi}_{\mathrm{h}}(w)$, $\psi_{\mathrm{h}}(w)$ and $\bar{\zeta}_{\mathrm{h}}(\vec{x})$ and the light-quark fields $\zeta(\vec{y})$ and $\bar{\zeta}'(\vec{u})$ are contracted with each other. The heavy-quark contractions yield precisely the kinetic and spin boundary-to-boundary propagators $\bar{S}_{\mathrm{hT}}^{\mathrm{kin}} = P_+ W_{\mathrm{T}}^{\mathrm{kin}}$ and $\bar{S}_{\mathrm{hT}}^{\mathrm{spin}} = P_+ W_{\mathrm{T}}^{\mathrm{spin}}$. So, the resulting expressions are

$$
\begin{aligned}
c_1^{\mathrm{kin}} &= \mathrm{Tr}\left[\gamma_5 \bar{S}_{\mathrm{T}}^{\dagger} \gamma_5 \cdot \mathcal{A} \cdot P_+ W_{\mathrm{T}}^{\mathrm{kin}} \cdot \mathcal{B}\right] && (3.137)\\
c_1^{\mathrm{spin}} &= \mathrm{Tr}\left[\gamma_5 \bar{S}_{\mathrm{T}}^{\dagger} \gamma_5 \cdot \mathcal{A} \cdot P_+ \left(\vec{\sigma} \cdot \vec{W}_{\mathrm{T}}^{\mathrm{spin}}\right) \cdot \mathcal{B}\right]. && (3.138)
\end{aligned}
$$

4. General Aspects of the SFCF Program

The SFCF program ("Schrödinger-functional correlation functions") is a measurement program, that means it is to compute the fermionic expectation value $\langle \ldots \rangle_{\mathrm{F}}$ of correlation function for a given gauge field $U_\mu(x)$. The gauge fields themselves are assumed to be produced by another program, the "update" program, which will not be discussed here.

In the past years the measurement of Schrödinger-functional correlation functions was done on the APE computers at DESY, Zeuthen [Bod03]. The APE computers were specially designed for lattice simulations of QCD. They consisted of many processors that could work in parallel. The lattice that was to be simulated could be divided into sub-lattices, and each sub-lattice could be handled by an individual processor. The processors were connected to each other as in a three-dimensional cubic lattice, so the necessary communication between neighbor sub-lattices could be performed. The program was written especially for the APE, in an own programming language ("TAO").

After more than twenty years the APE computers were finally switched off in spring 2011. However, the APE program could not simply be migrated to another platform, because it was written in TAO. Hence, our goal is to create a new, extended and more flexible measurement program for Schrödinger-functional correlations which can be run on a wide range of platforms and can be used for various physics applications. To do this, we use the C programming language [Ker88]. It is a rather low-level language and close to the hardware, which means that the hardware capabilities can be used more efficiently and the programs can run faster. In contrast to the TAO language, C is available for almost every platform. It does not support parallel programming natively, but different libraries and extensions exist for this purpose. In the SFCF program, we use the MPI standard ("message-passing interface") [Lus96, Gra05].

The fundamental structures of the SFCF program are based on the DD-HMC program. This affects e.g. the indexing of lattice sites, data types for gauge and spinor fields, the linear-algebra routines and also the Dirac operator. The DD-HMC program was written by Martin Lüscher et al. [Lü05]. It is an update and measurement program, but it employs periodic, not Schrödinger-functional boundary conditions. The adaptation to the Schrödinger functional and necessary extensions were done by Hubert Simma. My advisor Jochen Heitger and I were responsible for the implementation of the QCD correlation functions, e.g. f_{A}, f_1 and k_{V}. The routines for the calculation of HQET correlation functions were coded in collaboration with Michele Della Morte. The current solver implementation was provided by Andreas Athenodorou and Hubert Simma. In the future also an update routine may be developed that can be directly included in our program.

The subsequent sections will explain some basic aspects of the SFCF program which have relevance for the later chapters, like the way how lattice points are addressed and fields are stored. The next chapters 5 and 6 will describe the implementation of the QCD and HQET correlation functions.

4.1. Geometry

A basic aspect is how the lattice is represented in the program. The physical lattice is four-dimensional and its points x are labeled by four integer coordinates x_0, x_1, x_2 and x_3 within the limits

$$0 \le x_0 \le T \qquad 0 \le x_k \le L_k - 1 \qquad \text{for } k = 1, 2, 3. \tag{4.1}$$

The volume or the number of all lattice points is $V = (T+1) \cdot L_1 \cdot L_2 \cdot L_3$, the bulk volume (without the boundaries) is $V_\mathrm{B} = (T-1) \cdot L_1 \cdot L_2 \cdot L_3$, and the volume of one three-dimensional time slice is $V_3 = L_1 \cdot L_2 \cdot L_3$ (see section 2.3.1). In the SFCF program, these values are represented by preprocessor macros, which are listed in table 4.1. (The macros `TSLICE`, `BULK` and `VOLUME` are redundant, since they can be computed from `L1`, `L2`, `L3` and `TSIZE`.) A point that needs attention is that the program can be parallelized. Then the lattice is divided into several sub-lattices (see section 4.3), and the size of each local sub-lattice is only a fraction of the original lattice. The number of division in each space direction is given by the macros `NPROC1`, `NPROC2` and `NPROC3`. They can be used to obtain the size of the whole, global lattice. The time direction is never divided. However, in the following description we will assume that the program is in "single-node mode", i.e. it is not parallelized, and the lattice is not divided.

	SFCF expressions	
	local lattice	global lattice
T	`TSIZE`	`TSIZE`
L_1	`L1`	`L1*NPROC1`
L_2	`L2`	`L2*NPROC2`
L_3	`L3`	`L3*NPROC3`
V_3	`TSLICE`	`TSLICE*NPROC1*NPROC2*NPROC3`
V_B	`BULK`	`BULK*NPROC1*NPROC2*NPROC3`
V	`VOLUME`	`VOLUME*NPROC1*NPROC2*NPROC3`

Table 4.1.: Preprocessor macros that represent the geometrical properties of the local and global lattice in the SFCF program

Since there is only a finite number of lattice points, the four coordinates x_0, x_1, x_2, x_3 can also be merged into a single number that addresses each point in an unambiguous way. A common choice for this number is the lexicographical index

$$l_x = L_1 L_2 L_3 \cdot x_0 + L_2 L_3 \cdot x_1 + L_3 \cdot x_2 + x_3. \tag{4.2}$$

Its values range from 0 to $V - 1$.

But due to considerations about cache efficiency (see section 4.3) another number is used to address points in the SFCF program, which is simply called the index i_x. (This is a feature taken over from the DD-HMC program, see the introduction of this chapter.) Its relation to the coordinates or the lexicographical index is very involved, in addition it can depend on the type of field. If the macro `USE_GEOMETRY2` is defined, two different ways of indexing are used for gauge and spinor fields. The gauge index runs from 0 to $V - 1$ and addresses all lattice points, whereas the spinor index only runs from 0 to $V_\mathrm{B} - 1$ and addresses only the points in the bulk. In contrast, if the macro `USE_GEOMETRY1` is defined, the gauge geometry is used for all fields.

If one wants to know the index i_x belonging to a certain lexicographic index l_x, one can use the arrays `u_ipt` and `s_ipt`:

$$i_x = \begin{cases} \texttt{u\_ipt[}l_x\texttt{]} & \text{for gauge fields} \\ \texttt{s\_ipt[}l_x\texttt{]} & \text{for spinor fields.} \end{cases} \tag{4.3}$$

There are also arrays to find the neighbor of a point x in a certain direction:

- `u_iup[` i_x `][` μ `]` is the index of $x + \hat{\mu}$ (for gauge fields).
- `u_idn[` i_x `][` μ `]` is the index of $x - \hat{\mu}$ (for gauge fields).
- `s_iup[` i_x `][` μ `]` is the index of $x + \hat{\mu}$ (for spinor fields).
- `s_idn[` i_x `][` μ `]` is the index of $x - \hat{\mu}$ (for spinor fields).

However, if it is possible, the index should be used directly. This is possible, e.g., when one iterates over all points of the lattice or over the points of a specific time slice, since the index is guaranteed to obey these rules:

- *Even* points have indices between 0 and $V_{(\mathrm{B})}/2-1$, *odd* points have indices between $V_{(\mathrm{B})}/2$ and $V_{(\mathrm{B})}-1$. (A point is called even, if the sum of its coordinates $x_0 + x_1 + x_2 + x_3$ is even. Otherwise it is called odd.)
- Among the even and the odd points the indices are time-ordered, i.e. if x and y are both even (or both odd) and $x_0 < y_0$, then also $i_x < i_y$.

So, a loop over all, even and odd points of a spinor field in the time slice $x_0 = 3$ with `USE_GEOMETRY2` defined can look like:

```
for(ind=2*TSLICE/2; ind<3*TSLICE/2; ind++)
  /* ... */
for(ind=2*TSLICE/2+BULK/2; ind<3*TSLICE/2+BULK/2;
    ind++)
  /* ... */
```

4.2. Representation of Fields

The gauge field $U_\mu(x)$ is a global array in the SFCF program, which is declared in `src/include/global.h` as

```
su3_dp *PUD[VOLUME][4];
```

The first index is meant to be the index of a lattice point, and the second index the direction $\mu \in \{0, \ldots, 3\}$, where 0 is the time direction. The type `su3_dp` is a structure of nine complex numbers in double precision, which represent the components of an SU(3)-matrix. It is declared in `src/include/su3.h`:

```
typedef struct
{
  complex_dp c11,c12,c13,c21,c22,c23,c31,c32,c33;
} su3_dp;

typedef struct
{
  real_dp re,im;
} complex_dp;
```

So, the relation between the gauge field and the array `PUD` is

$$[U_\mu(x)]_{\alpha\beta} = \texttt{(*PUD[}i_x\texttt{][}\mu\texttt{]).c}\alpha\beta, \tag{4.4}$$

where i_x is the index in gauge-field geometry.

The spinor fields that are stored in one common, global array `PSD1e`, which is also declared in `src/include/global.h`:

```
spinor_dp *(*PSD1e);
```

It is indexed by the number n of the spinor field and the index i_x of a point. At the program's start, memory for a certain number of fields is allocated. This number is in the global variable `no_s`. So, the index n of a field may be in the range $\{0, \ldots, \texttt{no\_s}\}$.

`spinor_dp` is a structure of four components, which correspond to the four Dirac components. Each of which is a structure of three complex numbers, corresponding to the three values of the color index:

```
typedef struct
{
  complex_dp c1,c2,c3;
} su3_vector_dp;

typedef struct
{
  su3_vector_dp c1,c2,c3,c4;
} spinor_dp;

typedef struct
{
  su3_vector_dp c1,c2;
} weyl_dp;
```

There is also the structure type `weyl_dp`, which we will use to store the relativistic boundary-to-boundary propagator (section 5.1.1).

Actually, these type are not suited to hold fermionic fields, since their components are ordinary complex, commuting numbers and no anti-commuting Graßmann numbers. However, on the computer we will not handle Graßmann-valued fermionic fields directly but only such quantities as the relativistic propagator $\bar{S}(x)$, which really are fields of commuting complex numbers.

The value $\psi_n(x)$ of the n-th spinor field at point x is then obtained through

$$[\psi_n(x)]_{A\alpha} = \texttt{PSD1e[}n\texttt{][}i_x\texttt{].c}A\texttt{.c}\alpha. \tag{4.5}$$

4.3. Efficiency Enhancement

Several methods are applied in the SFCF program to improve its efficiency. Among these are a special way of indexing the points, the use of assembler code and the possibility to run the program in a parallel mode. These feature were already part of the DD-HMC code. In the following sections they are described in greater detail.

4.3.1. Indexing of Lattice Points

As described in section 4.1 points on the lattice are identified by an integer "index" between 0 and $V_{(\mathrm{B})} - 1$. The order in which the index addresses the points is meant to be more efficient than the order of the lexicographic index l_x for example. If we used the lexicographic index to loop over all points, we would start at $x = (0, 0, 0, 0)$ and go on in x_3-direction, until the point $(0, 0, 0, L_3 - 1)$ is reached. In the next step x_2 would be incremented, and again we would run through all points until $(0, 0, 1, L_3 - 1)$. This way is depicted in the left part of figure 4.1. Its disadvantage is that is does not use the processor's cache optimally. The cache stores recently used data, so it can be accessed quickly when it is used again and need not be fetched from the usual memory. But in lexicographic order it can take many steps, until a component of a field is needed again. For example when the Dirac operator is applied to a field, and the result at $(1,2,1,1)$ is to be computed, the field's value at the neighbor point $(1,1,1,1)$ is needed, but this point was accessed $L_2 \cdot L_3$ steps ago for the last time. Probably it is not in the cache any more, and must be retrieved from memory.

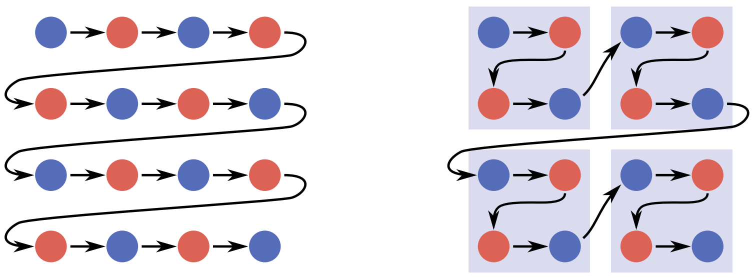

Figure 4.1.: Illustrations of the ways the lexicographic index (left) and the index in the SFCF program (right) run through the lattice, the single cache blocks are indicated by a blue background

For the index used in our program the lattice is divided into "cache blocks", illustrated in the right part of figure 4.1, and the index runs over all the points of one cache block, before it enters the next block. In this way we stay in a small region, before we go to another part of the lattice, and it is more likely that the field values we need have been accessed not too many steps ago and are still cached. Thus, we save the time to get them back from memory.

In fact the functions that compute the result of the Dirac operator do not run over all lattice points but only the "odd" points, i.e. the points whose coordinate sum is odd. (In the figures, the even points are colored blue and the odd points red.) The contributions to the neighbors of each odd point, who must be even then, are computed along with the result at the odd point. Therefore, the index handles even and odd points separately. The lower indices $0, \ldots, V_{(\mathrm{B})}/2 - 1$ address even points and the upper indices $V_{(\mathrm{B})}/2, \ldots, V_{(\mathrm{B})} - 1$ odd points.

4.3.2. SSE Instructions

SSE is a special set of processor instructions, which can be used to operate on vectors of floating-point or integer numbers. The acronym SSE means "Streaming SIMD Extensions", where SIMD stands for "single-instruction multiple-data", i.e. a single instruction can operate on multiple data. They are an extension of the instruction set of the IA-32 architecture. (SSE is documented in [Int].)

An example of the application of SSE instructions is the function `spinor_prod` (file `src/modules/linalg/linalg.c`). It computes the scalar product of two spinor fields. In each step it loads a half-spinor of type `weyl_dp` into the registers that were introduced with SSE. These are eight registers of 128 bit, so each of them can store two double-precision numbers of 64 bit or one complex number, and six of them suffice to hold a half-spinor. To compute the real part of the scalar product, the half-spinor is then multiplied by the corresponding half-spinor of the second field, the products are added, the imaginary part is computed in a similar way and the result is written back to memory.

In our program, the SSE instructions are used especially in the Dirac operator and in the solver routine, because they take the most time in a program run. They are included as assembler instructions in the program's source. Which version of SSE is supported by the processor can be specified at compile time through the macros `SSE`, `SSE2` or `SSE3`. If none of these is defined, plain C commands are used instead of SSE instructions.

4.3.3. Parallelization

The SFCF program can be run in a parallel mode. Then the lattice is divided into several sub-lattices. Each of them is handled by an own process, which can run on a separate processor. The communication between the processes is done via the Message Passing Interface (MPI), which defines functions to send data to other processes and receive data from them.[1] For example, this is necessary, when correlation functions are computed and the results must be summed over all lattice points. So the results from all processes must be collected and combined.

The parallel mode is triggered by the preprocessor macro `USE_MPI`. The macros `NPROC1`, `NPROC2` and `NPROC3` must be set to the number of divisions of the lattice in x_1-, x_2- and x_3-direction, respectively.

4.4. The Solver Routine

The solver function `solve` is a central part of the program, since most of the time is spent in this routine. It calculates the solution of the Dirac equation with a given source field, i.e. it applies the inverse Dirac operator to the source field.

The Hermitian Dirac operator Q can be interpreted as a big matrix acting on spinor fields. This matrix would have one row and one column for every lattice site, every Dirac and color component. For a 32^4-lattice, these are about 12 million rows and columns. Therefore, the determination of the exact solution of $Q\bar{S} = \eta$ is not feasible. It would take too much time ([Sto80, p. 536]). Furthermore, rounding errors limit the precision even of "exact" methods. Instead, the equation is usually solved by an iterative method, which determines a better approximation to the

[1]The implementation of MPI which I have used is OpenMPI, documented in [Ope].

exact solution in every step. After a preset tolerance is reached, the computation is aborted and the last obtained approximation is used.

At the moment we are using the standard conjugate-gradient method (CG) in the solver routine. This is not the most efficient algorithm, but it is relatively easy to implement and robust, i.e. it is guaranteed to converge. The implementation in the function `solve_cg` has been provided by Andreas Athenodorou and Hubert Simma. (The function `solve` is in fact a preprocessor macro that expands to `solve_cg` , so other solver routines could be employed by just changing this macro.) For completeness, the next section gives a brief description of how this algorithm works.

4.4.1. The Conjugate-Gradient Method

The conjugate-gradient method was introduced by Hestenes and Stiefel in 1952 [Hes52]. This description is based upon [Mon94, p. 410].

The purpose of the algorithm is the solution of a linear equation like

$$Ax = b \tag{4.6}$$

where x and b are complex vectors and A is a Hermitian and positive-definite matrix. This problem is translated into another form. The solution of the linear equation is equivalent to finding the minimum of the function

$$F(x, x^\dagger) = x^\dagger \cdot A \cdot x - b^\dagger \cdot x - x^\dagger \cdot b. \tag{4.7}$$

If A is Hermitian, the values of $F(x)$ will be real numbers. To show the equivalency, one can compute the derivative of F with respect to $x^\dagger$ and formally presume that x is constant, and arrives at the equation $Ax = b$ again.

The basic idea of the conjugate-gradient method is similar to the method of "steepest decent" [Sto80, p. 572], where approximations x_k to the exact solution are obtained iteratively by taking the last approximation x_{k-1} and moving it into the direction r_{k-1} of steepest decent of F, i.e. the negative gradient, and thus, minimizing F along the line from x_{k-1} in direction r_{k-1}. The negative gradient r_{k-1} is obtained through

$$r_{k-1} = \left(\frac{\partial F}{\partial x}\right)^\dagger = \frac{\partial F}{\partial x^\dagger} = b - Ax_{k-1}. \tag{4.8}$$

It is usually called the "residuum", because it denotes the difference between Ax_{k-1} and the right-hand side b.

However, the steepest-decent method leads to "zig-zag" paths, which may converge to the exact solution only slowly, because the next iteration step can spoil the minimization of F along the direction of the previous step. This is remedied in the CG method by replacing the r_k by directions s_k which are to be A-conjugate or A-orthogonal to each other, i.e. they must fulfill $s_k^\dagger A s_l = 0$ if $k \neq l$.

The CG algorithm achieves this as follows: The zeroth approximation is an arbitrary vector x_0, which may already be an approximate solution. It then calculates the residuum r_0, which is also the first direction s_0:

$$s_0 = r_0 = b - Ax_0. \tag{4.9}$$

In all subsequent steps the next approximation x_{k+1} is defined as

$$x_{k+1} = x_k + \alpha_k s_k \qquad \alpha_k = \frac{r_k^\dagger r_k}{s_k^\dagger A s_k}. \tag{4.10}$$

Again, α_k is chosen so that F becomes minimal along the path $x_k + \alpha_k s_k$. The new residuum r_{k+1} can be computed from the last one:

$$r_{k+1} = r_k - \alpha_k A s_k. \tag{4.11}$$

Then, the new direction s_{k+1} is

$$s_{k+1} = r_{k+1} + \beta_k s_k \qquad \beta_k = \frac{r_{k+1}^\dagger r_{k+1}}{r_k^\dagger r_k}. \tag{4.12}$$

It can be shown by induction that the requirement $s_k^\dagger A s_l = 0$ for $k \neq l$ is fulfilled by this choice.

The algorithm stops, when the residuum falls below a preset threshold, and the last approximation x_k is returned as result.

If the matrix A is not positive definite, the algorithm has to be slightly modified. The original problem $Ax = b$ is changed to $B^\dagger Ax = B^\dagger x$, where B is a matrix which is chosen so that $B^\dagger A$ is positive definite, e.g. one may choose $B = A$. The CG algorithm can be applied to the modified problem then.

The speed of convergence of the algorithm depends on the distribution of the eigenvalues of the matrix A. If the distance between any two eigenvalues is small, the CG method will quickly converge. It will be slower, if the eigenvalues are widespread. To improve the speed of convergence, the matrix A can be replaced by $\tilde{A} = CAD$ with two matrices C and D that are chosen appropriately. This is called "preconditioning". Then, solving $Ax = b$ is equivalent to solving $\tilde{A}\tilde{x} = Cb$ and multiplying the solution by D: $x = D\tilde{x}$.

5. Implementation of QCD Correlation Functions

We have explored different ways of computing the QCD, or relativistic, correlation functions. First, we wrote a dedicated function for each single CF, e.g. `f_a_rel` for f_{A}, `k_v_rel` for k_{V} and `f_1_rel` for f_1. These specific functions are efficient, but it is more work to maintain them, search for errors and add new correlations. The other way was to write one generic routine.[1] This generic function is less efficient, but in principle, it can be used to compute every correlation function. Finally, we have decided that we will use the generic routine. The time excess caused by the generic routine is not significant, because the time that is needed by the solver routine to compute the light propagator is much greater anyway. Nevertheless, the specific routines have been kept in the program, so their results and the times they need can be compared to the generic routines.

The first part of this chapter will describe the calculation of the relativistic propagator $\bar{S}(x)$. The second section will discuss the implementation of the specific CF routines and the third will treat the generic routine.

5.1. The Relativistic Propagator

5.1.1. How the Propagator is Stored

We use the spinor fields in `PSD1e` (see section 4.2) to store the relativistic propagators. However, a spinor field $\psi(x)_{A\alpha}$, with one color and one Dirac index, has fewer components than the propagator $\bar{S}(x)_{A\alpha,B\beta}$, which is a matrix in Dirac and color space and thus has a second index pair (B,β) with $3\times 4 = 12$ possible values. Therefore, we would need 12 spinor fields to store one propagator. The first field could hold the propagator with $B = 1$ and $\beta = 1$, the second one for $B = 1$ and $\beta = 2$ and so on. However, the propagator has a symmetry which we can use to reduce the number of fields to six: If we consider the definition (2.84) of the propagator, we can see that

$$\bar{S}(x) \cdot P_- = 0. \tag{5.1}$$

In the chiral representation this means for the components of $\bar{S}(x)$:

$$\bar{S}(x)_{A\alpha,3\beta} = -\bar{S}(x)_{A\alpha,1\beta} \qquad\qquad \bar{S}(x)_{A\alpha,4\beta} = -\bar{S}(x)_{A\alpha,2\beta}. \tag{5.2}$$

[1]I thank Michele Della Morte, who had the idea to write a generic function.

So, the components for $B = 3, 4$ differ from those for $B = 1, 2$ only by a minus sign and it is sufficient to store only the first two components.

The exact order of the propagator's components as we store it is:

Dirac index B	color index β	field index
1	1	$i_{\bar{S}} + 0$
1	2	$i_{\bar{S}} + 1$
1	3	$i_{\bar{S}} + 2$
2	1	$i_{\bar{S}} + 3$
2	2	$i_{\bar{S}} + 4$
2	3	$i_{\bar{S}} + 5$

Here, $i_{\bar{S}}$ is the index of the first spinor field that is used to store the propagator. So, in the source code, a specific component of the propagator is accessed via

$$\bar{S}(x)_{A\alpha,B\beta} \propto \texttt{PSD1e}[i_{\bar{S}} + 3 \cdot (B-1) + \beta - 1].[i_x].\texttt{c}A.\texttt{c}\alpha, \tag{5.3}$$

where i_x is the index of the point x in spinor geometry. I have not put an equal sign in the above formula, because usually some constant factors are left out in the computation of the propagator.

The backward propagator $\bar{R}(x)$ is stored in the same way as the forward propagator, the only difference is that here $\bar{R} \cdot P_+ = 0$ as can be seen from (2.88). So, we have to use the relation $\bar{R}(x)_{A\alpha,B\beta} = +\bar{R}(x)_{A\alpha,(B+2)\beta}$ (with $B = 1, 2$) to reconstruct the second half of components of $\bar{R}(x)$.

5.1.2. Computation of the Propagator

To compute the relativistic propagator, we start from its definition (2.84), which is repeated here with all indices:

$$\sum_{yB\beta} Q(x,y)_{A\alpha,B\beta} \cdot \bar{S}(y)_{B\beta,C\gamma} = \frac{\tilde{c}_{\mathrm{t}}}{\sqrt{V_3}} \cdot \delta_{x_0,1} \cdot [U_0(0,\vec{x})^\dagger]_{\alpha\gamma} \cdot [\gamma_5 P_+]_{AC}. \tag{5.4}$$

The indices C and γ of the propagator are not contracted with the operator Q, so we can compute the propagator for every combination of C and γ independently. As we have seen in the previous section, it is sufficient to consider $C = 1, 2$ and $\gamma = 1, 2, 3$. For each of the six possible combinations, the calculation of the propagator consists of two steps: First, a spinor field $\eta(x)$ that equals the right-hand side with the current values of C and γ must be prepared. Second, the equation $Q\bar{S}_{**,C\gamma} = \eta$ must be solved.

The first step, the initialization of the right-hand side η is done by the function `init_rhs`. Its design is not described in detail here, because for the greatest part it is the straight-forward assignment of the right-hand side to the specified spinor field. A detail should be mentioned: The right-hand side as it is produced by `init_rhs` is not exactly the one of equation 2.84. `init_rhs` leaves out a constant factor of

$\tilde{c}_t/2$. This is taken into account only in the very end, when the computed correlation functions are normalized, because it would need many operations to multiply a whole spinor field by this factor.

The more complex and time-consuming part is the second step, the inversion of the Hermitian Dirac operator Q. This is done by an iterative solver. At the moment, the function `solve` implements a "conjugate-gradient" (CG) solver. This may be changed in future versions and other, more efficient algorithms may be included, e.g. the BiCG method [Pre92, p. 85] which was used by the APE program, too. Section 4.4 describes the current solver routine in greater detail.

The source code for the computation of the forward propagator $\bar{S}$ may look like listing 5.1.

Listing 5.1: Calculation of relativistic propagators

```
/* This and the five subsequent spinor fields will hold
 * the propagator. */
int prop_field;
/* This spinor field holds the right-hand side \eta of
 * the definition. */
int rhs_field;
/* This and the three subsequent spinor fields are used
 * by the solver routine internally. */
int work_field;
int dirac, color;
/* This stores parameters for the solver and about the
 * considered quark flavor (kappa, theta). */
param_solver_t parameters;
int status;
/* to return the solver's status */

/* ... */

for(dirac=1; dirac<=2; dirac++)
{
  for(color=1; color<=3; color++)
  {
    /* 1st step: Initialize the right-hand side
     * \eta. */
    init_rhs(rhs_field, dirac, color, 0);

    /* 2nd step: Solve Q \bar S = \eta. */
    solve(parameters, work_field, work_field+3,
          rhs_field, prop_field+3*(dirac-1)+(color-1),
            &status);
    }
  }

  /* The relativistic propagator for the quark flavor
   * described by parameters is now in the spinor fields
   * prop_field,..., prop_field+5. */
```

For a backward propagator, only the last argument of `init_rhs` must be changed from `0` to `1`. It will then prepare the right-hand side η of the definition (2.88) of a backward propagator. The solution of $Q\bar{R} = \eta$ by the function `solve` is done in the same way as above then.

5.2. The Relativistic Boundary-to-Boundary Propagator

For the computation of boundary-to-boundary correlation like f_1 and k_1, a special boundary-to-boundary propagator $\bar{S}_\mathrm{T}$ has been defined in (2.90). It is a sum of the boundary-to-bulk propagator $\bar{S}(x)$ over the points x in the next-to-last time slice at $x_0 = T - 1$. In this sum, the boundary-to-bulk propagator is “transported” to the last time slice at T by multiplying it by the appropriate link matrices. This computation is performed in the function `boundary_boundary_propagator`. It is a direct implementation of the definition (2.90), but leaves out the constant factor of $\tilde{c}_\mathrm{t}/2$.

From its definition, we can see that $\bar{S}_\mathrm{T}$ has two symmetries:

$$\bar{S}_\mathrm{T} \cdot P_- = 0 \qquad\qquad P_- \cdot \bar{S}_\mathrm{T} = 0. \tag{5.5}$$

It inherits the first one from the boundary-to-bulk propagator $\bar{S}$, the second one is due to the P_+ matrix appearing in its definition. At component level, this means

$$\left(\bar{S}_\mathrm{T}\right)_{A\alpha,(B+2)\beta} = -\left(\bar{S}_\mathrm{T}\right)_{A\alpha,B\beta} \qquad \text{for } B = 1, 2, \tag{5.6}$$

$$\left(\bar{S}_\mathrm{T}\right)_{(A+2)\alpha,B\beta} = -\left(\bar{S}_\mathrm{T}\right)_{A\alpha,B\beta} \qquad \text{for } A = 1, 2. \tag{5.7}$$

So, only a quarter of all Dirac components needs to be saved, e.g. the ones with $A, B \in \{1, 2\}$. This does not save much memory, since $\bar{S}_\mathrm{T}$ is not a field, but only a single 12×12-matrix ($12 = 4$ Dirac components $\times$ 3color components), but to avoid redundancy I have done this anyway: The boundary-to-boundary propagator is stored in an array of six variables of type `weyl_dp`, which may be declared as

```
  weyl_dp bbprop[6];
```

The type `weyl_dp` represents Weyl spinors, which have only two spinor components in contrast to Dirac spinors, which have four. Hence, they are able to represent $\bar{S}_\mathrm{T}$, too, although the components of $\bar{S}_\mathrm{T}$ that we store in them do not actually constitute a Weyl spinor.

The exact storage order of the components of $\bar{S}_\mathrm{T}$ is very similar to the order in which the components of $\bar{S}$ are stored in six spinor fields. The relation between $\bar{S}_\mathrm{T}$ and its representation `bbprop` on the computer is

$$\left(\bar{S}_\mathrm{T}\right)_{A\alpha,B\beta} \propto \texttt{bbprop}[3 \cdot (B-1) + (\beta - 1)].\texttt{c}A.\texttt{c}\alpha. \tag{5.8}$$

Again, a proportionality sign has been used, because the computed propagator may differ from $\bar{S}_\mathrm{T}$ as defined here by a constant factor.

To compute the boundary-to-boundary propagator from the boundary-to-bulk propagator, the listing 5.1 for the boundary-to-bulk propagator can be extended by the lines in listing 5.2.

Listing 5.2: Computing the relativistic boundary-to-boundary propagator

```
/* ... */

weyl_dp  bbprop[6];  /* stores the boundary-to-boundary
                      * propagator */

/* ... */

boundary_boundary_propagator(prop_field, bbprop);

/* bbprop now holds the boundary-to-boundary
 * propagator. */
```

5.3. The Specific Functions

The functions `f_a_rel`, `f_p_rel`, `k_t_rel` and `k_v_rel` are specialized in computing the boundary-to-bulk correlations $f_\mathrm{A}(x_0)$, $f_\mathrm{P}(x_0)$, $k_\mathrm{T}(x_0)$, $k_\mathrm{V}(x_0)$ and their backward versions $g_\mathrm{A}(x_0)$, $g_\mathrm{P}(x_0)$, $l_\mathrm{T}(x_0)$ and $l_\mathrm{V}(x_0)$. There are also functions for the boundary-to-boundary correlations f_1 and k_1, which are called `f_1_rel` and `k_1_rel`.

These specific functions are the first ones I have written and resemble the corresponding routines that were used on the APE. Since each of them is to handle only one type of correlation function, they are adapted to the specific product of Dirac matrices that is used in these correlations and are several times faster than the generic function (see section 5.5).

The following sections will describe how the specific functions work and how correlation functions can be computed through them. The routines for f_{A} and f_{P} and their application are very similar to each other, since their definitions only differ by a γ_0 matrix, so both are explained in one section. Likewise, the descriptions of k_{T} and k_{V} are merged. The functions for the boundary-to-boundary correlations f_1 and k_1 are in principle similar to the routines for the boundary-to-bulk correlations, but they are simpler, because the additional symmetry of the propagator $\bar{S}_{\mathrm{T}}$ is exploited and there is no summation over $\vec{x}$. Therefore, they are only briefly described in the last section.

But first, it is explained how the specific functions are called.

5.3.1. Usage of the Specific Functions

The way the specific functions are called is the same for all of them. After the propagators have been computed, the correlations can be computed in the following way:

```c
#include "cf_rel.h"

/* ... */

/* Indices of the fields in which the first and second
 * (forward) propagator is stored. */
int prop_1, prop_2;
/* First and second boundary-to-boundary propagator */
weyl_dp bbprop_1, bbprop_2;
/* Output arrays to which the computed correlations are
 * written. */
complex_dp out_f_a_rel[TSIZE-1], out_f_p_rel[TSIZE-1],
           out_k_t_rel[TSIZE-1], out_k_v_rel[TSIZE-1],
           out_f_1_rel[1], out_k_1_rel[1];

/* ... */

f_a_rel(prop_1, prop_2, out_f_a_rel);
f_p_rel(prop_1, prop_2, out_f_p_rel);
k_t_rel(prop_1, prop_2, out_k_t_rel);
k_v_rel(prop_1, prop_2, out_k_v_rel);
f_1_rel(bbprop_1, bbprop_2, out_f_1_rel);
k_1_rel(bbprop_1, bbprop_2, out_k_1_rel);

/* Results are in the out_... arrays now. */
```

The header file `cf_rel.h` provides the declarations of the specific functions as well as of the generic functions.

The corresponding backward correlations g_{A}, g_{P}, l_{T} and l_v are obtained in the same way, if `prop_1` and `prop_2` are the indices of the backward propagators instead of the forward propagators (see section 5.1.2 about the computation of the propagator).

If the results are to match those from the APE (see section 2.3.8), they must be normalized differently. The values that are written to the output arrays must be multiplied by the numbers given in table 5.1. (Here, it is assumed that the propagators have been computed as described in section 5.1.2. If the propagators are normalized differently, other values for the normalization of correlation functions would have to be used, too.)

correlation function	additional normalization
f_{A}	$\tilde{c}_{\mathrm{t}}^2/(4V_3)$
f_{P}	$\tilde{c}_{\mathrm{t}}^2/(4V_3)$
k_{T}	$\tilde{c}_{\mathrm{t}}^2/(12V_3)$
k_{V}	$\tilde{c}_{\mathrm{t}}^2/(12V_3)$
f_1	$\tilde{c}_{\mathrm{t}}^4/(8V_3^2)$
k_1	$\tilde{c}_{\mathrm{t}}^4/(24V_3^2)$

correlation function	additional normalization
g_{A}	$-\tilde{c}_{\mathrm{t}}^2/(4V_3)$
g_{P}	$\tilde{c}_{\mathrm{t}}^2/(4V_3)$
l_{T}	$\tilde{c}_{\mathrm{t}}^2/(12V_3)$
l_{V}	$-\tilde{c}_{\mathrm{t}}^2/(12V_3)$

Table 5.1.: Additional factors that are needed to normalize the correlation functions from the specific functions in the same way as on the APE

5.3.2. The Correlations $f_{\mathrm{A}}(x_0)$ and $f_{\mathrm{P}}(x_0)$

The routines `f_a_rel` and `f_p_rel`, which compute the correlation functions f_{A} and f_{P} are based on equations (2.113) and (2.117), where f_{A} and f_{P} are written in terms of the relativistic propagators $\bar{S}$ by means of Wick's theorem. As indicated in these equations, the two propagators appearing there may be for different quark flavors i and j. However, the average over the gauge field $\langle \ldots \rangle_G$ is not performed within these functions, but by external programs. `f_a_rel` and `f_p_rel` only evaluate the fermionic expectation value for the given gauge field.

I will first describe the function `f_a_rel` in the following text, but the differences to `f_p_rel` are small and are explained afterwards.

The correlation f_{A} as expressed in (2.113) (without gauge average) may be rewritten at the level of Dirac and color components to better match the source code of `f_a_rel`:

$$f_{\mathrm{A}}^{ij}(x_0) = -\frac{1}{2}\sum_{\vec{x}} \mathrm{Tr}\left(\bar{S}^{j\dagger}(x)\gamma_0\bar{S}^i(x)\right) \tag{5.9}$$

$$= -\frac{1}{2}\sum_{\vec{x}}\sum_{A,B=1}^{4}\sum_{\alpha,\beta=1}^{3}\left[\bar{S}^j(x)\right]^*_{A\alpha,B\beta}\left[\gamma_0\bar{S}^i(x)\right]_{A\alpha,B\beta}. \tag{5.10}$$

Now we make use of the symmetry $\bar{S}(x)_{A\alpha,(B+2)\beta} = -\bar{S}(x)_{A\alpha,B\beta}$ (see (5.2)):

$$\begin{aligned}
&= -\frac{1}{2}\sum_{\vec{x}}\sum_{A=1}^{4}\sum_{B=1}^{2}\sum_{\alpha,\beta=1}^{3}\left[\bar{S}^j(x)\right]^*_{A\alpha,B\beta}\left[\gamma_0\bar{S}^i(x)\right]_{A\alpha,B\beta}\\
&\quad -\frac{1}{2}\sum_{\vec{x}}\sum_{A=1}^{4}\sum_{B=1}^{2}\sum_{\alpha,\beta=1}^{3}\left[\bar{S}^j(x)\right]^*_{A\alpha,(B+2)\beta}\left[\gamma_0\bar{S}^i(x)\right]_{A\alpha,(B+2)\beta} \qquad (5.11)\\
&= -\frac{1}{2}\sum_{\vec{x}}\sum_{A=1}^{4}\sum_{B=1}^{2}\sum_{\alpha,\beta=1}^{3}\left[\bar{S}^j(x)\right]^*_{A\alpha,B\beta}\left[\gamma_0\bar{S}^i(x)\right]_{A\alpha,B\beta}\\
&\quad -\frac{1}{2}\sum_{\vec{x}}\sum_{A=1}^{4}\sum_{B=1}^{2}\sum_{\alpha,\beta=1}^{3}\left[-\bar{S}^j(x)\right]^*_{A\alpha,B\beta}\left[-\gamma_0\bar{S}^i(x)\right]_{A\alpha,B\beta} \qquad (5.12)\\
&= \sum_{\vec{x}}\sum_{A=1}^{4}\sum_{B=1}^{2}\sum_{\alpha,\beta=1}^{3}\left[\bar{S}^j(x)\right]^*_{A\alpha,B\beta}\left[\gamma_0\bar{S}^i(x)\right]_{A\alpha,B\beta}. \qquad (5.13)
\end{aligned}$$

The use of this symmetry will be treated in another, more formal way for the generic function in the next section 5.4.

In the chiral representation, the term $[\gamma_0\bar{S}^i(x)]$ is obtained from $\bar{S}^i(x)$, when the upper and lower Dirac components are exchanged:

$$\begin{pmatrix}[\gamma_0\bar{S}^i(x)]_{1\alpha,B\beta}\\ [\gamma_0\bar{S}^i(x)]_{2\alpha,B\beta}\\ [\gamma_0\bar{S}^i(x)]_{3\alpha,B\beta}\\ [\gamma_0\bar{S}^i(x)]_{4\alpha,B\beta}\end{pmatrix} = \begin{pmatrix} & & 1 & \\ & & & 1\\ 1 & & & \\ & 1 & & \end{pmatrix}\cdot\begin{pmatrix}\bar{S}^i(x)_{1\alpha,B\beta}\\ \bar{S}^i(x)_{2\alpha,B\beta}\\ \bar{S}^i(x)_{3\alpha,B\beta}\\ \bar{S}^i(x)_{4\alpha,B\beta}\end{pmatrix} = \begin{pmatrix}\bar{S}^i(x)_{3\alpha,B\beta}\\ \bar{S}^i(x)_{4\alpha,B\beta}\\ \bar{S}^i(x)_{1\alpha,B\beta}\\ \bar{S}^i(x)_{2\alpha,B\beta}\end{pmatrix}. \tag{5.14}$$

Now we can contrast the equation (5.13) with the source code of the function `f_a_rel`. The relevant part of the source code are shown in listing 5.3. This part computes $f_{\mathrm{A}}(x_0)$ for a single value of x_0 and writes it to the element $(x_0 - 1)$ of the output array `out`. The following list explains what is done at each line of the code:

lines 1–6 The summation and count arrays are set to zero. Their purpose is described at line 30.

lines 8–11 These two loops correspond to the sum over all points of the current timeslice ($\sum_{\vec{x}}$). The outer loop is executed only twice, for `i_eo` = 0 and i_eo $= 1$. In the first case, the limits of the inner loop are set so that `ind` runs over the indices of all *even* points in the timeslice at `x0` $= x_0$, in the second case it runs over the indices of all *odd* points. This double-loop structure is necessary, because even and odd points do not occupy one contiguous range of indices (see section 4.1).

line 13 The variable `dir_col` corresponds to the combined indices B and β of the propagators. So, this loop represents the sum $\sum_B \sum_\beta$.

lines 17–18 `psi_1` and `psi_2` are set to the addresses of the propagators with the indices B and β at the point x, which has the index `ind`. So, `*psi_1` $\propto \bar{S}^i(x)_{**,B\beta}$ and `*psi_2` $\propto \bar{S}^j(x)_{**,B\beta}$.

lines 20–28 These lines compute the product of the propagators and sum over A and α: $\sum_{A\alpha}[\bar{S}^j(x)]^*_{A\alpha,B\beta}[\gamma_0\bar{S}^i(x)]_{A\alpha,B\beta}$. The summation over α is done by the preprocessor macros `_vector_prod_re` and `_vector_prod_im`. They take two complex, three-dimensional vectors as arguments and compute the real, and the imaginary part of their scalar product. The summation over A is done explicitly. To take into account the γ_0 matrix, the components of the propagator `psi_1` are interchanged as shown in (5.14). The result is added to `sum[0]`, which is the primary variable for summation, and `cnt[0]`, which counts the summations, is incremented (see next point).

lines 30–38 In the specific routines, the summations are done in a particular way to avoid rounding errors.[2] The basic idea is that the errors can be kept small, if only numbers of similar magnitude are added. Therefore, the summation is done at several levels (the number of levels is `SUM_LEVEL`): First, the results are added to the lowest level `sum[0]` (see previous lines). The number of summations is counted in `cnt[0]`. After `SUM_BLOCK` summands, the sum from `sum[0]` is added to the next level in `sum[1]`, and `sum[0]` and `cnt[0]` are reset to zero, whereas `cnt[1]` is advanced by one. When `cnt[1]` itself has counted `SUM_BLOCK` summations, the value in `sum[1]` is transfered to `sum[2]` and so on.

We may call this a hierarchical summation, because at each level, `SUM_BLOCK` results of the next-lower level are combined. Thus, if there are enough summation levels, a number is added to a previous sum of at most `SUM_LEVEL` $- 1$ numbers of presumably equal magnitude.

lines 43–49 In these lines the different summation levels are combined to obtain the total local sum, i.e. the sum over all points that belong to the local lattice.

lines 51–52 Here, the local sums of all processes are added to get the final result for $f_{\mathrm{A}}(x_0)$ (up to constant factors). To this purpose, the function `mpc_gsum_d` is

[2] This technique has already been used in the routines of the DD-HMC code by Martin Lüscher and Björn Leder.

employed, which uses the MPI routines `MPI_Reduce` and `MPI_Bcast` to perform the global summation.

This is the way the equation (5.13) is implemented in the SFCF program. The function `f_p_rel`, which computes f_{P}, is identical, except for the lines 20–28, where the spinor components of `psi_1` are not interchanged.

Listing 5.3: Some parts of the source code of the function `f_a_rel` (from file `src/modules/cf_rel/f_a_rel.c`)

```
 1  for(i=0; i<SUM_LEVEL; i++)
 2  {
 3    sum[i].re = 0.0;
 4    sum[i].im = 0.0;
 5    cnt[i] = 0;
 6  }
 7
 8  for(i_eo=0; i_eo<2; i_eo++)
 9  {
10    for(ind=(x0-1)*TSLICE/2+i_eo*BULK/2;
11        ind<x0*TSLICE/2+i_eo*BULK/2; ind++)
12    {
13      for(dir_col=0; dir_col<6; dir_col++)
14      {
15        spinor_dp *psi1, *psi2;
16
17        psi1 = PSD1e[prop_1+dir_col]+ind;
18        psi2 = PSD1e[prop_2+dir_col]+ind;
19
20        sum[0].re += +_vector_prod_re(psi2->c1, psi1->c3)
21                     +_vector_prod_re(psi2->c2, psi1->c4)
22                     +_vector_prod_re(psi2->c3, psi1->c1)
23                     +_vector_prod_re(psi2->c4, psi1->c2);
24        sum[0].im += +_vector_prod_im(psi2->c1, psi1->c3)
25                     +_vector_prod_im(psi2->c2, psi1->c4)
26                     +_vector_prod_im(psi2->c3, psi1->c1)
27                     +_vector_prod_im(psi2->c4, psi1->c2);
28        cnt[0]++;
29
30        for(i=0; (i<SUM_LEVEL-1)&&(cnt[i]>=SUM_BLOCK); i++)

31        {
32          sum[i+1].re += sum[i].re;
33          sum[i+1].im += sum[i].im;
```

```
        cnt[i+1]++;
        sum[i].re = 0.0;
        sum[i].im = 0.0;
        cnt[i] = 0;
      }
    }
  }
}

out_local[x0-1].re = 0.0;
out_local[x0-1].im = 0.0;
for(i=0; i<SUM_LEVEL; i++)
{
  out_local[x0-1].re += sum[i].re;
  out_local[x0-1].im += sum[i].im;
}

mpc_gsum_d(&out_local[x0-1].re, &out[x0-1].re, 1);
mpc_gsum_d(&out_local[x0-1].im, &out[x0-1].im, 1);
```

5.3.3. The Correlations $k_{\mathrm{T}}(x_0)$ and $k_{\mathrm{V}}(x_0)$

In equations (2.130) and (2.124), the definitions of the vector correlation functions $k_{\mathrm{T}}(x_0)$ and $k_{\mathrm{V}}(x_0)$ have already been written in terms of the spinor components of the relativistic propagators. Also the symmetry of the propagator ($\bar{S} \cdot P_- = 0$) has been used to simplify these expressions. Thus, these equations already reflect very closely how the functions `k_t_rel` and `k_v_rel` are implemented. Here, they are shown again, with explicit color indices:

$$\begin{aligned}
k_{\mathrm{T}}^{ij}(x_0) = \frac{1}{3}\sum_{\vec{x}}\sum_{\alpha,\beta=1}^{3}&\\
&\Big[\bar{S}^j(x)^*_{1\alpha,1\beta}(-2\bar{S}^i(x)_{2\alpha,2\beta} - \bar{S}^i(x)_{1\alpha,1\beta}\big) + \bar{S}^j(x)^*_{1\alpha,2\beta}\bar{S}^i(x)_{1\alpha,2\beta}\\
&\bar{S}^j(x)^*_{2\alpha,2\beta}\big(-2\bar{S}^i(x)_{1\alpha,1\beta} - \bar{S}^i(x)_{2\alpha,2\beta}\big) + \bar{S}^j(x)^*_{2\alpha,1\beta}\bar{S}^i(x)_{2\alpha,1\beta}\\
&\bar{S}^j(x)^*_{3\alpha,1\beta}\big(-2\bar{S}^i(x)_{4\alpha,2\beta} - \bar{S}^i(x)_{3\alpha,1\beta}\big) + \bar{S}^j(x)^*_{3\alpha,2\beta}\bar{S}^i(x)_{3\alpha,2\beta}\\
&\bar{S}^j(x)^*_{4\alpha,2\beta}\big(-2\bar{S}^i(x)_{3\alpha,1\beta} - \bar{S}^i(x)_{4\alpha,2\beta}\big) + \bar{S}^j(x)^*_{4\alpha,1\beta}\bar{S}^i(x)_{4\alpha,1\beta}\Big] \qquad (5.15)\\
k_{\mathrm{V}}^{ij}(x_0) = \frac{1}{3}\sum_{\vec{x}}\sum_{\alpha,\beta=1}^{3}&\\
&\Big[\bar{S}^j(x)^*_{1\alpha,1\beta}\big(-2\bar{S}^i(x)_{4\alpha,2\beta} - \bar{S}^i(x)_{3\alpha,1\beta}\big) + \bar{S}^j(x)^*_{1\alpha,2\beta}\bar{S}^i(x)_{3\alpha,2\beta}
\end{aligned}$$

$$\begin{aligned}
&\bar{S}^j(x)^*_{2\alpha,2\beta}\big(-2\bar{S}^i(x)_{3\alpha,1\beta}-\bar{S}^i(x)_{4\alpha,2\beta}\big)+\bar{S}^j(x)^*_{2\alpha,1\beta}\bar{S}^i(x)_{4\alpha,1\beta}\\
&\bar{S}^j(x)^*_{3\alpha,1\beta}\big(-2\bar{S}^i(x)_{2\alpha,2\beta}-\bar{S}^i(x)_{1\alpha,1\beta}\big)+\bar{S}^j(x)^*_{3\alpha,2\beta}\bar{S}^i(x)_{1\alpha,2\beta}\\
&\bar{S}^j(x)^*_{4\alpha,2\beta}\big(-2\bar{S}^i(x)_{1\alpha,1\beta}-\bar{S}^i(x)_{2\alpha,2\beta}\big)+\bar{S}^j(x)^*_{4\alpha,1\beta}\bar{S}^i(x)_{2\alpha,1\beta}\Big]
\end{aligned} \tag{5.16}$$

Because these correlation functions involve a sum over the indices of the γ_k matrices, the above equations are more involved than for f_{A} and f_{P}. However, they are quite similar to each other, since the difference between the definitions of k_{T} and k_{V} is only an additional γ_0 matrix as for f_{A} and f_{P}.[3] Thence, the equations are identical up to the first Dirac index of the propagator $\bar{S}^i$, and in the following text, only the source code for the computation of k_{T} is described in detail.

An excerpt from the function `k_t_rel` is displayed in listing 5.4. It is the body of the double loop over `i_eo` and `ind`, and corresponds to the lines 13 to 38 in listing 5.3. The double loop itself and the source code outside of it are the same in `f_a_rel` and `k_t_rel`.

The list below explains what is done by the commands in listing 5.4:

line 1 This loop over `col` corresponds to the sum over the second color index β. In listing 5.3 a loop over the variable `dir_col`, which represents also the second Dirac index B, was written here instead, but in equation (5.15) the second Dirac index is not simply summed over, it is entangled with the first Dirac index A.

lines 6–9 As shortcuts, the variables `si1`, `si2`, `sj1` and `sj2` are defined. The second letters `i` and `j` stand for the propagators $\bar{S}^i$ and $\bar{S}^j$, the numbers `1` and `2` are the value of the second Dirac index B. The second color index β is given by `col` (see above). So, for example, `si2` is a pointer to the place where $\bar{S}^i(x)_{**,B\beta}$ is stored.

lines 11–18 In these lines, the terms in round brackets in equation (5.15) are assigned to the variables `v1`, `v2`, `v3` and `v4`. For example, `v1` holds the term $(-2\bar{S}^i(x)_{2*,2\beta}-\bar{S}^i(x)_{1*,1\beta})$, which will be multiplied by $\bar{S}^j(x)^*_{1*,1\beta}$.

lines 20–36 The actual multiplication is done here, once for the real part of k_{T} and once for its imaginary part. There are two types of products: In the first line, `_vector_prod_re(sj1->c1, v1)` computes the scalar product of $\bar{S}^j(x)_{1*,1\beta}$ and the bracket term from the previous paragraph, which has been assigned to `v1`. In the second line, `_vector_prod_re(sj2->c1, si2->c1)` simply computes the scalar product of $\bar{S}^j(x)_{1*,2\beta}$ and $\bar{S}^i(x)_{1*,2\beta}$. The order of terms is the same as in eq. (5.15). (The macros `_vector_prod_re` and `_vector_prod_im` have been described in section 5.3.2 in the comments on listing 5.3.)

[3] k_{V} includes a γ_k matrix between the bulk fields, whereas k_{T} includes a $\mathrm{i}\sigma_{k0}$, which can be written as $\mathrm{i}\sigma_{k0}=\mathrm{i}\cdot\mathrm{i}/2\cdot[\gamma_k,\gamma_0]=-1/2\cdot(-2\gamma_0\gamma_k)=\gamma_0\gamma_k$. (The fact that different gamma matrices anti-commute has been used.)

lines 38–46 The hierarchical summation is performed to avoid round-off errors (see section 5.3.2).

As already mentioned above, the function `k_v_rel` has the same structure. Only the second Dirac indices of the propagator $\bar{S}^i$ in lines 20–36 and in the definition of the auxiliary variables in lines 11–18 are altered.

Listing 5.4: Excerpt from the function `k_t_rel` (file `src/modules/cf_rel/k_t_rel.c`)

```
 1  for(col=0; col<3; col++)
 2  {
 3    spinor_dp *si1, *si2, *sj1, *sj2;
 4    su3_vector_dp v1, v2, v3, v4;
 5
 6    si1 = PSD1e[prop_1+0*3+col]+ind;
 7    si2 = PSD1e[prop_1+1*3+col]+ind;
 8    sj1 = PSD1e[prop_2+0*3+col]+ind;
 9    sj2 = PSD1e[prop_2+1*3+col]+ind;
10
11    _vector_mul(v1, -2, si2->c2);
12    _vector_sub_assign(v1, si1->c1);
13    _vector_mul(v2, -2, si1->c1);
14    _vector_sub_assign(v2, si2->c2);
15    _vector_mul(v3, -2, si2->c4);
16    _vector_sub_assign(v3, si1->c3);
17    _vector_mul(v4, -2, si1->c3);
18    _vector_sub_assign(v4, si2->c4);
19
20    sum[0].re += +_vector_prod_re(sj1->c1, v1)
21                 +_vector_prod_re(sj2->c1, si2->c1)
22                 +_vector_prod_re(sj2->c2, v2)
23                 +_vector_prod_re(sj1->c2, si1->c2)
24                 +_vector_prod_re(sj1->c3, v3)
25                 +_vector_prod_re(sj2->c3, si2->c3)
26                 +_vector_prod_re(sj2->c4, v4)
27                 +_vector_prod_re(sj1->c4, si1->c4);
28    sum[0].im += +_vector_prod_im(sj1->c1, v1)
29                 +_vector_prod_im(sj2->c1, si2->c1)
30                 +_vector_prod_im(sj2->c2, v2)
31                 +_vector_prod_im(sj1->c2, si1->c2)
32                 +_vector_prod_im(sj1->c3, v3)
33                 +_vector_prod_im(sj2->c3, si2->c3)
34                 +_vector_prod_im(sj2->c4, v4)
```

```
                   +_vector_prod_im(sj1->c4, si1->c4);
  cnt[0]++;

  for(i=0; (i<SUM_LEVEL-1)&&(cnt[i]>=SUM_BLOCK); i++)
  {
    sum[i+1].re += sum[i].re;
    sum[i+1].im += sum[i].im;
    cnt[i+1]++;
    sum[i].re = 0.0;
    sum[i].im = 0.0;
    cnt[i] = 0;
  }
}
```

5.3.4. The Correlations f_1 and k_1

The boundary-to-boundary correlation functions f_1 and k_1 are computed by the functions `f_1_rel` and `k_1_rel`. `f_1_rel` is very similar to the functions `f_a_rel` and `f_p_rel`, and `k_1_rel` is very similar to `k_t_rel` and `k_v_rel`, but there are two main differences: There is no sum over $\vec{x}$ and the boundary-to-boundary propagator has the additional symmetry $P_-\bar{S}_\mathrm{T} = 0$.

We will first have a look at `f_1_rel`. By means of Wick's theorem, the definition of f_1 was rewritten in terms of the boundary-to-boundary propagators in equation (2.121). Here, this equation is repeated with explicit Dirac and color indices, but without the gauge-field expectation value:

$$f_1^{ij} = \frac{1}{2} \sum_{A,B=1}^{4} \sum_{\alpha,\beta=1}^{3} \left[\bar{S}_\mathrm{T}^j\right]^*_{A\alpha,B\beta} \cdot \left[\bar{S}_\mathrm{T}^i\right]_{A\alpha,B\beta} . \tag{5.17}$$

If we employ the symmetries of $\bar{S}_\mathrm{T}$ (see equation (5.6)), we can write

$$= 2 \sum_{A,B=1}^{2} \sum_{\alpha,\beta=1}^{3} \left[\bar{S}_\mathrm{T}^j\right]^*_{A\alpha,B\beta} \cdot \left[\bar{S}_\mathrm{T}^i\right]_{A\alpha,B\beta} . \tag{5.18}$$

This is the equation that has been implemented in `f_1_rel`, whose source code is shown in listing 5.5. Only the constant factor of 2 in front of the sum has been omitted. Since most of its elements have already been explained in the previous sections. I will give only a brief description here:

line 12 This loop corresponds to the sums $\sum_{B=1}^{2}\sum_{\beta=1}^{3}$.

lines 16–17 The addresses of $[\bar{S}_\mathrm{T}^i]_{**,B\beta}$ and $[\bar{S}_\mathrm{T}^j]_{**,B\beta}$ are assigned to `weyl1` and `weyl2`.

lines 19–22 The scalar product of the two Weyl spinors is calculated, corresponding to $\sum_{A=1}^{2}\sum_{\alpha=1}^{3}[\bar{S}^j_{\mathrm{T}}]^*_{A\alpha,B\beta}\cdot[\bar{S}^i_{\mathrm{T}}]_{A\alpha,B\beta}$.

Listing 5.5: The function `f_1_rel` (file `src/modules/cf_rel/f_1_rel.c`)

```
 1  void f_1_rel(const weyl_dp *const prop_1,
 2               const weyl_dp *const prop_2,
 3               complex_dp *const out)
 4  {
 5    int dir_col;
 6
 7    /* ... */
 8
 9    out->re = 0.0;
10    out->im = 0.0;
11
12    for(dir_col=0; dir_col<6; dir_col++)
13    {
14      const weyl_dp *weyl1, *weyl2;
15
16      weyl1 = prop_1+dir_col;
17      weyl2 = prop_2+dir_col;
18
19      out->re += +_vector_prod_re(weyl2->c1, weyl1->c1)
20                 +_vector_prod_re(weyl2->c2, weyl1->c2);
21      out->im += +_vector_prod_im(weyl2->c1, weyl1->c1)
22                 +_vector_prod_im(weyl2->c2, weyl1->c2);
23    }
24  }
```

The definition of k_1 has already been cast into the form that is implemented in the function `k_1_rel` in equation (2.136). The equation is shown here again, with explicit color indices and without gauge average:

$$k_1^{ij} = \frac{2}{3}\sum_{\alpha,\beta=1}^{3}\Big[\bar{S}^j_{\mathrm{T}}(x)^\dagger_{1\alpha,1\beta}\big(2\bar{S}^i_{\mathrm{T}}(x)_{2\alpha,2\beta}+\bar{S}^i_{\mathrm{T}}(x)_{1\alpha,1\beta}\big)-\bar{S}^j_{\mathrm{T}}(x)^\dagger_{1\alpha,2\beta}\bar{S}^i_{\mathrm{T}}(x)_{1\alpha,2\beta}$$
$$\bar{S}^j_{\mathrm{T}}(x)^\dagger_{2\alpha,2\beta}\big(2\bar{S}^i_{\mathrm{T}}(x)_{1\alpha,1\beta}+\bar{S}^i_{\mathrm{T}}(x)_{2\alpha,2\beta}\big)-\bar{S}^j_{\mathrm{T}}(x)^\dagger_{2\alpha,1\beta}\bar{S}^i_{\mathrm{T}}(x)_{2\alpha,1\beta}\Big] \quad (5.19)$$

The constant factor 2/3 is left out again in `k_1_rel`. Its source code is shown in listing 5.6 and explained briefly in the next paragraphs.

line 13 This loop runs over the second color index β, and thus corresponds to the sum $\sum_{\beta=1}^{3}$.

line 15–18 The terms in round brackets in equation (5.19) are computed and assigned to the variables `v1` and `v2`.

line 20–31 The product of the two propagators is computed. The terms are ordered in the same way as in equation (5.19).

Listing 5.6: Source code of `k_1_rel` (file `src/modules/cf_rel/k_1_rel.c`)

```c
 1  void k_1_rel(const weyl_dp *const prop_1,
 2               const weyl_dp *const prop_2,
 3               complex_dp *const out)
 4  {
 5    int col;
 6    su3_vector_dp v1, v2;
 7
 8    /* ... */
 9
10    out->re = 0.0;
11    out->im = 0.0;
12
13    for(col=0; col<3; col++)
14    {
15      _vector_mul(v1, 2, prop_1[1*3+col].c2);
16      _vector_add_assign(v1, prop_1[0*3+col].c1);
17      _vector_mul(v2, 2, prop_1[0*3+col].c1);
18      _vector_add_assign(v2, prop_1[1*3+col].c2);
19
20      out->re += +_vector_prod_re(prop_2[0*3+col].c1, v1)
21                 -(_vector_prod_re(prop_2[1*3+col].c1,
22                                   prop_1[1*3+col].c1))
23                 +_vector_prod_re(prop_2[1*3+col].c2, v2)
24                 -(_vector_prod_re(prop_2[0*3+col].c2,
25                                   prop_1[0*3+col].c2));
26      out->im += +_vector_prod_im(prop_2[0*3+col].c1, v1)
27                 -(_vector_prod_im(prop_2[1*3+col].c1,
28                                   prop_1[1*3+col].c1))
29                 +_vector_prod_im(prop_2[1*3+col].c2, v2)
30                 -(_vector_prod_im(prop_2[0*3+col].c2,
31                                   prop_1[0*3+col].c2));
32    }
33  }
```

5.4. The Generic Functions

In contrast to the specific functions from the previous section `f_a_rel`, `k_t_rel` etc., the generic function `cf_rel` is able to compute an arbitrary two-point boundary-to-bulk correlation function with a source at the lower or upper boundary. There is also a generic function for boundary-to-boundary correlations, which is called `cf_rel_1`. Both take as arguments two numbers that specify which gamma matrix product is to be used to contract the bulk fields and the boundary fields.

The generic functions are less efficient than the specific functions, because they can not exploit the structure of the special gamma matrices that are used in a correlation function. In the chiral representation, as we use it here, only a quarter of the components of a certain gamma matrix is non-zero. Therefore, the specific functions can omit many multiplication operations, which would yield zero anyway. Thus, they only need to perform a small fraction of the operations of the generic function (see section 5.5.4). The relation is even worse for the vector correlations k_{T} and k_{V}, because they involve a sum over the index k of the gamma matrices. For the specific functions, this sum could be worked out in terms of the propagators' components. Many terms canceled, and in the end, the computational effort for k_{T} and k_{V} was the same as for f_{A} and f_{P}. However, with the generic routine, the sum over k must be done explicitly. `cf_rel` must be called three times with the appropriate matrices and the results must be added. Hence, the necessary time is greater by an additional factor three.

Anyway, the computation of the correlation functions still needs much less time than the computation of the relativistic propagators, in particular the solver routine, which dominates the overall computing time. So, the total time does not increase significantly, when the generic routines are used, but their maintenance and the implementation of new correlations is easier and less error-prone. Therefore, we are going to use the generic functions in the future.

The next section will explain how the functions `cf_rel` and `cf_rel_1` are called. Later, it will be discussed how a correlation function with arbitrary Dirac matrices is evaluated and how this has been implemented in the generic functions.

5.4.1. Usage of the Generic Functions

Listing 5.7 shows, as an example, the computation of f_{A}, g_{P} and f_1 via the generic functions. Like the specific functions, they are declared in the header file `cf_rel.h`. For both generic functions, the first two arguments are integers that identify the matrix that is used to contract the bulk and the boundary fields. To this purpose, constants like `GAMMA_5` and `GAMMA_0_5` are defined. Here, `GAMMA_5` stands for the matrix γ_5, and `GAMMA_0_5` for the product $\gamma_0 \cdot \gamma_5$. These constants are defined in the header file `gamma.h`, which is included by `cf_rel.h`. The third and fourth argument are the propagators, either the indices of the spinor fields which hold $\bar{S}^i(x)$ and $\bar{S}^j(x)$, or the arrays in which the boundary-to-boundary propagators $\bar{S}^i_{\mathrm{T}}$ and $\bar{S}^j_{\mathrm{T}}$ are stored. The last argument is a pointer to the output variable to which the computed correlation function is written. Before the output pointers, the function `cf_rel` takes another argument. It is an integer, and if it is not zero, it indicates that a backward correlation shall be computed. So, for g_{P}, a `1` is passed to the function.

Listing 5.7: The computation of f_{A}, g_{P} and f_1 via the generic functions

```c
#include "cf_rel.h"

/* ... */

/* Relativistic propagators */
int prop_1, prop_2;
/* Relativistic boundary-to-boundary propagator */
weyl_dp bbprop_1, bbprop_2;
/* Output arrays */
complex_dp out_f_a[TSIZE-1], out_g_p[TSIZE-1],
           out_f_1[1];

/* ... */

/* f_A */
cf_rel(GAMMA_0_5, GAMMA_5, prop_1, prop_2, 0, out_f_a);
/* g_P */
cf_rel(GAMMA_5, GAMMA_5, prop_1, prop_2, 1, out_g_p);
/* f_1 */
cf_rel_1(GAMMA_5, GAMMA_5, bbprop_1, bbprop_2, out_f_1);

/* ... */
```

The correlation functions k_{T}, k_{V} and k_1 can not be obtained by a single call to `cf_rel` or `cf_rel_1`, because they involve a sum over contractions with different gamma matrices. Instead, the generic functions must be called three times and the results must be summed. This can be done as shown in listing 5.8 for k_{T} and k_1 (the computation of k_{V} is done in the same way as for k_{T}, only `GAMMA_0_`k is replaced by `GAMMA_`k). However, the code shown can still be improved. For example, the summation over the gamma matrix combinations may be done in a loop.

Listing 5.8: The computation of k_{T} and k_1 via the generic functions

```c
/* ... */

complex_dp out_k_t[TSIZE-1], out_k_1[1];      /* output
                                                * arrays */
complex_dp out_temp[TSIZE-1];       /* temporary results */
int x0m1;                            /* time x_0 minus 1 */

/* ... */

/* k_T */
cf_rel(GAMMA_0_1, GAMMA_1, prop_1, prop_2, 0, out_k_t);
cf_rel(GAMMA_0_2, GAMMA_2, prop_1, prop_2, 0, out_temp);
for(x0m1=0; x0m1<TSIZE-1; x0m1++)
{
  out_k_t[x0m1].re += out_temp[x0m1].re;
  out_k_t[x0m1].im += out_temp[x0m1].im;
}
cf_rel(GAMMA_0_3, GAMMA_3, prop_1, prop_2, 0, out_temp);
for(x0m1=0; x0m1<TSIZE-1; x0m1++)
{
  out_k_t[x0m1].re += out_temp[x0m1].re;
  out_k_t[x0m1].im += out_temp[x0m1].im;
}

/* k_1 */
cf_rel_1(GAMMA_1, GAMMA_1, bbprop_1, bbprop_2, out_k_1);
cf_rel_1(GAMMA_2, GAMMA_2, bbprop_1, bbprop_2, out_temp);
out_k_1->re += out_temp->re;
out_k_1->im += out_temp->im;
cf_rel_1(GAMMA_3, GAMMA_3, bbprop_1, bbprop_2, out_temp);
out_k_1->re += out_temp->re;
out_k_1->im += out_temp->im;

/* The correlations k_T and k_1 are now in out_k_t and
 * out_k_1. */
```

As with the specific functions, the results have to be multiplied by a constant factor to match the normalization from the APE. These factors are shown in table 5.2. They have their origin in the following steps:

- `init_rhs` omits the factor $\tilde{c}_{\mathrm{t}}/(2\sqrt{V_3})$, and thus the propagator is lacking it,

correlation function	additional normalization
f_A	$1/2 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$
f_P	$1/2 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$
k_T	$1/6 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$
k_V	$1/6 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$
f_1	$1/2 \cdot \tilde{c}_\mathrm{t}^4/(16V_3^2)$
k_1	$1/6 \cdot \tilde{c}_\mathrm{t}^4/(16V_3^2)$

correlation function	additional normalization
g_A	$-1/2 \times \tilde{c}_\mathrm{t}^2/(4V_3)$
g_P	$1/2 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$
l_T	$-1/6 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$
l_V	$1/6 \cdot \tilde{c}_\mathrm{t}^2/(4V_3)$

Table 5.2.: Additional factors that are needed to normalize the correlation functions from the generic functions in the same way as on the APE

too. Since each QCD correlation function includes two relativistic propagators, they must be multiplied by $\tilde{c}_\mathrm{t}^2/(4V_3)$.

- `boundary_boundary_propagator` leaves out another factor of $-\tilde{c}_\mathrm{t}/(2\sqrt{V_3})$. Again, because the boundary-to-boundary correlations contain two propagators, they must be multiplied by the square $\tilde{c}_\mathrm{t}^2/(4V_3)$.
- The generic functions themselves compute correlations as indicated in (5.20) and (5.42), with the factors $-1/V_3$ and $-1/V_3^2$ in front of the sums over the field contractions. Any additional normalization factor must be taken care of manually (e.g. 1/2 for f_A, f_P and f_1).

These points can be summarized in the following rules: The results from `cf_rel` must be multiplied by $-\tilde{c}_\mathrm{t}^2/4$ and the factor in front of the field contractions in the correlation's definition. The results from `cf_rel_1` must be multiplied by $-\tilde{c}_\mathrm{t}^4/16$ and the factor appearing in the correlation's definition.

For example, the definition (2.120) of f_1 includes the normalization $-1/(2V_3^2)$. So the output of `cf_rel_1` is to be multiplied by $-\tilde{c}_\mathrm{t}^4/16 \times (-1)/(2V_3^2) = \tilde{c}_\mathrm{t}^4/(32V_3^2)$.

5.4.2. Evaluation of a Generic Correlation Function

In this section, we will evaluate generic boundary-to-bulk and boundary-to-boundary correlations as they are computed by `cf_rel` and `cf_rel_1`.

The function `cf_rel` is meant to determine correlation functions of the form

$$c^{ij}(x_0) = -\frac{1}{V_3}\sum_{\vec{x}\vec{y}\vec{z}} \left\langle \bar{\psi}^j(x)\cdot\mathcal{A}\cdot\psi^i(x)\bar{\zeta}^i(\vec{y})\cdot\mathcal{B}\cdot\zeta^j(\vec{z})\right\rangle_\mathrm{F}. \tag{5.20}$$

Here, $\mathcal{A}$ and $\mathcal{B}$ are arbitrary 4×4-matrices in Dirac space. For example, we can choose $\mathcal{A} = \gamma_0\gamma_5$ and $\mathcal{B} = \gamma_5$ to obtain f_A (up to constant factors).

As in section 2.3.8, we employ the Wick theorem to write $c(x_0)$ in terms of propagators (the minus sign disappears, because the Graßmann-valued spinor fields are commuted:

$$c^{ij}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\frac{1}{\sqrt{V_3}}\sum_{\vec{z}}\left\langle\zeta^j(\vec{z})\bar{\psi}^j(x)\right\rangle_{\mathrm{F}}\cdot\mathcal{A}\cdot\frac{1}{\sqrt{V_3}}\sum_{\vec{y}}\left\langle\bar{\zeta}^i(\vec{y})\psi^i(x)\right\rangle_{\mathrm{F}}\cdot\mathcal{B}\right] \tag{5.21}$$

$$= \sum_{\vec{x}} \mathrm{Tr}\left[\gamma_5\bar{S}^j(x)^\dagger\gamma_5\cdot\mathcal{A}\cdot\bar{S}^i(x)\cdot\mathcal{B}\right] \tag{5.22}$$

$$= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}^j(x)^\dagger\cdot\gamma_5\mathcal{A}\cdot\bar{S}^i(x)\cdot\mathcal{B}\gamma_5\right]. \tag{5.23}$$

Now, we will go a more formal way to take into account the symmetry of the propagators than in the last sections. Instead of $\bar{S}(x)\cdot P_- = 0$, we can express the symmetry in the equivalent way via P_+:

$$\bar{S}(x)\cdot P_+ = \bar{S}(x). \tag{5.24}$$

So, we can always insert the matrix P_+ after a propagator. P_+ is a projector to a two-dimensional sub-space of the four-dimensional space of all Dirac spinors. Therefore, we can write it as the product of a 4×2- and a 2×4-matrix. We will use the following splitting:

$$P_+ = \frac{1}{2}\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} = P_+^{(1)}\cdot P_+^{(2)} \tag{5.25}$$

with

$$P_+^{(1)} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \qquad P_+^{(2)} = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}. \tag{5.26}$$

With $P_+^{(1)}$, we can define a "reduced" version $\bar{S}_{\mathrm{r}}(x)$ of the propagator $\bar{S}(x)$ which has only 4×2 Dirac components:

$$\bar{S}_{\mathrm{r}}(x) = \bar{S}(x)\cdot P_+^{(1)}. \tag{5.27}$$

If we define $P_+^{(1)}$ as above, this reduced propagator is just the same as the original propagator constrained to the values $B = 1, 2$ of the second Dirac index:

$$\bar{S}_{\mathrm{r}}(x)_{A\alpha,B\beta} = \bar{S}(x)_{A\alpha,B\beta} \qquad \text{for } B = 1, 2, \tag{5.28}$$

so it is the propagator as we store it in the SFCF program. Therefore, we should express (5.23) in terms of $\bar{S}_{\mathrm{r}}(x)$.

The usual propagator $\bar{S}(x)$ can be reobtained from $\bar{S}_\mathrm{r}(x)$ via

$$\bar{S}(x) = \bar{S}(x) \cdot P_+ = \bar{S}(x) \cdot P_+^{(1)} \cdot P_+^{(2)} \tag{5.29}$$

$$= \bar{S}_\mathrm{r}(x) \cdot P_+^{(2)}. \tag{5.30}$$

This shows that no information is lost, when we use $\bar{S}_\mathrm{r}(x)$ instead of $\bar{S}(x)$.

To replace $\bar{S}(x)$ by $\bar{S}_\mathrm{r}(x)$, we insert equation (5.30) into (5.23):

$$c^{ij}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}_\mathrm{r}^j(x)^\dagger \cdot \gamma_5 \mathcal{A} \cdot \bar{S}_\mathrm{r}^i(x) \cdot P_+^{(2)} \mathcal{B}\gamma_5 \left(P_+^{(2)}\right)^\dagger\right] \tag{5.31}$$

$$= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}_\mathrm{r}^j(x)^\dagger \cdot \mathcal{C} \cdot \bar{S}_\mathrm{r}^i(x) \cdot \mathcal{D}\right] \tag{5.32}$$

with a 4×4-matrix $\mathcal{C}$ and a 2×2-matrix $\mathcal{D}$:

$$\mathcal{C} = \gamma_5 \cdot \mathcal{A} \qquad\qquad \mathcal{D} = P_+^{(2)} \cdot \mathcal{B} \cdot \gamma_5 \cdot \left(P_+^{(2)}\right)^\dagger. \tag{5.33}$$

Instead of the 4×4-matrix $\mathcal{B}\gamma_5$, we only need to compute the product with the 2×2-matrix $\mathcal{D}$. So, the symmetry of the propagator has significantly reduced the number of operations (additions and multiplications) that we need to compute c^{ij}.

Backward correlation functions $d^{ij}(x_0)$ can be treated analogously. We assume that they have the form

$$d^{ij}(x_0) = -\frac{1}{V_3} \sum_{\vec{x}\vec{y}\vec{z}} \left\langle \bar{\psi}^j(x) \cdot \mathcal{A} \cdot \psi^i(x) \bar{\zeta}'^i(\vec{y}) \cdot \mathcal{B} \cdot \zeta'^j(\vec{z}) \right\rangle_\mathrm{F}. \tag{5.34}$$

Via the Wick theorem, this is transformed to

$$d^{ij}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\bar{R}^j(x)^\dagger \cdot \gamma_5 \mathcal{A} \cdot \bar{R}^i(x) \cdot \mathcal{B}\gamma_5\right]. \tag{5.35}$$

The difference between the forward and backward correlation functions is that the backward propagator fulfills $\bar{R}(x) \cdot P_- = \bar{R}(x)$. We can split the projector P_- like

$$P_- = \frac{1}{2}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} = P_-^{(1)} \cdot P_-^{(2)} \tag{5.36}$$

with

$$P_-^{(1)} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad\qquad P_-^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \tag{5.37}$$

and define the reduced backward propagator $\bar{R}_{\rm r}(x)$ as

$$\bar{R}_{\rm r}(x) = \bar{R}(x) \cdot P_-^{(1)} \tag{5.38}$$

$$\Longrightarrow \quad \bar{R}(x) = \bar{R}_{\rm r}(x) \cdot P_-^{(2)}. \tag{5.39}$$

Then, we arrive at the analog of equation (5.32):

$$d^{ij}(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\bar{R}_{\rm r}^j(x)^\dagger \cdot \mathcal{C} \cdot \bar{R}_{\rm r}^i(x) \cdot \mathcal{D}\right] \tag{5.40}$$

with the matrices:

$$\mathcal{C} = \gamma_5 \cdot \mathcal{A} \qquad\qquad \mathcal{D} = P_-^{(2)} \cdot \mathcal{B} \cdot \gamma_5 \cdot \left(P_-^{(2)}\right)^\dagger . \tag{5.41}$$

Our treatment of a generic boundary-to-boundary correlation function is similar. The function `cf_rel_1` is to evaluate correlations of the form

$$c_1^{ij} = -\frac{1}{V_3^2} \sum_{\vec{u}\vec{v}\vec{y}\vec{z}} \left\langle \bar{\zeta}'^j(\vec{u}) \cdot \mathcal{A} \cdot \zeta'^i(\vec{v}) \bar{\zeta}^i(\vec{y}) \cdot \mathcal{B} \cdot \zeta^j(\vec{z}) \right\rangle_{\rm F} . \tag{5.42}$$

Through Wick's theorem, this becomes

$$= \mathrm{Tr}\left[\frac{1}{V_3}\sum_{\vec{u}\vec{z}} \left\langle \zeta^j(\vec{z})\bar{\zeta}'^j(\vec{u})\right\rangle_{\rm F} \cdot \mathcal{A} \cdot \frac{1}{V_3}\sum_{\vec{v}\vec{y}} \left\langle \zeta'^i(\vec{v})\bar{\zeta}^i(\vec{y})\right\rangle_{\rm F} \cdot \mathcal{B}\right] \tag{5.43}$$

$$= \mathrm{Tr}\left[\bar{S}_{\rm T}^{j\dagger} \cdot \gamma_5 \mathcal{A} \cdot \bar{S}_{\rm T}^i \cdot \mathcal{B}\gamma_5\right]. \tag{5.44}$$

Now, the "reduced" boundary-to-boundary propagator can be defined as

$$\bar{S}_{\rm Tr} = \left(P_+^{(1)}\right)^\dagger \cdot \bar{S}_{\rm T} \cdot P_+^{(1)} \tag{5.45}$$

$$\Longrightarrow \quad \bar{S}_{\rm T} = \left(P_+^{(2)}\right)^\dagger \cdot \bar{S}_{\rm Tr} \cdot P_+^{(2)}. \tag{5.46}$$

It would be possible to use $P_+^{(2)}$ in the definition instead of $(P_+^{(1)})^\dagger$, too, but $\bar{S}_{\rm Tr}$ as defined above matches the boundary-to-boundary propagator as we store it on the computer.

If we insert equation (5.46) into (5.44), the result is

$$c_1^{ij} = \mathrm{Tr}\left[\bar{S}_{\rm Tr}^{j\dagger} \cdot \mathcal{C} \cdot \bar{S}_{\rm Tr}^i \cdot \mathcal{D}\right] \tag{5.47}$$

with the two 2×2-matrices

$$\mathcal{C} = P_+^{(2)} \gamma_5 \mathcal{A} \left(P_+^{(2)}\right)^\dagger \qquad\qquad \mathcal{D} = P_+^{(2)} \mathcal{B}\gamma_5 \left(P_+^{(2)}\right)^\dagger . \tag{5.48}$$

The computational effort that we save, because we were able to replace the 4×4-matrices by 2×2-matrices, is not significant, since (5.47) must only be evaluated once per correlation function and not for every point of the lattice. But for consistency, we will use the reduced matrices $\mathcal{C}$ and $\mathcal{D}$ as in (5.32).

In the next section, the implementation of the equations (5.32) and (5.47) will be analyzed.

5.4.3. The Functions `cf_rel` and `cf_rel_1`

The functions `cf_rel` and `cf_rel_1` are meant to calculate arbitrary boundary-to-bulk and boundary-to-boundary correlation functions after the equations (5.32) and (5.47). Both are similar in their structure, so we will first discuss the design of `cf_rel` and then briefly look at `cf_rel_1`.

The function `cf_rel` implements the equation (5.32). Its source code is printed in listing 5.9 and explained below:

lines 12–14 `gamma_bulk` is an integer number which specifies the gamma matrix combination $\mathcal{A}$ for the bulk fields. The actual matrix can be accessed via `gamma[gamma_bulk]`. It is a complex 4×4-matrix represented by a variable of the type `mat4`. In these lines, the product $\mathcal{C} = \gamma_5 \cdot \mathcal{A}$ is calculated and written to the variable `mat_c` through the preprocessor macro `_mat4_times_mat4`.

Likewise, `gamma_boundary` identifies the matrix $\mathcal{B}$ to be used for the boundary fields. The product $\mathcal{B} \cdot \gamma_5$ is computed and written to `mat_tmp`. The multiplication by $P_+^{(2)}$ or $P_-^{(2)}$ is done in the next step.

lines 15–30 At this point forward and backward correlations must be distinguished. For backward correlations, the 2×2-matrix $\mathcal{D} = P_-^{(2)} \cdot \mathcal{B}\gamma_5 \cdot (P_-^{(2)})^\dagger$ is calculated; for forward correlations, it is $\mathcal{D} = P_+^{(2)} \cdot \mathcal{B}\gamma_5 \cdot (P_+^{(2)})^\dagger$. The multiplication is done explicitly at component level, and the resulting matrix is stored in `mat_d`.

The following code is independent of whether a forward or backward correlation is computed. For simplicity, we will assume that it is a forward correlation $c^{ij}(x_0)$.

line 32 This is the loop over the time coordinate. In each iteration, the value of $c^{ij}(x_0 = \texttt{x0})$ is computed.

lines 34–37 These two loops run over all points $(x_0, \vec{x})$ with $x_0 = \texttt{x0}$ (see section 5.3.2 and the description of listing 5.3). Thus, they correspond to the sum $\sum_{\vec{x}}$.

line 41 This loop is to sum over the second color indices β of the propagators. The sum over the second Dirac indices can not be done in this way, since the boundary fields are contracted by the possibly non-trivial matrix $\mathcal{B}$.

lines 46–49 Pointers to the propagators at point $(x_0, \vec{x})$ are defined. Here, `phi` and `chi` stand for the flavors i and j. The indices `_1` and `_2` in the variable names are the second Dirac indices. So, `chi_2` for example points to the location where $\bar{S}^j(x)_{**,2,(\texttt{col}+1)}$ is stored.

lines 51–68 The propagators are contracted with the bulk matrix $\mathcal{C}$. The result is the 2×2-matrix $\mathcal{E}_{AB} = \sum_{CD\alpha\beta} \bar{S}^j(x)_{C\alpha,A\beta} \cdot \mathcal{C}_{CD} \cdot \bar{S}^i(x)_{D\alpha,B\beta}$.

In line 51, the product $\sum_D \mathcal{C}_{CD}\bar{S}^i(x)_{D\alpha,B\beta}$ with $B = 1$ and $\beta = \texttt{col} + 1$ is computed via the preprocessor macro `_mat4_times_spinor` and written to the spinor variable `s_temp`. Then, the macro `_spinor_dag_times_spinor_` is used to multiply the result by $\bar{S}^j(x)^*_{C\alpha,A\beta}$ with $A = 1$ and sum over C and α. This is repeated for the other three possible combinations of A and B. At last, the sum over β is done by looping over the variable `col` and adding the results each time.

lines 71–81 The matrix $\mathcal{E}$ that stems from the "bulk" contraction of the propagators is now contracted with the boundary matrix $\mathcal{D}$, i.e. $\sum_{AB} \mathcal{E}_{AB}\mathcal{D}_{BA}$ is calculated and added to the local sum in `out_local[x0-1]`.

lines 85–86 After the end of the double loop over $\vec{x}$, the local sums of all processes in `out_local[x0-1]` are added and the global sum is returned in the output argument `out[x0-1]`.

Listing 5.9: The source code of the function `cf_rel` (file `src/modules/cf_rel/cf_rel.c`)

```c
 1 void cf_rel(const int gamma_bulk,
 2             const int gamma_boundary, const int prop_1,
 3             const int prop_2, const int backward,
 4             complex_dp *const out)
 5 {
 6   mat4 mat_c, mat_tmp;
 7   mat2 mat_d;
 8   int x0, ind, i_eo, col;
 9   complex_dp out_local[TSIZE-1]={{0.0}};
10
11   _mat4_times_mat4(mat_c, gamma[GAMMA_5],
12                    gamma[gamma_bulk]);
13   _mat4_times_mat4(mat_tmp, gamma[gamma_boundary],
14                    gamma[GAMMA_5]);
15   if(backward)
16   {
17     mat_d.c11.re = +mat_tmp.c11.re+mat_tmp.c13.re
18                    +mat_tmp.c31.re+mat_tmp.c33.re;
19     mat_d.c11.im = +mat_tmp.c11.im+mat_tmp.c13.im
20                    +mat_tmp.c31.im+mat_tmp.c33.im;
21     /* ... */
22   }
23   else
24   {
25     mat_d.c11.re = +mat_tmp.c11.re-mat_tmp.c13.re
26                    -mat_tmp.c31.re+mat_tmp.c33.re;
27     mat_d.c11.im = +mat_tmp.c11.im-mat_tmp.c13.im
```

```
                        -mat_tmp.c31.im+mat_tmp.c33.im;
    /* ... */
  }

  for(x0=1; x0<TSIZE; x0++)
  {
    for(i_eo=0; i_eo<2; i_eo++)
    {
      for(ind=(x0-1)*TSLICE/2+i_eo*BULK/2;
          ind<x0*TSLICE/2+i_eo*BULK/2; ind++)
      {
        mat2 mat_e={{0.0}};

        for(col=0; col<3; col++)
        {
          spinor_dp *psi_1, *psi_2, *chi_1, *chi_2;
          spinor_dp s_temp;

          psi_1 = PSD1e[prop_1+0*3+col]+ind;
          psi_2 = PSD1e[prop_1+1*3+col]+ind;
          chi_1 = PSD1e[prop_2+0*3+col]+ind;
          chi_2 = PSD1e[prop_2+1*3+col]+ind;

          _mat4_times_spinor(s_temp, mat_c, *psi_1);
          mat_e.c11.re += _re_spinor_dag_times_spinor
                          (*chi_1, s_temp);
          mat_e.c11.im += _im_spinor_dag_times_spinor
                          (*chi_1, s_temp);
          mat_e.c21.re += _re_spinor_dag_times_spinor
                          (*chi_2, s_temp);
          mat_e.c21.im += _im_spinor_dag_times_spinor
                          (*chi_2, s_temp);
          _mat4_times_spinor(s_temp, mat_c, *psi_2);
          mat_e.c12.re += _re_spinor_dag_times_spinor
                          (*chi_1, s_temp);
          mat_e.c12.im += _im_spinor_dag_times_spinor
                          (*chi_1, s_temp);
          mat_e.c22.re += _re_spinor_dag_times_spinor
                          (*chi_2, s_temp);
          mat_e.c22.im += _im_spinor_dag_times_spinor
                          (*chi_2, s_temp);
        }

        out_local[x0-1].re += +mat_e.c11.re*mat_d.c11.re
```

```
                              -mat_e.c11.im*mat_d.c11.im
                              +mat_e.c12.re*mat_d.c21.re
                              -mat_e.c12.im*mat_d.c21.im
                              +mat_e.c21.re*mat_d.c12.re
                              -mat_e.c21.im*mat_d.c12.im
                              +mat_e.c22.re*mat_d.c22.re
                              -mat_e.c22.im*mat_d.c22.im;
        out_local[x0-1].im += +mat_e.c11.re*mat_d.c11.im
                              +mat_e.c11.im*mat_d.c11.re
                              /* ... */
      }
    }

    mpc_gsum_d(&out_local[x0-1].re, &out[x0-1].re, 1);
    mpc_gsum_d(&out_local[x0-1].im, &out[x0-1].im, 1);
  }
}
```

The function `cf_rel_1` for the boundary-to-boundary correlations is basically similar to `cf_rel`. As for the specific functions, the differences are that no summation over $\vec{x}$ is required and that due to the additional symmetry of $\bar{S}_{\mathrm{T}}$, also the matrix C can be reduced to a 2×2-matrix.

`cf_rel_1` calculates a generic boundary-to-boundary correlation function as specified in equation (5.47). Listing 5.10 shows its source code, which is explained in brief in the following paragraphs.

lines 12–14 Computation of $\gamma_5 \cdot \mathcal{A}$ and $\mathcal{B} \cdot \gamma_5$

lines 16–24 "Reduction" of the matrices: $\mathcal{C} = P_+^{(2)} \cdot \gamma_5\mathcal{A} \cdot (P_+^{(2)})^\dagger$ and $\mathcal{D} = P_+^{(2)} \cdot \mathcal{B}\gamma_5 \cdot (P_+^{(2)})^\dagger$ are computed.

line 26 Loop over second color index (β)

lines 31–34 Boundary-to-boundary propagators with current second color index β

lines 36–45 Contraction of propagators and matrix $\mathcal{C}$

lines 48–63 Contraction with matrix $\mathcal{D}$

Listing 5.10: Source code of the function `cf_rel_1` (file `src/modules/cf_rel/cf_rel.c`)

```c
void cf_rel_1(const int gamma_top,
              const int gamma_bottom,
              const weyl_dp *bbprop_1,
              const weyl_dp *bbprop_2,
              complex_dp *const out)
{
  mat4 mat_tmp1, mat_tmp2;
  mat2 mat_c, mat_d, mat_e={{0.0}};
  int col;

  _mat4_times_mat4(mat_tmp1, gamma[GAMMA_5],
                   gamma[gamma_top]);
  _mat4_times_mat4(mat_tmp2, gamma[gamma_bottom],
                   gamma[GAMMA_5]);
  mat_c.c11.re = +mat_tmp1.c11.re-mat_tmp1.c13.re
                 -mat_tmp1.c31.re+mat_tmp1.c33.re;
  mat_c.c11.im = +mat_tmp1.c11.im-mat_tmp1.c13.im
                 -mat_tmp1.c31.im+mat_tmp1.c33.im;
  /* ... */
  mat_d.c11.re = +mat_tmp2.c11.re-mat_tmp2.c13.re
                 -mat_tmp2.c31.re+mat_tmp2.c33.re;
  mat_d.c11.im = +mat_tmp2.c11.im-mat_tmp2.c13.im
                 -mat_tmp2.c31.im+mat_tmp2.c33.im;
  /* ... */

  for(col=0; col<3; col++)
  {
    const weyl_dp *psi_1, *psi_2, *chi_1, *chi_2;
    weyl_dp w;

    psi_1 = bbprop_1+0*3+col;
    psi_2 = bbprop_1+1*3+col;
    chi_1 = bbprop_2+0*3+col;
    chi_2 = bbprop_2+1*3+col;

    _mat2_times_weyl(w, mat_c, *psi_1);
    mat_e.c11.re += _re_weyl_dag_times_weyl(*chi_1, w);
    mat_e.c11.im += _im_weyl_dag_times_weyl(*chi_1, w);
    mat_e.c21.re += _re_weyl_dag_times_weyl(*chi_2, w);
    mat_e.c21.im += _im_weyl_dag_times_weyl(*chi_2, w);
    _mat2_times_weyl(w, mat_c, *psi_2);
    mat_e.c12.re += _re_weyl_dag_times_weyl(*chi_1, w);
    mat_e.c12.im += _im_weyl_dag_times_weyl(*chi_1, w);
    mat_e.c22.re += _re_weyl_dag_times_weyl(*chi_2, w);
    mat_e.c22.im += _im_weyl_dag_times_weyl(*chi_2, w);
  }

  out->re = +mat_e.c11.re*mat_d.c11.re
            -mat_e.c11.im*mat_d.c11.im
            +mat_e.c12.re*mat_d.c21.re
            -mat_e.c12.im*mat_d.c21.im
            +mat_e.c21.re*mat_d.c12.re
            -mat_e.c21.im*mat_d.c12.im
            +mat_e.c22.re*mat_d.c22.re
            -mat_e.c22.im*mat_d.c22.im;
  out->im = +mat_e.c11.re*mat_d.c11.im
            +mat_e.c11.im*mat_d.c11.re
            +mat_e.c12.re*mat_d.c21.im
            +mat_e.c12.im*mat_d.c21.re
            +mat_e.c21.re*mat_d.c12.im
            +mat_e.c21.im*mat_d.c12.re
            +mat_e.c22.re*mat_d.c22.im
            +mat_e.c22.im*mat_d.c22.re;
}
```

5.5. Comparison of the Running Times of Specific and Generic Functions

When I had written the generic functions, our main concern was that they could need too much time. So, I have performed a test series comparing specific and generic routines for different lattice sizes. The times measured in this series have great uncertainties. They fluctuate in repeated measurements on the same machine, and show quite different behavior on different machines (see section 5.5.6). They can only provide a rough impression of the execution times of the specific and generic functions, the solver and other routines.

5.5.1. The Computers

The measurements were done on three different computers of the Institut für Theoretische Physik at the University of Münster. They are named Tesla, Higgs and Kelvin. In principle, all of them could be used by all members of the institute to run complex, CPU-time-consuming calculations at the same time, which might have interfered with the measurements. In particular, this applies to Tesla and Higgs. However, it was checked that there was always a sufficient number of CPU cores available for the measurement program, so it could run at full speed.

All three machines were running the GNU/Linux distribution Debian, release Squeeze, had x86-64 architectures and supported SSE3 instructions. Below, they are described briefly.

Tesla: The computer "Tesla" is a central server in the institute, which can be accessed by all members to carry out complex calculations, but when I used it to measure running times, this was not the case and it was nearly idle. It has 24 gigabytes of memory and two Intel® Xeon® processors (model L5506, 2.13 GHz, each with four cores).

Higgs: This is a desktop computer with an Intel® Core2 Quad processor (model Q6600, 2.4 GHz) and 2.1 GB memory.

Kelvin: "Kelvin" is the desktop computer at my office place. During the tests it was not used by anyone but me. It contains an AMD Athlon 64 X2 processor (model 4400+, 2.2 GHz, two cores) and 1.9 GB memory.

5.5.2. The Test Program

The main program that was used to measure the times is `src/tests/src/tst-cw-cf-rel.c`. It computed the correlation functions f_A, f_P, k_T, k_V, g_A, g_P, l_T and l_V, as well as the boundary-to-boundary correlations f_1 and k_1 between two different quark flavors. This was done for three different lattice sizes, $T = L_1 = L_2 = L_3 = 8, 16, 32$. For the $L = 8$ lattice, a gauge configuration was used that had been created in a simulation with two dynamical, light quark flavors. The mass of the light quark, represented by κ_1 was carried over from this simulation, as well as the parameters β and θ. The improvement coefficients c_sw, c_t and $\tilde{c}_\mathrm{t}$ were calculated after equations (2.67), (2.56) and (2.72) with $n_f = 2$. The hopping parameter κ_2 for the heavier quark has been chosen arbitrarily. It lies within the typical range for the quarks we consider for B-physics applications, but the precise value has no meaning. The values of all parameters are shown below:

$$\beta = 5.4689 \qquad c_\mathrm{sw} = 1.772253895800868 \tag{5.49}$$

$$c_\mathrm{t} = 0.9082469422217880 \qquad \tilde{c}_\mathrm{t} = 0.9803104825467645 \tag{5.50}$$

$$\kappa_1 = 0.1367 \qquad \kappa_2 = 0.126736 \tag{5.51}$$

$$\theta = 0.5 \tag{5.52}$$

The gauge fields on the bigger lattices $L = 16$ and $L = 32$ were created randomly by the program. Hence, the time taken by the solver will deviate from the times that it needs in praxis, when it is applied to a thermalized gauge field. The solver was set so that it stopped at a squared residuum of 10^{-13} (parameter `errsq`). The correlation functions obtained in this way are exact to circa seven decimal places.[4]

A run of the program proceeded as follows: After it had allocated memory for the fields, read or created the gauge field and finished other initialization tasks, the first thing was to prepare the right-hand sides for the computation of the relativistic propagators by means of `init_rhs`. This had to be done for two Dirac indices, three color indices and for forward and backward propagators, summed up, `init_rhs` had to be called twelve times (see section 5.1.2). Second, the propagators were computed via the `solve` function. The solver had to be called 24 times, since two quark flavors with different masses were taken into account. This was the most time consuming step. After the boundary-to-bulk propagators were determined, it computed the boundary-to-boundary propagators via `boundary_boundary_propagator`. This function was employed only twice, for the forward and backward boundary-to-boundary propagator. Finally, the correlation functions were calculated, as well via the specific routines as via the generic routines. The correlation functions and the measured times were written to the standard output.

Since all three computers that were used for the measurements had the same architecture and operating system, it was possible to compile the test program only once (for each lattice size) and use the same executable files on all computers.

5.5.3. Time Measurement

The time was obtained via the function `clock_gettime`[5] in non-MPI mode and via `mpc_time` in MPI mode, which is a wrapper for the function `MPI_Wtime` (documented in [Ope]).

`clock_gettime` can return the time according to different clocks. The clock is chosen by the first argument. In our case, this was `CLOCK_PROCESS_CPUTIME_ID`. This constant identifies a clock that measures the time which the process has spent actively executing instructions on the CPU. In contrast, `MPI_Wtime` returns the "wall-clock time", the real time that has past since an arbitrary, but constant date. This is the time as a wall clock would show it. These two kinds of time measurement should be well distinguished from each other, but here the difference probably does not matter, because the parts of the program we are considering did not have to wait for input or output and they did not have to wait for other processes running on the same CPU core, either (see previous section 5.5.1).

To examine how much the execution times vary, each measured function was called a hundred times[6] in series by the program and the average time and the standard deviation were calculated. Only the solver function was called only once, because even computing the propagators a single time could take circa 45 minutes (for $L = 32$

[4]This is the number of digits that is identical to the results that are obtained, when the solver is run farther until the squared residuum falls below 10^{-30}.

[5]This function is defined in the POSIX.1b standard [Pos] and we used the implementation that is distributed along with the GNU C library [McG], version 2.11.

[6]This can be adjusted through the macro constant `MEAS_CNT`.

on Tesla, see table 5.4).

From the other parts of the program, the average execution time and its standard deviation were returned. So, if $t_1, \ldots, t_{100}$ are the single measurements, the values that were written to standard output are

$$t = \frac{1}{100}\sum_{i=1}^{100} t_i \qquad \text{and} \qquad \sigma_t = \sqrt{\frac{1}{100-1}\sum_{i=1}^{100}(t_i - t)^2}. \tag{5.53}$$

5.5.4. Expectations

I have tried to estimate the execution times of specific and generic relative to each other by counting the number of operations to be performed, especially complex multiplications. Their number can also be determined quite simply from the mathematical expression that is evaluated. We can look at the specific routine `f_p_rel` as an example. It computes the expression given in equation (2.117), which involves the term $\mathrm{Tr}\,[\bar{S}_\mathrm{r}(x)^\dagger \bar{S}_\mathrm{r}(x)]$ (here expressed in terms of “reduced” propagators). At component level, this becomes $\sum_{A\alpha,B\beta}(\bar{S}_\mathrm{r}(x)_{A\alpha,B\beta})^* \cdot \bar{S}_\mathrm{r}(x)_{A\alpha,B\beta}$ and we can see that the number of complex multiplications is the number of all different combinations of values for A, B, α and β, i.e. $2 \times 4 \times 3 \times 3 = 72$. So, per lattice site 72 complex multiplications are to be done. This corresponds to 288 real multiplications and the same number of real additions (since each product is added to a previous result). The same holds for the routine `f_a_rel` evaluating equation (2.113), although the corresponding term includes a γ_0-matrix here: $\mathrm{Tr}\,[\bar{S}_\mathrm{r}(x)^\dagger \gamma_0 \bar{S}_\mathrm{r}(x)]$, but this can be taken into account by permuting the components of one propagator, as shown in listing 5.3.

If we consider the source code in listing 5.3, lines 20–28 directly, we arrive at the same numbers (plus an incrementation of `cnt[0]`), but we can see that other operations are done, e.g. the dereferencing of pointers in lines 17 to 18 and the hierarchical summation in lines 28 and 30 to 38. In the following, I will assume that they can be neglected.

For `k_t_rel` and `k_v_rel`, we find the same number of multiplications and additions, if we take into account only the lines 20 to 36 of listing 5.4. (Note that the loop over `col` is executed three times, whereas the loop over `dir_col` in `f_a_rel` is repeated six times.) However, the auxiliary variables `v1`,…, `v4` must be prepared previously (lines 11 to 18). Including them, I have counted 360 real multiplications and 360 real additions per lattice site.

The generic function `cf_rel` calculates the trace $\mathrm{Tr}\,[\bar{S}_\mathrm{r}^\dagger \mathcal{C} \bar{S}_\mathrm{r} \mathcal{D}]$ with arbitrary 4×4 and 2×2 matrices $\mathcal{C}$ and $\mathcal{D}$. It can not assume any special structure of $\mathcal{C}$ and $\mathcal{D}$ (e.g. some components' being zero), so it must carry out the full computation. This amounts to 1744 real multiplications and 1600 real additions per lattice site.

The numbers of operations in the functions are summarized in table 5.3.

The boundary-to-boundary functions like `f_1_rel` and `cf_rel_1` will take much less time than the functions computing boundary-to-bulk correlations, because they

function	**real multiplications**	**real additions**
`f_a_rel`, `f_p_rel`	288	288
`k_t_rel`, `k_v_rel`	360	360
`cf_rel`	1744	1600

Table 5.3.: Number of operations per lattice site that are performed by the specific and generic functions

do not sum over the lattice sites. So, their time consumption will be negligible anyway.

5.5.5. Results in Single-Node Mode on Tesla

Table 5.4 shows the execution times of the specific and generic functions on Tesla for the three lattice sizes 8^4, 16^4 and 32^4 in single-node mode, i.e. the program was not parallelized. As described in the previous section, the times have been averaged over 100 calls and the numbers after the $\pm$ sign are the standard deviations of the measurements (see equation (5.53)), not the measurement uncertainties, which would be smaller by a factor $1/\sqrt{100}$.

For each lattice size, we can observe that the specific functions take least time. For $L = 8$, they compute the pseudo-scalar and axial-vector-current correlations (f_A, f_P, g_A, g_P) in circa 1.7 ms. The vector-current correlations (k_V, k_T, l_V, l_T) need circa 2.1 ms, i.e. about 25% longer. This is exactly the factor we would expect from the number of operations in table 5.3.

The generic function `cf_rel` needs about 6.1 ms for the computation of the axial correlation functions, which is between 3.4 and 3.7 times as long as the specific functions. This is faster than the number of multiplications and additions in table 5.3 suggests. There, the ratio is 6.1 for multiplications and 5.6 for additions. So, `cf_rel` seems to be more efficient than the specific functions, that is it takes less time per operation. This may be due to compiler optimization, but this does not explain why other ratios were obtained on other computers (see section 5.5.6), since the same executables were used. Instead, I suppose that the code could be executed faster, because some variables of `cf_rel` were kept in the processor's cache. In contrast to e.g. `f_a_rel`, many commands in `cf_rel` contain variables that have been computed or used a few lines before. So, they could still reside in the cache and the program could access them faster, but this has not been studied further.

For the vector correlations, the generic function `cf_rel` had to be called three times, once for each value of the index k of the gamma matrices (see e.g. equation (2.122)). And indeed, the computations of k_T, k_V, l_T and l_V take between 2.9 and 3.1 times as long as the computations of the axial correlations.

The computations of the boundary-to-boundary correlations need only negligible times, because they do not include a summation over $\vec{x}$. This is done in

	Execution times on Tesla in ms					
	lattice size $T = L = 8$		lattice size $T = L = 16$		lattice size $T = L = 32$	
	specific	generic	specific	generic	specific	generic
f_A	1.688 ± 0.014	6.250 ± 0.233	32.568 ± 0.100	108.885 ± 0.239	595.450 ± 0.092	1843.022 ± 1.572
f_P	1.688 ± 0.036	6.221 ± 0.057	32.298 ± 0.111	108.836 ± 0.092	593.514 ± 0.912	1845.914 ± 0.103
f_1	0.000 ± 0.000	0.001 ± 0.002	0.000 ± 0.000	0.002 ± 0.009	0.000 ± 0.000	0.001 ± 0.002
k_V	2.150 ± 0.041	18.544 ± 0.122	39.738 ± 0.175	326.477 ± 0.169	697.864 ± 0.063	5537.732 ± 0.203
k_T	2.140 ± 0.021	18.474 ± 0.094	39.993 ± 0.149	326.488 ± 0.190	707.954 ± 1.056	5527.792 ± 2.197
k_1	0.000 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.003 ± 0.000
g_A	1.837 ± 0.021	6.299 ± 0.029	32.520 ± 0.102	108.722 ± 0.091	603.161 ± 1.315	1862.050 ± 0.130
g_P	1.816 ± 0.024	6.307 ± 0.032	32.260 ± 0.109	108.752 ± 0.072	601.837 ± 0.098	1862.093 ± 0.143
l_V	2.307 ± 0.024	19.120 ± 0.371	39.536 ± 0.109	326.244 ± 0.075	704.775 ± 0.089	5580.370 ± 5.667
l_T	2.319 ± 0.032	19.061 ± 0.406	39.862 ± 0.108	326.171 ± 0.165	716.199 ± 3.157	5575.717 ± 2.945

Table 5.4.: Execution times of specific and generic functions for different correlations and lattice sizes, measured on the computer Tesla (the values after the $\pm$ signs are standard deviations, not measurement uncertainties, see text)

`boundary_boundary_propagator` instead (its running time is listed in table 5.5).

Of course, the times increase for the larger lattices with $L = 16$ and $L = 32$, but the proportions remain rougly the same. There is a vague trend that the generic function `cf_rel` gets faster compared to the specific functions. For $L = 32$, it needs only 3.1 times longer for the axial correlation functions.

Other parts of the program whose execution times have been measured are the functions `init_rhs`, `solve` and `boundary_boundary_propagator`. In this test program, `init_rhs` had to be called six times to compute one relativistic propagator (2 Dirac indices $\times$ 3 color indices). Since forward and backward correlations were calculated, it was called 12 times in the end. The solver, too, had to be employed six times for a single propagator, but here forward and backward propagators for two quark flavors were needed. So, it was called 24 times all in all. The function `boundary_boundary_propagator` was called only twice, once for each quark flavor. Table 5.5 shows the total times for all calls to each of these functions, as well as the total times spent in the specific and generic routines for the computation of the correlations as listed in table 5.4.

	Total execution times on Tesla in s		
	$T = L = 8$	$T = L = 16$	$T = L = 32$
specific functions	16	289	5221
generic functions	100	1741	29635
`init_rhs`	2	39	508
solver	42633	180565	2763433
`boundary_boundary_propagator`	0	6	47

Table 5.5.: Times spent in different parts of the program for three lattice sizes measured on the computer Tesla

Obviously, the solver takes by far the most time. On the $L = 16$ and $L = 32$ lattices, where random gauge fields were used, it took about 100 times as long as the generic functions, and on the $L = 8$ lattice, with a realistic gauge field, it took even 400 times as long. This last number is probably closer to what will be seen in real computations, but to a certain degree, it may be an overestimation. The solver was a simple conjugate-gradient solver without preconditioning. Other solver methods will be faster, and in real computations, one may compute more correlation functions, so the time spent on the solver could be shorter and the time spent on the computation of the correlation functions could be longer. Nonetheless, the solver will dominate the time budget. The times of the functions `init_rhs` and `boundary_boundary_propagator` play only a minor role.

5.5.6. Results in Single-Node Mode on Higgs and Kelvin

The test program was also run on the desktop computers Higgs and Kelvin (when they were idle). The results for the correlation functions on the $L = 8$ lattice are displayed in table 5.6 and the times of the solver and other program parts are shown in table 5.7. To compare them with Higgs and Kelvin, the times from Tesla are listed, too.

The times obtained on Higgs are longer, but the ratios are very similar to the results from Tesla. However, Kelvin shows a different behavior. In particular, `cf_rel` runs only about twice as long as e.g. `f_a_rel`. So, in relation to the specific functions, `cf_rel` is much faster on Kelvin than on Tesla and Higgs, or the specific functions are much slower.

Considering the times spent in the solver (table 5.7), the results from Higgs and Tesla are very similar again. For the 8^4-lattice, the solver needs between 400 and 500 times the time of the generic functions on both machines. But on Kelvin this ratio is twice as big, there the solver needs about 800 times of what `cf_rel` needs. I do not know the origin of this difference, but since the executed code was exactly the same on all machines, it must be caused by differences in the computers' architectures.

5.5.7. Results in MPI mode

Some tests were also performed with a parallelized program on Tesla. To do this, the preprocessor macro `USE_MPI` must be defined and the macros `NPROC1`, `NPROC2` and `NPROC3` must be set to the number of splits in each space direction. So, the total number of nodes is `NPROC1` $\times$ `NPROC2` $\times$ `NPROC3`. Here, one test was done on two nodes and one on four nodes. The `NPROC` macros were set to the following values:

`NPROC1`	1	1
`NPROC2`	1	2
`NPROC3`	2	2
total number of nodes	2	4

In all other aspects, the test programs were identical to the single-node version. In tables 5.8 and 5.9, the times for the 8^4-lattice are contrasted with the times of the single-node version.

The ratios between the times of specific and generic functions and the solver for the two-node version are approximately the same as for the single-node version. On four nodes, the generic function becomes faster relative to the other functions. The absolute times of the specific and generic functions are roughly proportional to the inverse number of nodes, but the solver time decreases more slowly. Thus, on four nodes, the solver takes nearly 900 times as long as the generic functions, instead of about 400 times in single-node mode.

	Execution times for $T = L = 8$ in ms					
	on Tesla		on Higgs		on Kelvin	
	specific	generic	specific	generic	specific	generic
f_A	1.688 ± 0.014	6.250 ± 0.233	2.598 ± 0.016	8.722 ± 0.011	5.006 ± 0.037	8.571 ± 0.018
f_P	1.688 ± 0.036	6.221 ± 0.057	2.553 ± 0.010	8.719 ± 0.009	4.001 ± 0.037	8.569 ± 0.018
f_1	0.000 ± 0.000	0.001 ± 0.002	0.001 ± 0.000	0.002 ± 0.003	0.001 ± 0.000	0.002 ± 0.003
k_V	2.150 ± 0.041	18.544 ± 0.122	3.196 ± 0.013	26.157 ± 0.020	5.073 ± 0.024	25.704 ± 0.045
k_T	2.140 ± 0.021	18.474 ± 0.094	3.296 ± 0.010	26.154 ± 0.016	4.628 ± 0.025	25.700 ± 0.035
k_1	0.000 ± 0.000	0.003 ± 0.000	0.001 ± 0.000	0.004 ± 0.002	0.001 ± 0.000	0.004 ± 0.001
g_A	1.837 ± 0.021	6.299 ± 0.029	2.648 ± 0.014	8.746 ± 0.009	4.982 ± 0.026	8.580 ± 0.024
g_P	1.816 ± 0.024	6.307 ± 0.032	2.601 ± 0.010	8.749 ± 0.013	3.945 ± 0.017	8.578 ± 0.022
l_V	2.307 ± 0.024	19.120 ± 0.371	3.256 ± 0.013	26.234 ± 0.015	5.106 ± 0.030	25.719 ± 0.045
l_T	2.319 ± 0.032	19.061 ± 0.406	3.442 ± 0.013	26.234 ± 0.024	4.656 ± 0.027	25.718 ± 0.045

Table 5.6.: Execution times of specific and generic functions for an 8^4-lattice on the three computers "Tesla", "Higgs" and "Kelvin" (the values after the $\pm$ signs are standard deviations, not measurement uncertainties, see text)

	Total execution times for $T = L = 8$ in s		
	on Tesla	on Higgs	on Kelvin
specific functions	16	24	37
generic functions	100	140	137
`init_rhs`	2	4	7
solver	42633	64732	112274
`boundary_boundary_propagator`	< 1	1	1

Table 5.7.: Times spent in different parts of the program on the three computers Tesla, Higgs and Kelvin

5.5.8. Conclusion

The results of the previous sections indicate that the time for the computation of correlation functions is negligible compared to the solver time, regardless whether it is done via the specific or generic functions. Therefore, the advantages of the generic approach (better maintainability, easier implementation of new functions etc.) will probably outweigh its additional time consumption. The situation may change if another solver method is implemented or if more correlation functions are computed. Yet, then the generic function can still be made more efficient as suggested in the next section.

5.5.9. Suggestions for Improvements

The generic approach exhibits many advantages, and if its time excess becomes a problem, one can solve it in different ways. For example the gamma matrices could be stored as sparse matrices, i.e. only their non-zero entries would be stored, and the function `cf_rel` could drop the unnecessary multiplications with components that are zero. This is just what is done for each correlation in the specific functions.

The time for the vector correlations could be reduced by performing the sum over the index of the gamma matrices within `cf_rel` and within the loop over the point $\vec{x}$. Thus, the propagators' components would still reside in the cache, when the contractions with the different gamma matrices are computed, and they could be retrieved quickly.[7] In a similar situation with the spin correlation functions of HQET[8], this yielded a time reduction of circa 25%.

An even greater reduction could be achieved if not the gamma matrices or $\mathcal{C}$ and $\mathcal{D}$ were passed to `cf_rel`, but their tensor product, i.e. $\mathcal{W}_{ABCD} = \mathcal{C}_{AB} \cdot \mathcal{D}_{CD}$, because for the vector correlations, $\mathcal{W}$ could be summed over, before it is passed to `cf_rel`.

[7]Thanks to Hubert Simma, who pointed this out to me.

[8]The question was whether the summation coming from the scalar product $\vec{\sigma} \cdot \vec{B}$ would be better done inside or outside the function `cf_hqet` (see section 6.2.1).

	Execution times for $T = L = 8$ on Tesla in ms					
	single node		$1 \times 1 \times 2$ nodes		$1 \times 2 \times 2$ nodes	
f_A	1.688 ± 0.014	6.250 ± 0.233	0.922 ± 0.039	3.170 ± 0.038	0.582 ± 0.045	1.663 ± 0.040
f_P	1.688 ± 0.036	6.221 ± 0.057	0.918 ± 0.041	3.171 ± 0.041	0.574 ± 0.025	1.661 ± 0.040
f_1	0.000 ± 0.000	0.001 ± 0.002	0.004 ± 0.001	0.001 ± 0.002	0.008 ± 0.001	0.001 ± 0.002
k_V	2.150 ± 0.041	18.544 ± 0.122	1.148 ± 0.045	9.526 ± 0.064	0.693 ± 0.027	4.982 ± 0.043
k_T	2.140 ± 0.021	18.474 ± 0.094	1.130 ± 0.051	9.516 ± 0.056	0.686 ± 0.032	4.979 ± 0.037
k_1	0.000 ± 0.000	0.003 ± 0.000	0.005 ± 0.001	0.003 ± 0.001	0.008 ± 0.001	0.003 ± 0.001
g_A	1.837 ± 0.021	6.299 ± 0.029	0.934 ± 0.047	3.176 ± 0.037	0.592 ± 0.037	1.672 ± 0.040
g_P	1.816 ± 0.024	6.307 ± 0.032	0.924 ± 0.038	3.177 ± 0.041	0.584 ± 0.023	1.672 ± 0.035
l_V	2.307 ± 0.024	19.120 ± 0.371	1.150 ± 0.041	9.545 ± 0.060	0.701 ± 0.027	5.013 ± 0.040
l_T	2.319 ± 0.032	19.061 ± 0.406	1.124 ± 0.042	9.540 ± 0.063	0.695 ± 0.028	5.007 ± 0.034

Table 5.8.: Execution times of specific and generic functions for an 8^4-lattice on Tesla, in the single-node version and parallelized on two and four nodes (the values after the $\pm$ signs are standard deviations, not measurement uncertainties, see text)

	Total execution times for $T = L = 8$ on Tesla in s		
	single node	$1 \times 1 \times 2$	$1 \times 2 \times 2$
specific functions	16	8	5
generic functions	100	51	27
`init_rhs`	2	1	1
solver	42633	23881	14972
`boundary_boundary_propagator`	< 1	1	1

Table 5.9.: The times spent in different parts of the program on the computer Tesla, in the single-node version, parallelized in z-direction ($1 \times 1 \times 2$) and parallelized in y- and z-direction ($1 \times 2 \times 2$)

Suppose we want to compute $k_\mathrm{V}(x_0)$. With the current version of `cf_rel`, this is done by evaluating

$$k_\mathrm{V}(x_0) \propto \sum_{k=1}^{3} \left\{ \sum_{\vec{x}\vec{y}\vec{z}} \mathrm{Tr}\, \left[\bar{S}(x)^\dagger \cdot \gamma_5\gamma_k \cdot \bar{S}(x) \cdot \gamma_K\gamma_5 \right] \right\} \tag{5.54}$$

$$\propto \sum_{k=1}^{3} \left\{ \sum_{\vec{x}\vec{y}\vec{z}} \mathrm{Tr}\, \left[\bar{S}(x)^\dagger \cdot \mathcal{C}_k \cdot \bar{S}(x) \cdot \mathcal{D}_k \right] \right\} \tag{5.55}$$

with $\mathcal{C}_k = \gamma_5\gamma_k$ and $\mathcal{D} = P_+^{(2)}\gamma_k\gamma_5(P_+^{(2)})^\dagger$. The expression in curly brackets $\{\dots\}$ can be computed by `cf_rel`, but the sum over k must be done outside.

Instead, we can now rewrite this as

$$k_\mathrm{V}(x_0) \propto \sum_{\vec{x}\vec{y}\vec{z}} \mathrm{Tr}\, \left[\bar{S}(x)^*_{A\alpha,B\beta} \cdot \bar{S}(x)_{C\alpha,D\beta} \cdot \mathcal{W}_{ABCD} \right] \tag{5.56}$$

with

$$\mathcal{W}_{ABCD} = \sum_{k=1}^{3} (\mathcal{C}_k)_{AB} \cdot (\mathcal{D}_k)_{CD} \,. \tag{5.57}$$

The sum is incorporated in the definition of $\mathcal{W}$ and the whole expression could be computed in a single call to a modified version of `cf_rel`.

However, $\mathcal{W}$ has many components, so the evaluation of the products in (5.56) would be very time-consuming. Therefore, it would be necessary to implement $\mathcal{W}$ as a sparse matrix as described above. Ideally, the generic function would then have to compute only the propagator products that appear in (2.124), like the specific function `k_v_rel`.

In any case, these modifications must be compared to the additional complexity of the program. The previous measurements suggest that there is likely no need to make the computation of the correlation functions more efficient, because it only contributes a small fraction to the total execution time.

6. Implementation of HQET Correlation Functions

For the computation of HQET correlation functions, we have not written any specific functions, i.e. routines that are dedicated to the computation of a certain correlation, like $f_{\mathrm{A}}^{\mathrm{kin}}(x_0)$. This was the way it was done on the APE, e.g. $f_{\mathrm{A}}^{\mathrm{kin}}$ was calculated by the routine `fAkin`. Instead, we have decided to use two generic functions, one for boundary-to-bulk (`cf_hqet`) and one for boundary-to-boundary correlations (`cf_hqet_1`). Their syntax and functioning is similar to their QCD counterparts `cf_rel` and `cf_rel_1`.

The first of the following sections describes how the static, kinetic and spin propagators are computed and stored. Then, in the next sections, the functions `cf_hqet` and `cf_hqet_1` are explained and how they compute correlation functions from the propagators.

6.1. The HQET Propagators

6.1.1. How the Propagators are Stored

The propagators that we need to compute the correlation functions considered in section 3.6 are the ones for a source at the bottom. Those are $\bar{S}_{\mathrm{h}}(x)$, $\bar{S}_{\mathrm{h}}^{\mathrm{kin}}(x)$ and $\bar{S}_{\mathrm{h}}^{\mathrm{spin}}(x)$ (see equations (3.68), (3.79) and (3.85)). All these propagators have a constant Dirac structure, which does not depend on x:

$$\bar{S}_{\mathrm{h}}(x) = P_+ \cdot W(x) \qquad \bar{S}_{\mathrm{h}}^{\mathrm{kin}}(x) = P_+ \cdot W^{\mathrm{kin}}(x) \qquad \bar{S}_{\mathrm{h}}(x) = P_+\sigma_j \cdot W_j^{\mathrm{spin}}(x). \tag{6.1}$$

The only parts that depend on x and on the gauge-field configuration are $W(x)$, $W^{\mathrm{kin}}(x)$ and $W_j^{\mathrm{spin}}(x)$. So, it is sufficient to store them instead of the full propagators. For the sake of simplicity, they will often be referred to as the "propagators" simply in the following sections.

$W(x)$ and $W^{\mathrm{kin}}(x)$ are fields of SU(3)-matrices. They are represented by the global variables (defined in `src/include/global.h`)

```
su3_dp *wline;
su3_dp *wkin;
```

The geometry used for these fields is that of the gauge field.

The last one, $W_j^{\mathrm{spin}}(x)$, is special, since it has an additional vector index j, which stems from the inserted chromo-magnetic field B_j. Therefore, it is stored in three fields of SU(3)-matrices corresponding to the three values of j:

```
su3_dp *wspin1;
su3_dp *wspin2;
su3_dp *wspin3;
```

On the APE, it was stored in a different way: The σ_j-matrices were included, and the object that was stored was $W_{\mathrm{APE}}^{\mathrm{spin}}(x) = \sigma_j W_j^{\mathrm{spin}}(x)$. Thus, $W_{\mathrm{APE}}^{\mathrm{spin}}(x)$ had no longer a vector index, but in turn, it became a 2×2-matrix in Dirac space because of the σ_j. We have decided to store it as $W_j^{\mathrm{spin}}(x)$, mainly because in this way we can use the same function `cf_hqet` to compute static, kin and spin correlations.

So, in the program, the components of $W(x)$, $W^{\mathrm{kin}}(x)$ and $W_j^{\mathrm{spin}}(x)$ can be accessed via

$$W(x)_{\alpha,\beta} = \texttt{wline[}i_x\texttt{].c}\alpha\beta \tag{6.2}$$

$$W^{\mathrm{kin}}(x)_{\alpha,\beta} = \texttt{wkin[}i_x\texttt{].c}\alpha\beta \tag{6.3}$$

$$\left[W_j^{\mathrm{spin}}(x)\right]_{\alpha,\beta} = \texttt{wspin}j\texttt{[}i_x\texttt{].c}\alpha\beta, \tag{6.4}$$

where i_x is the index of the point x in gauge-field geometry.

The boundary-to-boundary propagators W_{T}, $W_{\mathrm{T}}^{\mathrm{kin}}$ and $W_{\mathrm{T}}^{\mathrm{spin}}$ are SU(3)-matrices, and thus, they are stored in variables of type `su3_dp`.

6.1.2. Computation of the Propagators

The routines that compute the smeared gauge field had already been used in other projects. They were written by Bastian Knippschild and Michele Della Morte, who included them in our program. Michele Della Morte also created the functions that determine the HQET propagators $W(x)$, $W^{\mathrm{kin}}(x)$ and $W_j^{\mathrm{spin}}(x)$.

As described in section 3.3, the gauge field $U_\mu(x)$ in the HQET action is often replaced by a "smeared" field $V_\mu(x)$, which also enters the HQET propagators, which are described in section 3.5. Hence, before the propagators are computed, the smeared gauge field must be determined. At the moment, only HYP smearing (see [DM05]) is implemented in our program. It is done by the function `HYP_smearing` and the resulting field is written to the global array `PUD_sm1`. Then, the propagators can be computed in the function `compute_wline`. It calculates $W(x)$, $W^{\mathrm{kin}}(x)$ and $W_j^{\mathrm{spin}}(x)$ and stores them in the global arrays `wline`, `wkin` and `wspin1`, `wspin2`, `wspin3` (see previous section). The corresponding boundary-to-boundary propagators W_{T}, $W_{\mathrm{T}}^{\mathrm{kin}}$ and $(W_{\mathrm{T}}^{\mathrm{spin}})_j$ are computed in the function `heavy_boundary_boundary_propagator`.

The source code which computes the propagators may look like the listing 6.1.

Listing 6.1: Source code for the computation of the smeared gauge field and the HQET propagators

```c
#include "global.h"
#include "smearing.h"
#include "cf_hqet.h"

/* ... */

int main(int argc,char *argv[])
{
  /* Structure holding the smearing parameters */
  smearparm smear;
  /* HQET boundary-to-boundary propagators */
  su3_dp bbprop_h[1], bbprop_h_kin[1],
         bbprop_h_spin[3][1];
  double theta_h;  /* theta of heavy quark */

  /* ... */

  /* Smearing */
  smear.alpha[0]=1.0;
  smear.alpha[1]=1.0;
  smear.alpha[2]=0.5;
  /* PUD:      gauge field
   * PUD_sm1:  smeared gauge field */
  HYP_smearing(PUD, PUD_sm1, smear);

  /* HQET propagators */
  alloc_wline();  /* Allocation of wline, wkin,... */
  compute_wline(theta_h, theta_h, theta_h);

  /* HQET boundary-to-boundary propagators */
  heavy_boundary_boundary_propagator(wline, bbprop_h);
  heavy_boundary_boundary_propagator(wkin, bbprop_h_kin);
  heavy_boundary_boundary_propagator(wspin1,
                                     bbprop_h_spin[0]);
  heavy_boundary_boundary_propagator(wspin2,
                                     bbprop_h_spin[1]);
  heavy_boundary_boundary_propagator(wspin3,
                                     bbprop_h_spin[2]);
}
```

6.2. The Generic Routines for HQET Correlation Functions

There are two generic functions which can compute HQET correlation functions. These are `cf_hqet` for boundary-to-bulk correlation functions and `cf_hqet_1` for boundary-to-boundary correlation functions.

The next section describes how they are meant to be used. After this, their implementations are explained.

6.2.1. Usage of the Generic Routines for HQET Correlation Functions

The syntax of `cf_hqet` and `cf_hqet_1` is very similar to their QCD partners `cf_rel` and `cf_rel_1`. The first and second argument are integer constants that specify the gamma matrix combination $\mathcal{A}$ that is inserted in the bulk (for `cf_hqet`) or at the upper boundary (for `cf_hqet_1`) and the matrix $\mathcal{B}$ at the lower boundary. The third and fourth argument are the light, relativistic propagator and the heavy propagator. The last argument is a pointer to the output array. There is no argument for indicating backward correlations, because we do not plan to use such correlation functions.

Listing 6.2 shows as an example how $f_{\mathrm{A}}^{\mathrm{stat}}(x_0)$ is computed. In the listing, the heavy propagator `prop_h` means the static propagator $W(x)$, since a static correlation is to be computed. Usually, `prop_h` will point to the global array `wline`, but in principle, the propagator can be stored at any other place, too.

Listing 6.2: Computation of $f_{\mathrm{A}}^{\mathrm{stat}}$ using the function `cf_hqet`

```c
#include "cf_hqet.h"

int main(int argc,char *argv[])
{
  complex_dp out_f_a_stat[TSIZE-1];  /* output arrays */
  /* propagators */
  int prop_l;       /* light */
  su3_dp *prop_h;   /* heavy (static) */

  /* ... */

  /* f_A^stat */
  cf_hqet(GAMMA_0_5, GAMMA_5, prop_l, prop_h,
          out_f_a_stat);

  /* ... */
```

Kinetic correlations like $f_{\mathrm{A}}^{\mathrm{kin}}$ are calculated in the same way, only the static propagator is replaced by the kinetic propagator $W^{\mathrm{kin}}(x)$ or $W_{\mathrm{T}}^{\mathrm{kin}}$. Listing 6.3 shows this for $f_{\mathrm{A}}^{\mathrm{kin}}(x_0)$. Usually, `prop_h_kin` will be set to `wkin`.

Listing 6.3: Computation of $f_{\mathrm{A}}^{\mathrm{kin}}$ using `cf_hqet`

```
/* output arrays */
complex_dp out_f_a_kin[TSIZE-1];
/* propagators */
int prop_l;           /* light */
su3_dp *prop_h_kin;   /* heavy (kin) */

/* ... */

/* f_A^kin */
cf_hqet(GAMMA_0_5, GAMMA_5, prop_l, prop_h_kin,
        out_f_a_kin);
```

The determination of spin correlations like $f_{\mathrm{A}}^{\mathrm{spin}}(x_0)$ is more involved, because the scalar product $\vec{\sigma}\cdot\vec{W}^{\mathrm{spin}}(x)$ (see (6.1)) is not computed by `cf_hqet`. This must be done explicitly. A way to do it is displayed in listing 6.4. The σ_j-matrices are supplied with the matrix that is inserted at the lower boundary, here γ_5 (see the next section 6.2.2 for why this works). It is multiplied by σ_j from the left. With each σ_j, the corresponding component W_j^{spin} of the spin propagator is passed as argument, and the results for $j = 1, 2, 3$ are summed. A very similar way may be used to obtain correlation functions like $k_{\mathrm{V}}^{\mathrm{stat}}$, where a sum over different gamma matrices in the bulk and boundary insertions is to be done. The principle is also explained in section 5.4.1 for the relativistic correlations k_{T}, k_{V} and k_1.

Listing 6.4: Computation of $f_{\mathrm{A}}^{\mathrm{spin}}$ using `cf_hqet`

```
complex_dp out_f_a_spin[TSIZE-1];   /* output arrays */
complex_dp out_temp[TSIZE-1];    /* temporary results */
int x0m1;                         /* time x_0 minus 1 */
/* propagators */
int prop_l;               /* light */
su3_dp *prop_h_spin[3];   /* heavy (spin) */

/* ... */

/* f_A^spin */
cf_hqet(GAMMA_0_5, SIGMA_1_GAMMA_5, prop_l,
        prop_h_spin[0], out_f_a_spin);
cf_hqet(GAMMA_0_5, SIGMA_2_GAMMA_5, prop_l,
        prop_h_spin[1], out_temp);
for(x0m1=0; x0m1<TSIZE-1; x0m1++)
{
  out_f_a_spin[x0m1].re += out_temp[x0m1].re;
  out_f_a_spin[x0m1].im += out_temp[x0m1].im;
}
cf_hqet(GAMMA_0_5, SIGMA_3_GAMMA_5, prop_l,
        prop_h_spin[2], out_temp);
for(x0m1=0; x0m1<TSIZE-1; x0m1++)
{
  out_f_a_spin[x0m1].re += out_temp[x0m1].re;
  out_f_a_spin[x0m1].im += out_temp[x0m1].im;
}
```

Correlations that include a derivative acting on the light or heavy quark constitute another special case. As an example, we will look at $f_{A_k^{(4)}}(x_0)$, which is computed in listing 6.5. In equation (3.104), it was defined as

$$f^{\rm stat}_{A_k^{(4)}}(x_0) = \frac{\rm i}{6}\sum_{\vec{x}\vec{y}\vec{z}k}\left\langle \bar{\psi}(x)\frac{1}{2}\left(\widetilde{\nabla}_k - \overleftarrow{\widetilde{\nabla}}_k\right)\gamma_5\psi_{\rm h}(x)\bar{\zeta}_{\rm h}(\vec{y})\gamma_5\zeta(\vec{z})\right\rangle \tag{6.5}$$

and in section 3.6.3 it was shown how a generic correlation function of this type can be rewritten in terms of the propagators and how the sum of backward and forward derivatives can be turned into a single forward derivative acting on the heavy propagator with modified phase factors μ_k (equation (3.126)). If we apply this procedure to $f^{\rm stat}_{A_k^{(4)}}(x_0)$, the result (without integration over gauge fields) is

$$\begin{aligned} f^{\rm stat}_{A_k^{(4)}}(x_0) = -\frac{{\rm i}\sqrt{V_3}}{6}\sum_{\vec{x}k}{\rm Tr}\Bigg[\bar{S}(x)^\dagger \\ \times\frac{\mu_k U_k(x)W(x+\hat{k}) - \mu_k^* U_k(x-\hat{k})^\dagger W(x-\hat{k})}{2}P_+\Bigg]. \end{aligned} \tag{6.6}$$

This is the expression that is evaluated in listing 6.5: The derivative of the heavy propagator is computed by the function `compute_wline_delta` in line 18. Its arguments are the heavy propagator (here, the static propagator, but the kin and spin propagators are also possible), the coefficients of the backward and forward derivative (here, -0.5 and $+0.5$), the θ-values of the light quark in x-, y- and z-direction, the θ-values of the heavy quark and a pointer to an output array to which the calculated derivative is written.[1] Here, the propagator is written to the vari-

[1] The modified phase factors μ_k are computed automatically by `compute_wline_delta` from the θ-values and the coefficients of the derivatives according to equation (3.125).

able `prop_h_delta`. It is an array of three fields of SU(3)-matrices. Its first index specifies the space direction of the derivative:

$$\frac{1}{2}(\widetilde{\nabla}_k - \overleftarrow{\widetilde{\nabla}}_k)W(x) \quad\longleftrightarrow\quad \texttt{prop\_h\_delta}[k-1]\,[i_x]. \tag{6.7}$$

In the lines 22 to 36 the derivatives are passed to `cf_hqet` to compute the contraction with the light propagator, and the results are summed over the direction index k.

Listing 6.5: Computation of $f^{\rm stat}_{A_k^{(4)}}$

```
 1  complex_dp out_f_a4_stat[TSIZE-1];   /* output array */
 2  complex_dp out_temp[TSIZE-1];   /* temporary results */
 3  int x0m1;                        /* time x_0 minus 1 */
 4  int prop_l;                      /* light propagator */
 5  su3_dp *prop_h;         /* heavy (static) propagator */
 6  su3_dp *proph_delta;         /* derivative of prop_h */
 7
 8  /* ... */
 9
10  /* Memory allocation for derivative */
11  for(j=0; j<3; j++)
12    prop_h_delta[j] = amalloc(VOLUME*sizeof(su3_dp),
13                              ALIGN);
14  /* Computing the derivative */
15  compute_wline_delta(prop_h, -0.5, 0.5,
16                      theta_l_x, theta_l_y, theta_l_z,
17                      theta_h_x, theta_h_y, theta_h_z,
18                      prop_h_delta);
19
20  /* f_Ak4^stat */
21  cf_hqet(GAMMA_5, GAMMA_5, prop_l, prop_h_delta[0],
22          out_f_a4_stat);
23  cf_hqet(GAMMA_5, GAMMA_5, prop_l, prop_h_delta[1],
24          out_temp)
25  for(x0m1=0; x0m1<TSIZE-1; x0m1++)
26  {
27    out_f_a4_stat[x0m1].re += out_temp[1][x0m1].re;
28    out_f_a4_stat[x0m1].im += out_temp[1][x0m1].im;
29  };
30  cf_hqet(GAMMA_5, GAMMA_5, prop_l, prop_h_delta[2],
31          out_temp);
32  for(x0m1=0; x0m1<TSIZE-1; x0m1++)
33  {
    out_f_a4_stat[x0m1].re += out_temp[1][x0m1].re;
    out_f_a4_stat[x0m1].im += out_temp[1][x0m1].im;
  }
```

One can also combine the procedures described above, e.g. to compute a correlation function with derivative and spin insertion, like $f^{\mathrm{spin}}_{A^{(4)}_k}$. For this purpose, one must compute the derivative of the spin propagator, which has two vector indices and thus $3 \times 3 = 9$ components. So, `cf_hqet` must be called nine times with the appropriate gamma matrices and propagator components, and the results must be summed.

Boundary-to-boundary correlation functions are computed in a way analogous to the boundary-to-bulk correlations.[2] Only, `cf_hqet` must be replaced by `cf_hqet_1` and the propagators must be replaced by the appropriate boundary-to-boundary propagators. For example, f_1^{stat} is calculated in listing 6.6.

Listing 6.6: Computation of f_1^{stat}

```
/* output array */
complex_dp out_f_1_stat[1];
/* bnd.-to-bnd. propagators */
weyl_dp bbprop_l[6];  /* light */
su3_dp bbprop_h[1];   /* heavy (static) */

/* ... */

cf_hqet_1(GAMMA_5, GAMMA_5, bbprop_l, bbprop_h,
          out_f_1_stat);
```

Like the QCD correlation functions, the HQET correlations should be normalized in the end. The correlation that were defined in section 3.6, e.g. $f_{\mathrm{A}}^{\mathrm{stat}}$ and $f_{\delta\mathrm{A}}^{\mathrm{kin}}$, were normalized as on the APE. To get the same normalization in the program, the output of `cf_hqet` or `cf_hqet_1` must be multiplied by the factors given in table 6.1.

For other correlations the necessary normalization factors may be obtained by the following considerations:

- The function `init_rhs` omits the factor $\tilde{c}_{\mathrm{t}}/(2\sqrt{V_3})$ relative to (2.84), which is missing in the relativistic propagator.
- In boundary-to-boundary correlations, another factor $-\tilde{c}_{\mathrm{t}}/(2\sqrt{V_3})$ is left out

[2]The only exception is that boundary-to-boundary correlation with a derivative would make no sense, since they would be zero.

correlation function	additional normalization
$f_{\mathrm{A}}^{\mathrm{stat/kin}}$, $f_{\delta\mathrm{A}}^{\mathrm{stat/kin}}$, $f_{A_k^{(4)}}^{\mathrm{stat/kin}}$	$1/2 \cdot \tilde{c}_{\mathrm{t}}/2$
$f_{\mathrm{A}}^{\mathrm{spin}}$, $f_{\delta\mathrm{A}}^{\mathrm{spin}}$, $f_{A_k^{(4)}}^{\mathrm{spin}}$	$1/2 \cdot \tilde{c}_{\mathrm{t}}/2 \cdot 1/8$
$k_{\mathrm{V}}^{\mathrm{stat/kin}}$	$1/6 \cdot \tilde{c}_{\mathrm{t}}/2$
$f_1^{\mathrm{stat/kin}}$	$1/2 \cdot (\tilde{c}_{\mathrm{t}}/2)^2 \cdot 1/\sqrt{V_3}$
f_1^{spin}	$1/2 \cdot (\tilde{c}_{\mathrm{t}}/2)^2 \cdot 1/\sqrt{V_3} \cdot 1/8$

Table 6.1.: Additional factors that are needed to normalize some HQET correlation functions in the same way as on the APE

in the function `boundary_boundary_propagator`.

- The static and kinetic propagator that are computed in `compute_wline` are the same as $W(x)$ and $W^{\mathrm{kin}}(x)$. No factors are missing.
- However, the spin propagator from `compute_wline` is $8 \times \vec{W}^{\mathrm{spin}}(x)$. So, all spin correlations must be multiplied by 1/8.
- `heavy_boundary_boundary_propagator` and `compute_wline_delta` do not introduce any additional missing factors.
- In the end, every additional factor from the correlation function's definition must be taken into account. The correlations that are computed by `cf_hqet` include $-1/\sqrt{V_3}$ before the expectation value, in `cf_hqet_1` it is $1/V_3$. On the other hand, e.g. $f_{\mathrm{A}}^{\mathrm{stat}}$ has only the factor $-1/2$ in its definition (3.99). So the results of `cf_hqet` must be multiplied by $\sqrt{V_3}/2$. With all the previous factors, this amounts to $\tilde{c}_{\mathrm{t}}/4$.

In general, the results from `cf_hqet` must be multiplied by $-\tilde{c}_{\mathrm{t}}/2$, or $-\tilde{c}_{\mathrm{t}}/16$ for spin correlations, and by the factor appearing before the expectation value in their definition. The results from `cf_hqet_1` must be multiplied by $-\tilde{c}_{\mathrm{t}}^2/4$, or by $-\tilde{c}_{\mathrm{t}}^2/32$ for spin correlations, and by the factor appearing before the expectation value in their definition.

For example, the definition (3.105) of f_1^{stat} includes $-1/(2\sqrt{V_3}$. To match this definition, the results from `cf_hqet_1` need the additional normalization $-\tilde{c}_{\mathrm{t}}^2/4 \times (-1)/(2\sqrt{V_3}) = \tilde{c}_{\mathrm{t}}^2/(8\sqrt{V_3})$.

6.2.2. Evaluation of a Generic HQET Correlation Function

In section 3.6 different types of two-point HQET correlation functions with one heavy and one light quark were expressed in terms of propagators. Here, they will be transformed to their final form, which is implemented in the generic functions `cf_hqet` and `cf_hqet_1`. First, we will look at the boundary-to-bulk correlation functions.

The results that were obtained in section 3.6 for the boundary-to-bulk correlations, e.g. (3.110) for simple static correlations and (3.130) for a spin correlation with derivative, have a common structure. All of them look like

$$c(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}(x)^\dagger \cdot \gamma_5 \mathcal{A} \cdot P_+ W^X(x) \cdot \mathcal{B}\gamma_5\right], \tag{6.8}$$

where $W^X(x)$ can stand for various heavy propagators, such as $W(x)$, $W^{\mathrm{kin}}(x)$ or the derivative of a propagator. The case of spin correlations is special. There, we would actually have to insert $\sum_j \sigma_j \cdot W_j^{\mathrm{spin}}(x)$ for $W^X(x)$. But we can incorporate σ_j into the matrix $\mathcal{B}$ and perform the sum over j outside the sum over $\vec{x}$: [3]

$$c^{\mathrm{spin}}(x_0) = \sum_j \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}(x)^\dagger \cdot \gamma_5 \mathcal{A} \cdot P_+ W_j^{\mathrm{spin}}(x) \cdot \mathcal{B}'_j \gamma_5\right] \tag{6.9}$$

with $\mathcal{B}'_j = \sigma_j \cdot \mathcal{B}$. (It is also possible to include σ_j in $\mathcal{A}$, since it commutes with P_+.) So, spin correlations may be looked at as the sum of three correlations of the type in equation (6.8).

Since all correlations can be reduced to the structure of (6.8), they all can be computed by a common function, that is `cf_hqet`. But before we implement it, we will discuss how this can be done efficiently.

As in section 5.4.2, we can replace the light propagator $\bar{S}(x)$ by its reduced version $\bar{S}_{\mathrm{r}}(x)$ via equation (5.30). This leads us to

$$c(x_0) = \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}_{\mathrm{r}}(x)^\dagger \cdot \gamma_5 \mathcal{A} \cdot P_+ W^X(x) \cdot \mathcal{B}\gamma_5 \cdot \left(P_+^{(2)}\right)^\dagger\right]. \tag{6.10}$$

Since $W^X(x)$ is an SU(3)-matrix, which acts only in color space, it and the Dirac matrices γ_5, P_+, $\mathcal{A}$ and $\mathcal{B}$ can be exchanged:

$$\begin{aligned} c(x_0) &= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}_{\mathrm{r}}(x)^\dagger \cdot W^X(x) \cdot \gamma_5 \mathcal{A} P_+ \mathcal{B}\gamma_5 \left(P_+^{(2)}\right)^\dagger\right] && (6.11)\\ &= \sum_{\vec{x}} \mathrm{Tr}\left[\bar{S}_{\mathrm{r}}(x)^\dagger \cdot W^X(x) \cdot \mathcal{D}\right] && (6.12)\end{aligned}$$

with the 4×2-matrix

$$\mathcal{D} = \gamma_5 \mathcal{A} P_+ \mathcal{B} \gamma_5 \left(P_+^{(2)}\right)^\dagger. \tag{6.13}$$

The matrix $\mathcal{D}$ is constant and does not depend on x. So, it need only be computed once.

[3] σ_j acts in Dirac space, and therefore it can be exchanged with the SU(3)-matrix $W^{\mathrm{spin}}(x)$.

Now, at every lattice site x, there is a product of three factors, $\bar{S}_\mathrm{r}(x)^\dagger$, $W^X(x)$ and $\mathcal{D}$. Written in components, it is

$$c(x_0) = \sum_{\vec{x}} \bar{S}_\mathrm{r}(x)^*_{A\alpha,B\beta} \cdot W^X(x)_{\alpha,\beta} \cdot \mathcal{D}_{A,B}. \tag{6.14}$$

In the above equation, Einstein notation is used. The index A runs from 1 to 4, B from 1 to 2 and α and β from 1 to 3.

The product can be evaluated in three different orders. To estimate the time for the computation after each order, I have counted the number of complex multiplications that must be done. (The same strategy was used in section 5.5.4.) The results are shown in table 6.2.

product	complex multiplications
$\left[(\bar{S}_\mathrm{r})^*_{A\alpha,B\beta} \cdot W^X_{\alpha\beta}\right] \cdot \mathcal{D}_{AB}$	80
$\left[W^X_{\alpha\beta} \cdot \mathcal{D}_{AB}\right] \cdot (\bar{S}_\mathrm{r})^*_{A\alpha,B\beta}$	144
$\left[(\bar{S}_\mathrm{r})^*_{A\alpha,B\beta} \cdot \mathcal{D}_{AB}\right] \cdot W^X_{\alpha\beta}$	81

Table 6.2.: Number of complex multiplications for the computation of each summand in equation (6.14)

Following this table, the best way is to contract $\bar{S}_\mathrm{r}$ and W^X first. One can then sum over their color indices and the result is a 2×4-matrix in Dirac space (or a 4×2-matrix, depending on how the indices are arranged), which is then contracted with $\mathcal{D}$. So, the way in which `cf_hqet` computes a correlation is described by

$$\begin{aligned} c(x_0) &= \sum_{\vec{x}} \left[\bar{S}_\mathrm{r}(x)^*_{A\alpha,B\beta} \cdot W^X(x)_{\alpha,\beta}\right] \cdot \mathcal{D}_{A,B} \\ &= \sum_{\vec{x}} \mathrm{Tr}\left\{\mathrm{Tr}_\mathrm{col}\left[\bar{S}_\mathrm{r}(x)^\dagger \cdot W^X(x)\right] \cdot \mathcal{D}\right\}. \end{aligned} \tag{6.15}$$

`cf_hqet_1` evaluates boundary-to-boundary correlations in the same way, although it is not necessary to pay attention to efficiency there, since no sum over the lattice points must be done and the function will take only a short time anyway.

In section 3.6, the different types of boundary-to-boundary correlations could be converted to the form

$$c_1 = \mathrm{Tr}\left\{\gamma_5 \bar{S}_\mathrm{T}^\dagger \gamma_5 \cdot \mathcal{A} \cdot P_+ W_\mathrm{T}^X \cdot \mathcal{B}\right\}. \tag{6.16}$$

(see (3.134), (3.137) and (3.138)). Here, W_T^X may stand for W_T, $W_\mathrm{T}^\mathrm{kin}$ or $(W_\mathrm{T}^\mathrm{spin})_j$. As for the boundary-to-bulk correlations, boundary-to-boundary correlations with a spin insertion can be seen as a sum of three correlation of this type, where the Pauli matrices σ_j are included in $\mathcal{B}$ or $\mathcal{A}$.

If the relativistic propagator is substituted with its reduced version $\bar{S}_{\mathrm{Tr}}$ (see equation (5.46)), we obtain

$$c_1 = \mathrm{Tr}\left\{\bar{S}_{\mathrm{Tr}}^{\dagger} \cdot P_+^{(2)}\gamma_5\mathcal{A} \cdot P_+ W_{\mathrm{T}}^X \cdot \mathcal{B}\gamma_5 \left(P_+^{(2)}\right)^{\dagger}\right\}. \tag{6.17}$$

The propagator W_{T}^X and the matrices acting in Dirac space can be exchanged, so we may write

$$c_1 = \mathrm{Tr}\left\{\bar{S}_{\mathrm{Tr}}^{\dagger} \cdot W_{\mathrm{T}}^X \cdot \mathcal{D}\right\} \tag{6.18}$$

with

$$\mathcal{D} = P_+^{(2)}\gamma_5\mathcal{A}P_+\mathcal{B}\gamma_5 \left(P_+^{(2)}\right)^{\dagger}. \tag{6.19}$$

Again, the order of evaluation with the fewest multiplications of complex numbers is to contract $\bar{S}_{\mathrm{Tr}}^{\dagger}$ and W_{T}^X first (40 complex products):

$$c_1 = \mathrm{Tr}\left\{\mathrm{Tr}_{\mathrm{col}}\left[\bar{S}_{\mathrm{Tr}}^{\dagger} \cdot W_{\mathrm{T}}^X\right] \cdot \mathcal{D}\right\}. \tag{6.20}$$

6.2.3. The Functions `cf_hqet` and `cf_hqet_1`

The functions `cf_hqet` and `cf_hqet_1` are to compute generic boundary-to-bulk and boundary-to-boundary HQET correlations as described in the previous section. First, we will have a look at the implementation of `cf_hqet`. Later, we will examine the working of `cf_hqet_1`, but only briefly, since it is very similar to `cf_hqet`.

`cf_hqet` evaluates a correlation function according to equation (6.15). Its source code is printed in listing 6.7, and the list below explains what is done in each part.

lines 13–16 These lines compute the matrix product $\gamma_5\mathcal{A}P_+\mathcal{B}\gamma_5$. (The constant matrix `p_plus` representing P_+ is set in line 6.)

lines 17–20 The final matrix $\mathcal{D}$ is computed (see (6.13)): The product of $\gamma_5\mathcal{A}P_+\mathcal{B}\gamma_5$ and $(P_+^{(2)})^{\dagger}$ is written in components and the result is stored in `d`, which is of the type `mat42`. This type is defined in the same file and represents complex 4×2-matrices.

line 22 Loop over the time coordinate x_0. In each iteration, $c(x_0)$ is computed.

lines 24–25 The array `out_local` will be used for the sum over all points $(x_0, \vec{x})$ with constant x_0 of the local sub-lattice. Here, at the beginning, it is initialized to zero.

line 27 Loop over all points $(x_0, \vec{x})$ with a constant x_0. In contrast to `cf_rel`, the lexicographic index is used here. This means that the actual indices used by the program must be looked up in the appropriate tables (see lines 34–36). This detour is necessary, because we will have to access the light propagator that uses spinor-field geometry as well as the heavy propagator that uses gauge-field geometry, and both geometries may be different (see section 4.1). Therefore, we have to fall back to the lexicographic index as a method to address points independently of the current geometry.

lines 34–36 The lexicographic index `lex` is turned into the index after spinor geometry `ind_s`, and into the index after gauge geometry `ind_u`. A pointer `w` is set to the heavy propagator $W^X(x)$ at the current point $x = (x_0, \vec{x})$.

lines 38–40 Pointers to the relativistic propagator $\bar{S}_\mathrm{r}(x)_{A\alpha,B\beta}$ for the second Dirac index $B = 1$ are set. `psi1`, `psi2` and `psi3` correspond to the three values of the second color index $\beta = 1, 2, 3$.

lines 41–63 The product of light and heavy propagator $\mathcal{E}_{BA} = \sum_{\alpha\beta} \bar{S}_\mathrm{r}(x)^*_{A\alpha,B\beta} \cdot W^X(x)_{\alpha\beta}$ is calculated and written to the 2×4-matrix `e`. In fact, only the first half of $\mathcal{E}_{BA}$ is computed here, for $B = 1$. The second half follows in lines 67–71.

lines 64–66 The pointers `psi1`, `psi2` and `psi3` are set to the components of the relativistic propagator $\bar{S}_\mathrm{r}(x)_{A\alpha,B\beta}$ with $B = 2$.

lines 67–71 The second half of $\mathcal{E}_{BA}$, with $B = 2$, is computed and written to `e`.

lines 74–82 The contraction of $\mathcal{E}$ and $\mathcal{D}$ is done, $\sum_{AB} \mathcal{E}_{BA}\mathcal{D}_{AB}$. The result is added to the local sum in `out_local[x0-1]`.

lines 85–86 The sums in `out_local[x0-1]` over the local lattice points are combined into the global sum, which is returned in the output array `out`.

Listing 6.7: Source code of the function `cf_hqet` (file `src/modules/cf_hqet/cf_hqet.c`)

```
 1  void cf_hqet(const int gamma_bulk,
 2               const int gamma_boundary, const int prop_l,
 3               const su3_dp *const prop_h,
 4               complex_dp *const out)
 5  {
 6    const mat4 p_plus={{0.5, 0.0}, {0.0, 0.0}, /* ... */
 7    mat4 temp1, temp2;
 8    mat42 d;
 9    int x0, lex;
10    complex_dp out_local[TSIZE-1];
11
12    _mat4_times_mat4(temp1, gamma[GAMMA_5],
                    gamma[gamma_bulk]);
_mat4_times_mat4(temp2, temp1, p_plus);
_mat4_times_mat4(temp1, temp2, gamma[gamma_boundary]);
_mat4_times_mat4(temp2, temp1, gamma[GAMMA_5]);
d.c11.re = temp2.c11.re-temp2.c13.re;
d.c12.re = temp2.c12.re-temp2.c14.re;
d.c21.re = temp2.c21.re-temp2.c23.re;
/* ... */

for(x0=1; x0<TSIZE; x0++)
{
  out_local[x0-1].re = 0.0;
  out_local[x0-1].im = 0.0;

  for(lex=x0*TSLICE; lex<(x0+1)*TSLICE; lex++)
  {
    mat24 e;
    int ind_s, ind_u;
    const su3_dp *w;
    const spinor_dp *psi1, *psi2, *psi3;

    ind_s = s_ipt[lex];
    ind_u = u_ipt[lex];
    w = prop_h+ind_u;

    psi1 = PSD1e[prop_l+0*3+0]+ind_s;
    psi2 = PSD1e[prop_l+0*3+1]+ind_s;
    psi3 = PSD1e[prop_l+0*3+2]+ind_s;
    e.c11.re = +psi1->c1.c1.re*w->c11.re
               +psi1->c1.c1.im*w->c11.im
               +psi1->c1.c2.re*w->c21.re
               +psi1->c1.c2.im*w->c21.im
               +psi1->c1.c3.re*w->c31.re
               +psi1->c1.c3.im*w->c31.im
               +psi2->c1.c1.re*w->c12.re
               +psi2->c1.c1.im*w->c12.im
               +psi2->c1.c2.re*w->c22.re
               +psi2->c1.c2.im*w->c22.im
               +psi2->c1.c3.re*w->c32.re
               +psi2->c1.c3.im*w->c32.im
               +psi3->c1.c1.re*w->c13.re
               +psi3->c1.c1.im*w->c13.im
               +psi3->c1.c2.re*w->c23.re
               +psi3->c1.c2.im*w->c23.im
```

```
                 +psi3->c1.c3.re*w->c33.re
                 +psi3->c1.c3.im*w->c33.im;
      e.c11.im = +psi1->c1.c1.re*w->c11.im
                 -psi1->c1.c1.im*w->c11.re
                 +psi1->c1.c2.re*w->c21.im
                 -psi1->c1.c2.im*w->c21.re
                 /* ... */
      psi1 = PSD1e[prop_l+1*3+0]+ind_s;
      psi2 = PSD1e[prop_l+1*3+1]+ind_s;
      psi3 = PSD1e[prop_l+1*3+2]+ind_s;
      e.c21.re = +psi1->c1.c1.re*w->c11.re
                 +psi1->c1.c1.im*w->c11.im
                 +psi1->c1.c2.re*w->c21.re
                 +psi1->c1.c2.im*w->c21.im
                 /* ... */

      out_local[x0-1].re += +e.c11.re*d.c11.re
                            -e.c11.im*d.c11.im
                            +e.c12.re*d.c21.re
                            -e.c12.im*d.c21.im
                            /* ... */
      out_local[x0-1].im += +e.c11.re*d.c11.im
                            +e.c11.im*d.c11.re
                            +e.c12.re*d.c21.im
                            +e.c12.im*d.c21.re
                            /* ... */
    }

    mpc_gsum_d(&out_local[x0-1].re, &out[x0-1].re, 1);
    mpc_gsum_d(&out_local[x0-1].im, &out[x0-1].im, 1);
  }
}
```

The function `cf_hqet_1` works in a very similar way. It evaluates boundary-to-boundary correlation functions as described in equation (6.20). In contrast to `cf_hqet`, no sum over the lattice points is needed (this has already been done in `boundary_boundary_propagator` and `heavy_boundary_boundary_propagator`) and the relativistic propagator has the additional symmetry $P_+ \cdot \bar{S}_\mathrm{T} = \bar{S}_\mathrm{T}$, so it can be reduced to a 2×2-matrix in Dirac space.

The source code of `cf_hqet_1` can be found in listing 6.8. The list below explains shortly what it does.

lines 12–16 The product $\gamma_5 \mathcal{A} P_+ \mathcal{B} \gamma_5$ is computed.

lines 17–23 The matrix $\mathcal{D}$ (see (6.19)) is obtained by multiplying the above product by $P_+^{(2)}$ from the left and by $(P_+^{(2)})^\dagger$ from the right.

lines 25–27 Pointers to the components of the light boundary-to-boundary propagator $(\bar{S}_{\mathrm{Tr}})_{A\alpha,B\beta}$ with $B = 1$ are set.

lines 28–38 The first half of the matrix $\mathcal{E}_{BA} = \sum_{\alpha\beta}(\bar{S}_{\mathrm{Tr}})^*_{A\alpha,B\beta} \cdot (W_{\mathrm{T}})_{\alpha\beta}$ is calculated (for $B = 1$).

lines 39–46 The last two steps are repeated for $B = 2$.

lines 48–53 The matrices $\mathcal{E}$ and $\mathcal{D}$ are contracted. The result $\sum_{AB} \mathcal{E}_{BA}\mathcal{D}_{AB}$ is written to the output variable `out`.

Listing 6.8: Source code of `cf_hqet_1` (file `src/modules/cf_hqet/cf_hqet.c`)

```
 1  void cf_hqet_1(const int gamma_top,
 2                const int gamma_bottom,
 3                const weyl_dp *const bbprop_l,
 4                const su3_dp *const bbprop_h,
 5                complex_dp *const out)
 6  {
 7    const mat4 p_plus={{0.5, 0.0}, {0.0, 0.0}, /* ... */
 8    mat4 temp1, temp2;
 9    mat2 d, e;
10    const weyl_dp *psi1, *psi2, *psi3;
11
12    _mat4_times_mat4(temp1, gamma[GAMMA_5],
13                     gamma[gamma_top]);
14    _mat4_times_mat4(temp2, temp1, p_plus);
15    _mat4_times_mat4(temp1, temp2, gamma[gamma_bottom]);
16    _mat4_times_mat4(temp2, temp1, gamma[GAMMA_5]);
17    d.c11.re = temp2.c11.re-temp2.c13.re-temp2.c31.re
18             +temp2.c33.re;
19    d.c12.re = temp2.c12.re-temp2.c14.re-temp2.c32.re
20             +temp2.c34.re;
21    d.c21.re = temp2.c21.re-temp2.c23.re-temp2.c41.re
22             +temp2.c43.re;
23    /* ... */
24
25    psi1 = bbprop_l+0*3+0;
26    psi2 = bbprop_l+0*3+1;
27    psi3 = bbprop_l+0*3+2;
28    e.c11.re = +psi1->c1.c1.re*bbprop_h->c11.re
```

```
                 +psi1->c1.c1.im*bbprop_h->c11.im
                 +psi1->c1.c2.re*bbprop_h->c21.re
                 +psi1->c1.c2.im*bbprop_h->c21.im
                 /* ... */
    e.c11.im = +psi1->c1.c1.re*bbprop_h->c11.im
                 -psi1->c1.c1.im*bbprop_h->c11.re
                 +psi1->c1.c2.re*bbprop_h->c21.im
                 -psi1->c1.c2.im*bbprop_h->c21.re
                 /* ... */
    /* ... */
    psi1 = bbprop_l+1*3+0;
    psi2 = bbprop_l+1*3+1;
    psi3 = bbprop_l+1*3+2;
    e.c21.re = +psi1->c1.c1.re*bbprop_h->c11.re
                 +psi1->c1.c1.im*bbprop_h->c11.im
                 +psi1->c1.c2.re*bbprop_h->c21.re
                 +psi1->c1.c2.im*bbprop_h->c21.im
    /* ... */

    out->re = +e.c11.re*d.c11.re-e.c11.im*d.c11.im
                +e.c12.re*d.c21.re-e.c12.im*d.c21.im
                +e.c21.re*d.c12.re-e.c21.im*d.c12.im
                +e.c22.re*d.c22.re-e.c22.im*d.c22.im;
    out->im = +e.c11.re*d.c11.im+e.c11.im*d.c11.re
                /* ... */
}
```

7. Main Program

7.1. Summary of the Computational Steps

This chapter is to provide a summary of the steps that are necessary to compute QCD or HQET correlation functions. These steps will be done in a main program, but yet there is no finished general-purpose main program, only test versions[1]. The final main program will probably deviate from the source-code examples presented in the previous chapters, but the basic computations that are to be done stay the same.

Initialization At the beginning of the program several initialization routines must be executed. These are the initialization of the MPI system (`mpc_start`), of the geometry arrays `u_ipt`, `s_ipt` etc. (`geometry`), the allocation of memory for the gauge field (`alloc_ud`), the smeared gauge field (`alloc_pud_sm1`), the spinor fields (`alloc_sd`) and the field for the Sheikoleslami–Wohlert term (`alloc_swd`).

Input If they are not hard-coded into the program, it will have to read and interpret input parameters and files. These are, e.g., the values of N_f, β, the $\vec{\theta}$ and κ values, smearing parameters and parameters for the solver.

Also the gauge field must be read in (`read_ud`), since the SFCF program does not include an update routine so far.

Relativistic propagators The relativistic QCD propagators can be computed as shown in listing 5.1, a boundary-to-boundary propagator is computed in listing 5.2. This is the step that consumes the greatest part of the running time, because the inversion of the Dirac operator is very laborious.

The propagators may be computed all at once, before they are used to compute correlation functions. This was the way it was done on the APE. A possible problem is that the propagators can need very much memory. For example, on a 32^4-lattice one relativistic propagator stored in double precision occupies about one gigabyte. To reduce the necessary memory space, one can combine the computation of the propagators and of the correlation functions. Then, old propagators that are not needed for the following correlations can be overwritten by new ones. For example, if only the correlations with equal $\vec{\theta}$ and κ values for both quarks are computed, only

[1]For example `src/tests/src/tst-cw-cf-rel.c` computes QCD correlations and `src/tests/src/tst-cw-cf-hqet.c` HQET correlations. However, the structure of both programs is quite different.

one propagator need be kept in memory. After each step it would be overwritten by the propagator for the next pair of $\vec{\theta}$ and κ. However, this depends on the choice of $\vec{\theta}$ and κ pairs for which correlations are to be computed. In a worst-case scenario, still all propagators would have to be kept. If this is not possible because of memory limitations, some propagators may be written to hard disk and read from there, when they are needed again. I suppose this is much faster than to compute them anew (see table 5.5).

HQET propagators The computation of the HQET propagators is done in listing 6.1. At first the smeared gauge field $V_\mu(x)$ is calculated, and memory for the propagators is reserved by `alloc_wline`. If we need several different propagators (e.g. with different smearings), we must allocate memory for each propagator and copy the values of the global variables `wline`, `wkin` etc. each time, so we are still able to access them later. The propagators themselves are determined in `compute_wline`. This function needs much less time than the computation of the relativistic propagators, because the static equation of motion (3.59) can be solved analytically. Thus, in contrast to the relativistic propagators, it can be faster to compute them a second time than to read them from disk, if they can not be kept in memory. They need not be computed as often as the relativistic propagators anyway, because they only depend on the smearing parameters and the kinetic propagator $W^{\text{kin}}(x)$ also on $\vec{\theta}$.

The HQET boundary-to-boundary propagators can be obtained via the routine `heavy_boundary_boundary_propagator`.

Correlation functions With the propagators, the QCD or HQET correlation functions can be calculated. Although the routines that are specialized in one QCD correlation (e.g. `f_a_rel`) are still part of the program, the computation will probably be done via the generic functions as described in section 5.4.1 for QCD and section 6.2.1 for HQET. This will be done in a loop over all desired combinations of $\vec{\theta}$, κ and smearing parameters, possibly together with the computation of propagators.

The computation of correlations by means of the generic functions, in particular spin correlations and correlations in the vector channel, is rather laborious, because the sum over vector indices must be done explicitly. This may be changed (see section 7.2).

Output Finally, the program must output the computed correlation functions. On the APE, they were written in a certain order to a binary output file.

7.2. Suggestions for a Future Main Program

One of the next steps will be to write a standard main program that can be used in the planned research projects. In this section, I have collected some suggestions how it can be designed to be easily adaptable.

- As Jochen Heitger suggested, the basic structure of the APE program could be adopted, and the computation of the propagators and the correlations could be encapsulated in an own measurement function. This would simplify the integration of an update routine for the gauge field. In each iteration of a loop, the program would call the update routine first and then the measurement routine to compute the correlation functions on the current gauge field.
- There could be an own function for the computation of relativistic propagators, which would call `init_rhs` and `solve` in turn.
- At the moment the main program must keep track of what is stored in the fermionic fields. To simplify this, the fields could have static roles, e.g. as the propagator for a certain quark type (i.e. a defined combination of $\vec{\theta}$ and κ) or as fields to be used by the solver. The indices of the propagator fields could be stored in the structure `param_rel_quark_t`. (This structure is defined in `src/include/param.h` and contains only the κ and $\vec{\theta}$ values of a quark type at the moment. It is part of the structure `param_solver_t`, which combines several parameters for the solver routine.)
- The computation of complex correlations, e.g. spin correlation functions or correlations including a derivative, could be automatized in one or more dedicated functions, which would call `cf_rel` with the appropriate arguments and sum the results if necessary. One could create an own routine for each single correlation function, or write routines for classes of correlation functions, e.g. all of them with a spin insertion, or there could be even only one routine, which accepts a structure as argument describing the desired correlation function. It could incorporate the computation of propagators, too. A call to such a function might look like

```
cf_one_routine(cf_type, quark_1, quark_2, out);
```

  where `cf_type` defines the correlation function to be computed, e.g. $f_{\mathrm{A}}^{\mathrm{spin}}$, `quark_1` and `quark_2` are structures that specify the quark flavors (the types `param_rel_quark_t` and `param_hqet_quark_t` may be used here) and `out` is the output array.

  An advantage of this approach is that the underlying computational structure need not be revealed to the main program, so it can be changed, e.g. if it proves inefficient, without altering the main program. However, it may not be easy to implement. In particular, I do not know how a structure could be organized that describes correlation functions in a simple, comprehensive and easy to evaluate way.

8. A First Application – the Static Limit of QCD Observables

After the implementation of correlation functions in the SFCF program has been described in the previous chapters, we now look at a first practical application: the investigation of the behavior of heavy–light QCD observables in the static limit, i.e. when the mass of one of the quark flavors becomes very large. In this limit, they are expected to approach the static approximation of HQET, so a comparison of both can serve as a test for the validity of the HQET expansion.

I thank Jochen Heitger, who computed the necessary QCD correlation functions on the APE in earlier stages, and later using the SFCF program on the PALMA cluster computer of the university of Münster, and Michael Topp, who analyzed them and computed the continuum limit of the observables [Top12]. I have also taken data for three different mass values from [DM07].

8.1. Parameters and QCD Observables

In the following, I will give a sketch of the simulation setup. It will closely follow the explanations in [Hei04a], as this whole chapter. Like them, we work in a Schrödinger-functional space-time with equal extents in all direction, $L \times L \times L \times L$. The size L is implicitly defined by the renormalized coupling $\bar{g}$ in the Schrödinger-functional scheme:

$$\bar{g}^2(L/4) = 1.8811. \tag{8.1}$$

This results in $L \approx 0.36\,\mathrm{fm}$ (see also [Hei04b, p. 6], note that there the argument of $\bar{g}^2$ is $L/2$).

The mass of the heavy quark is parametrized by the variable z:

$$\frac{1}{z} = \frac{1}{ML}, \tag{8.2}$$

where M is the renormalization-group invariant (RGI) mass. The (PCAC current) mass of the light quark is set to zero. This is possible, since the boundary conditions of the Schrödinger functional induce a gap in the eigenvalues of the Dirac operator [Fri09, p. 44]. For the θ-phases, two different values have been used: $\theta_0 = \theta_1 = 0.5$ and $\theta_2 = 1.0$ (the names are to be consistent with the θ-values in the observables from section 1.2), but in each single correlation function, they were the same for every flavor and in every direction.

The renormalized parameters can be translated to bare parameters $(L/a, g_0, \kappa_\mathrm{l})$ at different finite lattice spacings a via the data from [Cap99]. (For how the heavy hopping parameter κ_h that corresponds to a certain z-value was determined see [Hei04b, p. 8].) These can then be used in lattice simulations, and the renormalized observables can be extrapolated to the continuum limit $a \to 0$.

The QCD observables that we consider are basically the matching observables defined in section 1.2, but here we will leave out some prefactors and logarithms:

$$\Gamma_\mathrm{PS}(L,M) = -\,\widetilde{\partial}_0 \ln\left(f_\mathrm{A}(x_0,\theta_0)\right)\Big|_{x_0=L/2} \tag{8.3}$$

$$Y_\mathrm{PS}(L,M) = \frac{f_\mathrm{A}(L/2,\theta_0)}{\sqrt{f_1(\theta_0)}} \tag{8.4}$$

$$\Theta_\mathrm{A}(L,M) = \frac{f_\mathrm{A}(L/2,\theta_1)}{f_\mathrm{A}(L/2,\theta_2)} \tag{8.5}$$

$$\Theta_1(L,M) = \frac{f_1(\theta_1)k_1(\theta_1)^3}{f_1(\theta_2)k_1(\theta_2)^3} \tag{8.6}$$

$$C_1(L,M) = \frac{f_1(\theta_0)}{k_1(\theta_0)}. \tag{8.7}$$

(Here, the θ-angles that are to be imposed have been written as additional arguments to the correlation functions.)

8.2. Expectations from the Static Approximation

For the first three observables, we can infer expectations from the static approximation, we will start with Y_PS: The static approximation X of Y_PS is simply obtained by replacing the correlation functions in its definition by their static counterparts:

$$X(L) = \frac{f_\mathrm{A}^\mathrm{stat}(L/2)}{\sqrt{f_1^\mathrm{stat}}}. \tag{8.8}$$

X does not depend on a finite value of M any more, and the correlation functions are independent of θ, since the heavy quark does not propagate in space. (θ does not enter the definition of the static propagator W, it appears only in the subleading terms that involve the kinetic propagator W^kin, see section 3.5.) At the classical level X would be the limit of Y_PS for $1/z \to 0$, but in the quantum theory, renormalization must be taken into account. Since the static approximation breaks chiral symmetry, the static axial current acquires a scale-dependent (and scheme-dependent) renormalization factor $Z_\mathrm{A}^\mathrm{stat}(\mu)$:

$$X_\mathrm{R}(L,\mu) = Z_\mathrm{A}^\mathrm{stat}(\mu) \cdot X(L). \tag{8.9}$$

In [Hei04a, p. 8] the renormalization-group invariant current X_RGI is introduced instead, which does not depend on μ any more:

$$X_\mathrm{RGI}(L) = \lim_{\mu\to\infty} \left\{ \left[2b_0 \cdot \bar{g}^2(\mu)\right]^{-\gamma_0/(2b_0)} \cdot X_\mathrm{R}(L,\mu) \right\} = Z_\mathrm{RGI} \cdot X(L) \tag{8.10}$$

with

$$b_0 = \frac{11}{(4\pi)^2} \qquad \gamma_0 = -\frac{1}{(4\pi)^2} \qquad \text{(for } N_\mathrm{f} = 0\text{)}. \tag{8.11}$$

A preliminary result for X_RGI on the given lattice size has been reconstructed from old data:

$$X_\mathrm{RGI} = -1.35(1). \tag{8.12}$$

The real error of X_RGI is probably greater than 0.01, because there are additional uncertainties, e.g. in the continuum extrapolation that have not yet been taken fully into account. We will try to provide a better estimate soon.

In order to compare X_RGI to the QCD observable Y_PS, we still need to take into account a conversion function C_PS. So, the relation between them is expected to be

$$Y_\mathrm{PS}(L, M) = C_\mathrm{PS}(x) \cdot X_\mathrm{RGI}(L) \cdot \big(1 + \mathcal{O}(z^{-1})\big) \qquad \text{for } M \to \infty \tag{8.13}$$

with

$$x = \frac{1}{\ln\big(M/\Lambda_{\overline{\mathrm{MS}}}\big)}. \tag{8.14}$$

Here, the conversion function has been parametrized by (the inverse of the logarithm of) the RGI mass in units of the Λ parameter in the $\overline{\mathrm{MS}}$ scheme. The multiplication by C_PS can be interpreted as the conversion to another renormalization scheme, the "matching scheme", where the static results from HQET are to match those from QCD at a certain mass scale up to $\mathcal{O}(m^{-1})$ corrections [Hei03, p. 12]. An approximation of C_PS will be quoted below.

The QCD observable Γ_PS approaches the heavy quark's mass as it becomes large [Blo10, p. 19]. As for Y_PS, we need a conversion function C_mass to match it to the finite mass $M = z/L$:

$$\Gamma_\mathrm{PS}(L, M) = C_\mathrm{mass}(x) \cdot \frac{z}{L} \cdot \big(1 + \mathcal{O}(z^{-1})\big) \qquad \text{for } M \to \infty. \tag{8.15}$$

The conversion functions have been re-parametrized for our purposes in [Hei04a, p. 23] from the known perturbative ingredients. It was found that the results can be well approximated by

$$C_\mathrm{PS}(x) = x^{\gamma_0^\mathrm{PS}/(2b_0)} \cdot \big\{1 - 0.068 \cdot x - 0.087 \cdot x^2 + 0.079 \cdot x^3\big\} \tag{8.16}$$

$$C_\mathrm{mass}(x) = x^{d_0/(2b_0)} \cdot \big\{1 + 0.179 \cdot x + 0.694 \cdot x^2 + 0.065 \cdot x^3\big\} \tag{8.17}$$

with the anomalous-dimension coefficients

$$\gamma_0^\mathrm{PS} = -\frac{1}{4\pi^2} \qquad d_0 = \frac{8}{(4\pi)^2}. \tag{8.18}$$

For $x < 0.6$ these fits are guaranteed to give a precision of at least 0.2%. (In our case, x is stays always below 0.32.) The argument x can be calculated from z via

$$x = \frac{1}{\ln\big(z/(L\Lambda_{\overline{\rm MS}})\big)} \qquad \text{with } L\Lambda_{\overline{\rm MS}} = 0.216. \tag{8.19}$$

The value of $L\Lambda_{\overline{\rm MS}}$ was inferred from [Cap99, p. 15].

For the QCD observable C_1, we expect that it approaches one for large masses, because the static correlations $f_1^{\rm stat}$ and $k_1^{\rm stat}$ are the same due to the spin symmetry of the static action (see e.g. [Doo11, p. 14]):[1]

$$C_1(L,M) \approx \frac{f_1^{\rm stat}}{k_1^{\rm stat}} = 1 \qquad \text{for } M \to \infty. \tag{8.20}$$

For the last two observables $\Theta_{\rm A}$ and Θ_1, I can not give a prediction of their large-mass behavior at the moment. Nevertheless, their data are included here.

8.3. Results

The results for the QCD observables in the continuum limit at different masses z are listed in table 8.1. The given errors of the observables do not take into account the uncertainty in determining z. In addition to the results that were computed by Jochen Heitger and analyzed together with Michael Topp, also results from [DM07] for $z = 10.4, 12.1, 13.3$ are included. They are plotted in the figures 8.1, 8.2, 8.4, 8.5 and 8.3. For each graph, a linear and a quadratic curve have been fitted to the data over the whole z range. In all graphs except for figure 8.2, only the linear fit is shown, because the quadratic fit curve is too close to it. The expected limit for $1/z \to 0$ has been included in the fit, when it was known for the observable. The fit parameters can be found in table 8.2. They and their errors have been determined by the general-linear-least-squares method described in [Pre92, p. 671]. As a reference point, the inverse mass $1/z_{\rm b}$ of the b quark is marked in the plots.

In the plots of $\Gamma_{\rm PS}$ (figure 8.1) and $Y_{\rm PS}$ (figure 8.2), the conversion functions have been taken into account. Instead of plotting the observables themselves, the ratios

$$\frac{L\Gamma_{\rm PS}}{zC_{\rm mass}} = \big(1 + \mathcal{O}(z^{-1})\big) \qquad \frac{Y_{\rm PS}}{C_{\rm PS}} = X_{\rm RGI} \cdot \big(1 + \mathcal{O}(z^{-1})\big) \tag{8.21}$$

are shown, because we expect that they linearly approach their expected limits, 1.0 and $X_{\rm RGI}$.

For $\Gamma_{\rm PS}$ the fit even becomes nearly a horizontal line. The data points show only small deviations from one, and the slope of the linear fit is within its accuracy zero. $Y_{\rm PS}$ shows a greater tendency towards quadratic behavior, if one looks at the

[1] One may also derive this via the expression (6.14) for a generic HQET correlation function in terms of propagators. The matrix $\mathcal{D}$ (defined in (6.13)) that enters this expression comes out the same for both $f_1^{\rm stat}$ and $k_1^{\rm stat}$.

z	Γ_{PS}	Y_{PS}	Θ_{A}	Θ_1	C_1
12.48	0.718(11)	−1.575(6)	1.611(9)	3.112(14)	1.0162(9)
14	0.706(11)	−1.590(6)	1.596(8)	3.050(13)	1.0148(8)
15	0.699(11)	−1.598(6)	1.588(8)	3.015(13)	1.0140(8)
16	0.692(11)	−1.606(6)	1.580(8)	2.984(12)	1.0133(7)
18	0.680(11)	−1.620(6)	1.568(8)	2.932(12)	1.0121(7)
20	0.668(11)	−1.631(6)	1.559(7)	2.891(12)	1.0111(6)
The following values have been taken from [DM07]:					
10.4	0.738(10)	−1.546(5)	1.643(7)	3.226(7)	1.0193(7)
12.1	0.723(10)	−1.567(5)	1.620(6)	3.133(7)	1.0172(7)
13.3	0.714(10)	−1.580(5)	1.607(6)	3.080(7)	1.0160(6)

Table 8.1.: The QCD observables in the continuum limit for various masses z

	linear fit $a + b/z$		quadratic fit $a + b/z + c/z^2$		
	a	b	a	b	c
$L\Gamma_{\mathrm{PS}}/(zC_{\mathrm{mass}})$	1	0.00(7)	1	0.0(4)	0.(5)
$Y_{\mathrm{PS}}/C_{\mathrm{PS}}$	−1.336(6)	0.52(8)	−1.348(10)	0.9(3)	−3.(2)
Θ_{A}	1.465(12)	1.86(17)	1.47(6)	1.8(16)	1.(11)
Θ_1	2.531(17)	7.3(2)	2.47(8)	9.(2)	−11.(14)
C_1	1	0.210(3)	1	0.240(17)	−0.4(2)

Table 8.2.: Parameters for linear and quadratic fits to the data shown in figures 8.1, 8.2, 8.4, 8.5 and 8.3 (except for the figure 8.2, only the linear fit is drawn in the graphs), the constant parameter a is exact for $L\Gamma_{\mathrm{PS}}/(zC_{\mathrm{mass}})$ and C_1, because they were constrained to the static limit

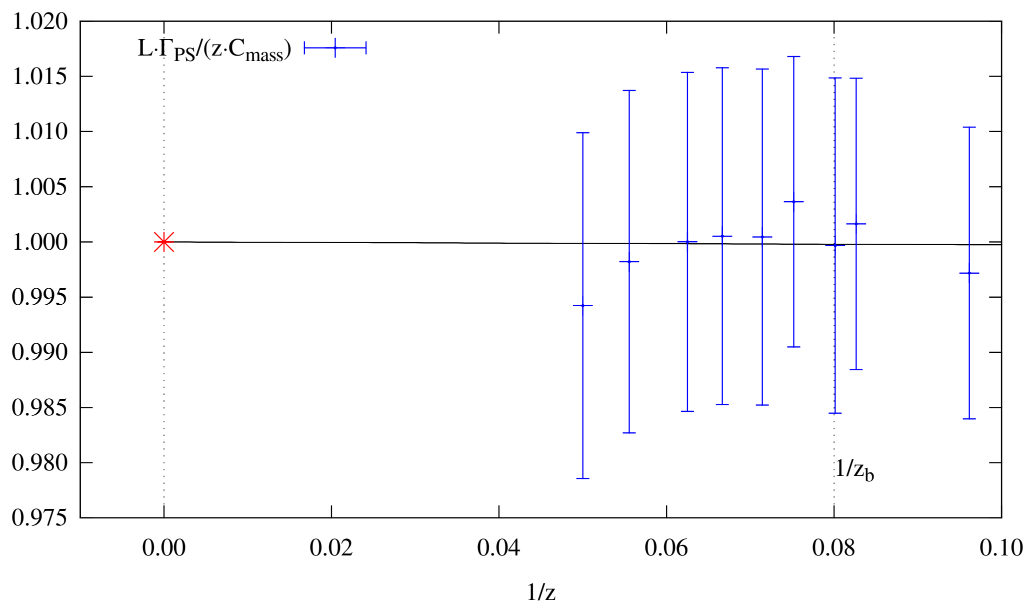

Figure 8.1.: The ratio $L\Gamma_{\mathrm{PS}}/(zC_{\mathrm{mass}})$ as a function of $1/z$ and a linear fit to the data, which is constrained to 1.0 at $1/z \to 0$ and results in a nearly horizontal line

coefficient $c = -3 \pm 2$ of its quadratic fit. But it seems that this is rather because the data points at finite z are systematically too high compared to X_{RGI}. (If the limit value X_{RGI} is not included in the fit, the quadratic term is only 1 ± 6.)

For C_1 (figure 8.3) and Γ_{PS}, the expected large-mass behavior is in good agreement with their values at finite z. As mentioned above, I do not know what behavior Θ_{A} and Θ_1 are expected to show, but the plots in figures 8.4 and 8.5 suggest that they linearly converge for $1/z \to 0$. The limits inferred from the linear fit are 1.465(12) for Θ_{A} and 2.531(17) for Θ_1.

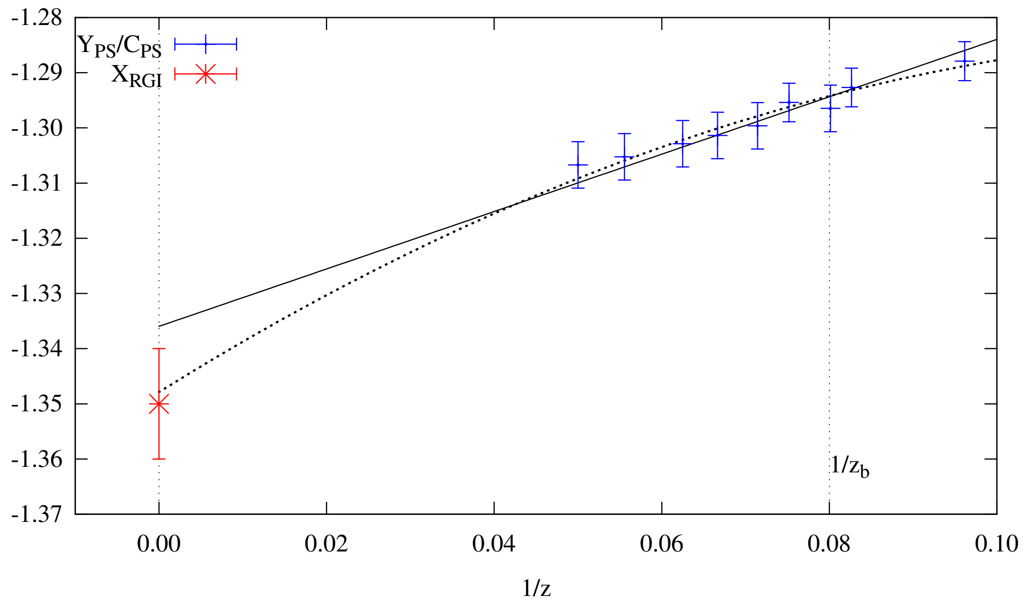

Figure 8.2.: $Y_{\mathrm{PS}}/C_{\mathrm{PS}}$ plotted against the inverse mass $1/z$, the linear (solid) and the quadratic (dotted) fit are shown, both include the static limit X_{RGI}, they are not constrained to it, but take into account its finite error

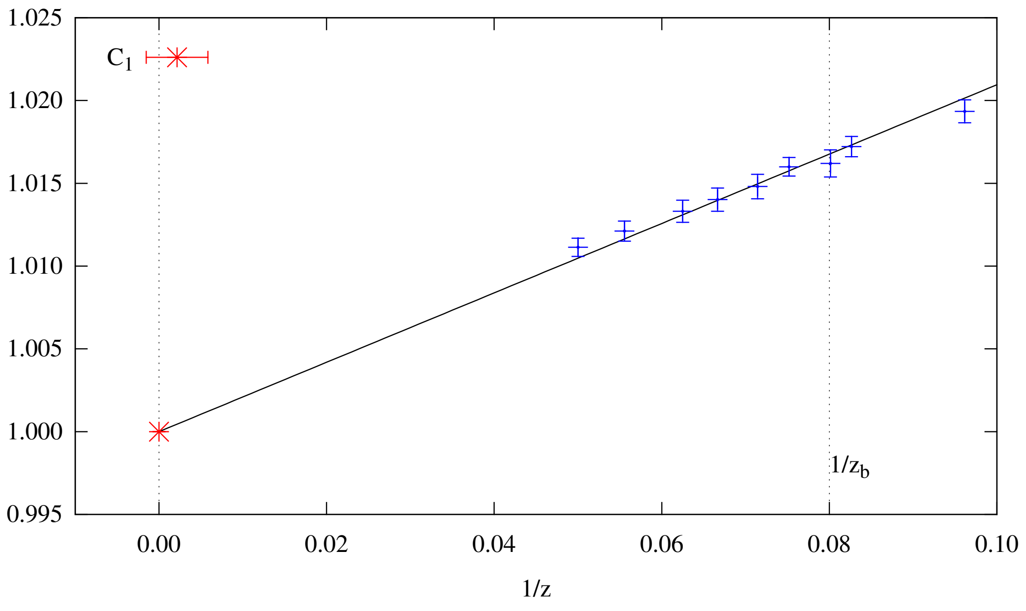

Figure 8.3.: The observable $C_1 = f_1/k_1$ as a function of the inverse mass $1/z$, and a linear fit to the data, which has been constrained to run through 1.0 at $1/z \to 0$

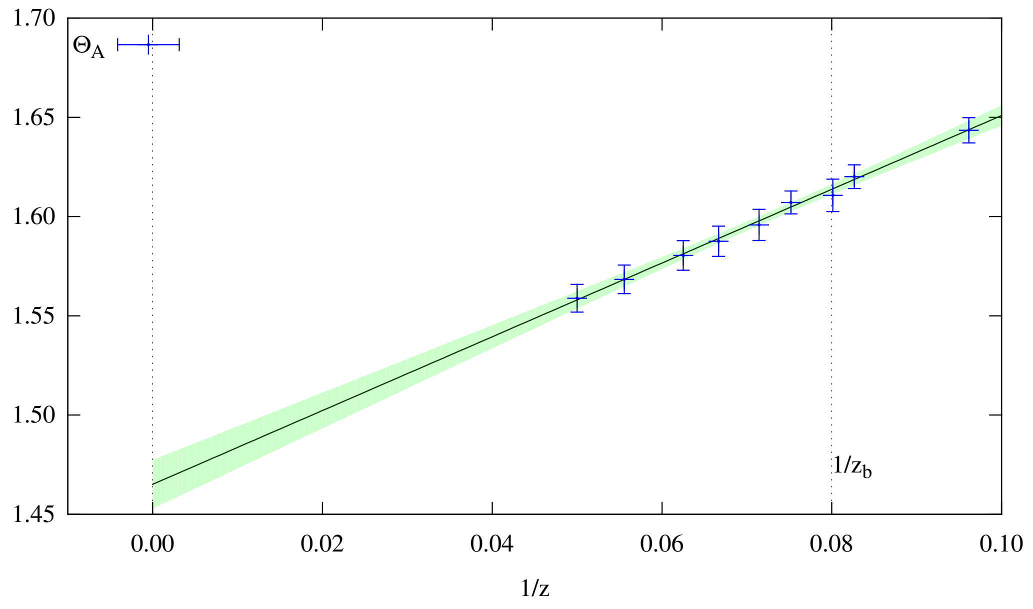

Figure 8.4.: $\Theta_A = f_A(L/2, \theta_1)/f_A(L/2, \theta_2)$ plotted against $1/z$, a linear fit to the data is shown, its uncertainty is depicted as light-blue area

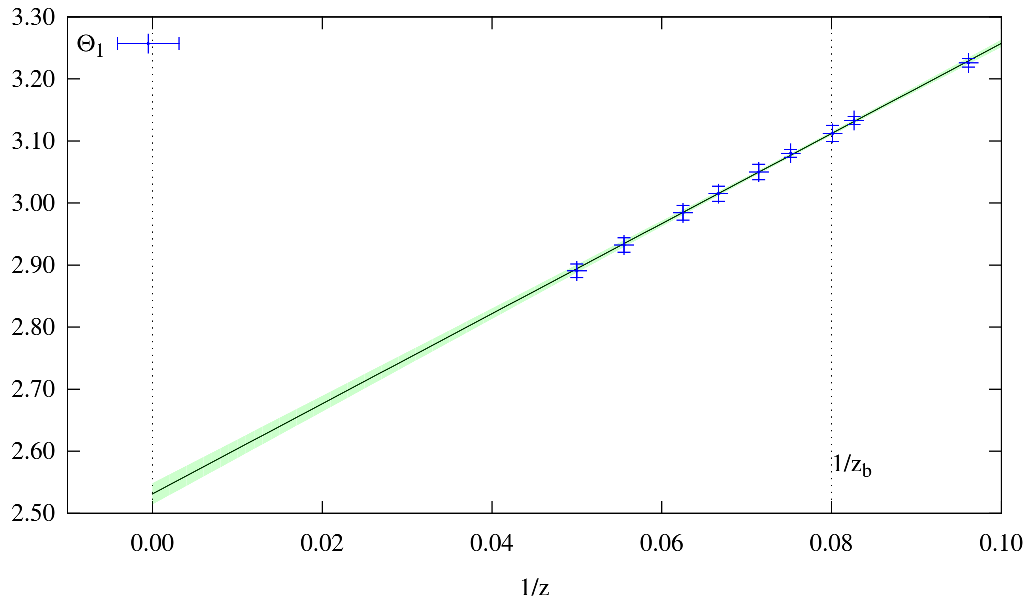

Figure 8.5.: Plot of the data for $\Theta_1 = f_1(\theta_1)k_1(\theta_1)^3/(f_1(\theta_2)k_1(\theta_2)^3)$ for different $1/z$, a linear fit to the data is shown, its uncertainty is depicted as light-blue area

9. Outlook

Different approaches for the implementation of correlation functions have been exhibited in the previous chapters, specific and efficient routines for each single correlation on one side, and only a few generic routines on the other side. I suppose we will follow the second track, since it is easier to manage and more flexible. The additional time needed by the generic functions is irrelevant compared to the solver. However, their complex usage, in particular for correlations involving derivatives or a sum over vector indices, is a problem which I hope can be solved, when a proper main program is created.

Other possible developments are e.g. the inclusion of a gauge-field update routine and the computation of three-point correlation functions like

$$f_{\mathrm{V}}(x_0) \propto \sum_{\vec{x}\vec{y}\vec{z}\vec{u}\vec{v}k} \left\langle \bar{\zeta}'^{\mathrm{d}}(\vec{u})\gamma_5\zeta'^{\mathrm{u}}(\vec{v}) \cdot V_k^{\mathrm{ub}}(x) \cdot \bar{\zeta}^{\mathrm{b}}(\vec{y})\gamma_5\zeta^{\mathrm{d}}(\vec{z}) \right\rangle \tag{9.1}$$

with the vector current

$$V_k^{\mathrm{ub}}(x) = \bar{\psi}^{\mathrm{u}}(x)\gamma_k\psi^{\mathrm{b}}(x), \tag{9.2}$$

where the indices u, d and b stand for the up, down and beauty quark [Doo11, p. 100]. This correlation function is connected to the semi-leptonic decay process of a B meson into a pion, a lepton and its corresponding anti-neutrino, $\mathrm{B} \to \pi^+ + l^- + \bar{\nu}_l$, which are important in the study of the CKM matrix. In QCD, f_{V} and similar correlations would be computed as the contraction of a boundary-to-bulk propagator from the lower boundary $\bar{S}(x)$, from the upper boundary $\bar{R}(x)$ and a boundary-to-boundary propagator $\bar{S}_{\mathrm{T}}$:

$$f_{\mathrm{V}}(x_0) \propto \sum_{\vec{x}k} \left\langle \mathrm{Tr}\, \left\{ \bar{S}_{\mathrm{T}}^{\mathrm{d}\dagger}\gamma_5\bar{R}^{\mathrm{u}}(x)^\dagger\gamma_5\gamma_k\bar{S}^{\mathrm{b}}(x) \right\} \right\rangle_{\mathrm{G}}. \tag{9.3}$$

In fact, this form is very similar to the two-point boundary-to-bulk correlations that were considered in section 2.3.8. The only difference for the computation is an additional matrix which acts not only in Dirac but also in color space, the boundary-to-bulk propagator $\bar{S}_{\mathrm{T}}$. So, their implementation does probably not pose a severe problem.

In principle, the SFCF program can also be applied in larger volumes than those that are needed for the matching, e.g. for the study of the hadron spectrum, the non-perturbative renormalization of the coupling or improvement coefficients. Since it is not tied to an update routine and works on externally created gauge-field configurations, it may also be used with configurations from simulations with $N_{\mathrm{f}} = 4$, i.e. including the strange and charm quark as dynamical flavors.

Although, the SFCF program is not yet in a state where it can be used “out of the box”, without making some adaptions to the program code, it basically works, and as section 8 has shown, it can produce practical results. I hope that it will be able to serve as a successor for the APE program and that it will be flexible and adaptable enough for future applications.

10. Acknowledgements

I want to thank my advisor Jochen Heitger for his always having time and his friendly attitude towards his students, which he did not loose even when some meetings took longer than expected. I am also indebted to Hubert Simma and Michele Della Morte, who gave me the possibility to participate in the development of the SFCF program. I thank Nora Schmidt, who proofread my thesis, and my colleagues at the Institut für Theoretische Physik in Münster for the kind atmosphere, especially Florian König and Michael Topp, who provided me with the necessary distraction after hours of sitting in front of the computer, and Eva Baresel, who provided me with cookies when I was writing this thesis late at night.

Most of all, I say thanks to my parents and my family. I really do not know what I would do without their encouragement and support. *Danke!*

A. Notations

A.1. Chiral Representation of Euclidean Gamma Matrices

In this document the chiral representation of gamma matrices in Euclidean space is used as found in [Lü97b, p. 22]. In this representation the gamma matrices assume the form

$$\gamma_i = \begin{pmatrix} 0 & e_\mu \\ e_\mu^\dagger & 0 \end{pmatrix} \tag{A.1}$$

with 2×2 matrices e_μ that are defined as

$$e_0 = -1 \qquad\qquad e_k = -i\sigma_k \quad \text{for } k = 1, 2, 3 \tag{A.2}$$

where σ_k are the Pauli matrices.

The fifth gamma matrix is

$$\gamma_5 = \gamma_0 \cdot \gamma_1 \cdot \gamma_2 \cdot \gamma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{A.3}$$

The matrices $\sigma_{\mu\nu}$ (not to be confused with the Pauli matrices σ_k) are

$$\sigma_{\mu\nu} = \frac{i}{2} \cdot [\gamma_\mu, \gamma_\nu]\,. \tag{A.4}$$

In the above representation this evaluates to

$$\sigma_{0k} = \begin{pmatrix} \sigma_k & 0 \\ 0 & -\sigma_k \end{pmatrix} \qquad\qquad \sigma_{jk} = -\varepsilon_{jkl} \begin{pmatrix} \sigma_l & 0 \\ 0 & \sigma_l \end{pmatrix}. \tag{A.5}$$

A.2. Lattice Derivatives

The forward covariant lattice derivative acting on a quark field is defined as

$$\nabla_\mu \psi(x) = \lambda_\mu U_\mu(x) \psi(x + \hat{\mu}) - \psi(x), \tag{A.6}$$

likewise, the backward derivative is:

$$\nabla_\mu^* \psi(x) = \psi(x) - \lambda_\mu^* U_\mu(x - \hat{\mu})^\dagger \psi(x - \hat{\mu}) \tag{A.7}$$

and the symmetric derivative acts as:

$$\widetilde{\nabla}_\mu \psi(x) = \frac{1}{2}\Big(\lambda_\mu U_\mu(x)\psi(x+\hat{\mu}) - \lambda_\mu^* U_\mu(x-\hat{\mu})^\dagger \psi(x-\hat{\mu})\Big) \tag{A.8}$$

$$= \frac{1}{2}\left(\nabla_\mu + \nabla_\mu^*\right)\psi(x). \tag{A.9}$$

The additional phases λ_μ that are included in the derivatives can be expressed via the θ angles:

$$\lambda_0 = 1 \qquad \text{and} \qquad \lambda_j = \mathrm{e}^{i\theta_j/L_j} \text{ for } j = 1,2,3. \tag{A.10}$$

The symbol $\widetilde{\partial}_\mu$ is used for the symmetric non-covariant derivative on a lattice:

$$\widetilde{\partial}_\mu f(x) = \frac{1}{2}\left(f(x+\hat{\mu}) - f(x-\hat{\mu})\right). \tag{A.11}$$

B. Graßmann Numbers

This overview of Graßmann numbers is based on the chapters 5.1.3–5.1.6 of [Gat10].

We consider a set of N Graßmann numbers $\eta_1, \dots, \eta_N$. Graßmann numbers are objects which can be added and multiplied. In contrast to the usual real or complex numbers, however, they do not obey the commutativity law of multiplication, instead they anti-commute:

$$\eta_i \eta_j = -\eta_j \eta_i \qquad \Longrightarrow \qquad \eta_i^2 = 0. \tag{B.1}$$

Note that $\eta_i \eta_j$ is not a Graßmann number itself, for it commutes with all $\eta_1, \dots, \eta_N$. We can also add products of several Graßmann numbers and thus get an element of the Graßmann *algebra*. If we use the anti-commutation property, we can write a general element A of the Graßmann algebra as

$$A = a + \sum_i a_i \eta_i + \sum_{ij} a_{ij} \eta_i \eta_j + \sum_{ijk} a_{ijk} \eta_i \eta_j \eta_k + \dots + a_{12\dots N} \eta_1 \eta_2 \dots \eta_N. \tag{B.2}$$

Here, a, a_i etc. are usual complex numbers. So, the Graßmann algebra comprises the complex numbers (first term), the Graßmann *numbers* (second term) and all combinations of products of them. However, we can terminate the series at the product $\eta_1 \eta_2 \dots \eta_N$, because any higher product would have to involve at least one Graßmann number twice, so it would be zero. This is also true for a function of a Graßmann variable, e.g. $\exp(\eta_1) = 1 + \eta_1$ (the next term $\eta_1^2/2$ is already zero).

The derivative with respect to a Graßmann number is defined as

$$\frac{\partial}{\partial \eta_i} 1 = 0 \qquad \frac{\partial}{\partial \eta_i} \eta_j = \delta_{ij}. \tag{B.3}$$

Derivatives of products of Graßmann numbers can be evaluated in a straight-forward way, if one keeps in mind that also the derivatives must anti-commute with Graßmann numbers as well as with other derivatives to avoid inconsistencies, e.g.

$$\frac{\partial}{\partial \eta_1} \frac{\partial}{\partial \eta_2} (\eta_1 \eta_3 \eta_2) = -\frac{\partial}{\partial \eta_2} \frac{\partial}{\partial \eta_1} (\eta_1 \eta_3 \eta_2) = -\frac{\partial}{\partial \eta_2} (\eta_3 \eta_2) = +\eta_3 \frac{\partial}{\partial \eta_2} \eta_2 = \eta_3. \tag{B.4}$$

Furthermore, an integration over a Graßmann number is defined. The integral is demanded to be a complex linear functional and the integral of a derivative shall be zero:

$$\int \mathrm{d}\eta_i \frac{\partial}{\partial \eta_i} A = 0. \tag{B.5}$$

This corresponds to the requirement that a usual function on an interval $[a, b] \in \mathbb{R}$ vanish at the boundaries (or that it assume the same value), so that $\int_a^b \mathrm{d}x \partial f/\partial x = f(b) - f(a) = 0$.

This entails that

$$\int \mathrm{d}\eta_i 1 = 0, \tag{B.6}$$

because $1 = \partial\eta_i/\partial\eta_i$. The integral of η_i, however, does not vanish, since η_i can not be written as a derivative with respect to itself. It is normalized to

$$\int \mathrm{d}\eta_i \eta_i = 1. \tag{B.7}$$

And as the derivatives, two integrals over Graßmann numbers $\int \eta_i$ and $\int \eta_j$ anticommute. This has the consequence that integrals and derivatives obey the same rules and yield the same results: $\partial A/\partial\eta_i = \int \mathrm{d}\eta_i A$.

We will now look at Gaußian integrals with Graßmann numbers, since the path integral over fermionic fields is of this very type. We introduce a second set of Graßmann numbers $\bar\eta_1, \ldots, \bar\eta_N$. The bar over their symbols has no meaning for the operations defined above. They obey the same rules as $\eta_1, \ldots, \eta_N$, and we could have labeled them $\eta_{N+1}, \ldots, \eta_{2N}$, too.

A Gaußian integral over these numbers can be computed by expanding the exponential in products of Graßmann numbers as in (B.2) and applying the integration rules. One arrives at the Matthews–Salam formula:

$$Z_\mathrm{F} = \int \mathrm{d}\bar\eta_N \mathrm{d}\eta_N \ldots \mathrm{d}\bar\eta_1 \mathrm{d}\eta_1 \, \exp\left(-\sum_{ij} \bar\eta_i M_{ij} \eta_j\right) = \det(M), \tag{B.8}$$

where M is a complex $N \times N$-matrix, and in analogy to the path integral, we have named this integral Z_F.

If we want to compute expectation values of fermionic fields, we also need integrals with an insertion of a product of several Graßmann numbers. For the derivation, I refer to [Gat10, p. 109]. The result is

$$\begin{aligned}
&\langle \eta_{i_1} \bar\eta_{j_1} \ldots \eta_{i_N} \bar\eta_{j_N} \rangle_\mathrm{F} \\
&= \frac{1}{Z_\mathrm{F}} \int \mathrm{d}\bar\eta_N \mathrm{d}\eta_N \ldots \mathrm{d}\bar\eta_1 \mathrm{d}\eta_1 \, \eta_{i_1} \bar\eta_{j_1} \ldots \eta_{i_N} \bar\eta_{j_N} \, \exp\left(-\sum_{kl} \bar\eta_k M_{kl} \eta_l\right) \\
&= \sum_{P(1,2,\ldots,N)} \mathrm{sign}(P) \cdot \left(M^{-1}\right)_{i_1 j_{P(1)}} \left(M^{-1}\right)_{i_2 j_{P(2)}} \ldots \left(M^{-1}\right)_{i_N j_{P(N)}},
\end{aligned} \tag{B.9}$$

where the sum runs over all permutations of the numbers $1, 2, \ldots, N$ and $\mathrm{sign}(P)$ is the sign of the permutation. This formula is a version to state *Wick's theorem.*

Bibliography

[Aok03] S. Aoki, et al., *Light hadron spectrum and quark masses from quenched lattice QCD*, Phys.Rev., volume D67: page 034 503, 2003, doi:10.1103/PhysRevD.67.034503, `hep-lat/0206009`

[Blo10] Benoit Blossier, Michele della Morte, Nicolas Garron, Rainer Sommer, *HQET at order 1/m: I. Non-perturbative parameters in the quenched approximation*, JHEP, volume 06: page 002, 2010, doi:10.1007/JHEP06(2010)002, `1001.4783`

[Bod00] Achim Bode, Peter Weisz, Ulli Wolff, *Two loop computation of the Schrödinger functional in lattice QCD*, Nucl. Phys., volume B576: pages 517–539, 2000, doi:10.1016/S0550-3213(00)00187-5,10.1016/S0550-3213(00)00187-5, `hep-lat/9911018`

[Bod03] F. Bodin, Philippe Boucaud, J. Micheli, et al., *The apeNEXT project*, eConf, volume C0303241: page THIT005, 2003, `hep-lat/0306018`

[Bur07] Cliff Burgess, Guy Moore, *The Standard Model – A Primer*, Cambridge University Press, 2007

[Cap99] Stefano Capitani, Martin Lüscher, Rainer Sommer, Hartmut Wittig, *Non-perturbative quark mass renormalization in quenched lattice QCD*, Nucl. Phys., volume B544: pages 669–698, 1999, doi:10.1016/S0550-3213(98)00857-8, `hep-lat/9810063`

[Col84] John Collins, *Renormalization*, Cambridge University Press, 1984

[Cot07] W. N. Cottingham, D. A. Greenwood, *An introduction to the Standard Model of Particle Physics*, Cambridge University Press, 2007

[Das93] Ashok Das, *Field Theory: A Path Integral Approach*, World Scientific, 1993

[DM] Michele Della Morte, Samantha Dooling, Jochen Heitger, Hubert Simma, *Matching of the axial/vector currents at 1/m*, under construction

[DM05] Michele Della Morte, Andrea Shindler, Rainer Sommer, *On lattice actions for static quarks*, JHEP, volume 08: page 051, 2005, doi:10.1088/1126-6708/2005/08/051, `hep-lat/0506008`

[DM07] Michele Della Morte, Nicolas Garron, Mauro Papinutto, Rainer Sommer, *Heavy quark effective theory computation of the mass of the bottom quark*, JHEP, volume 0701: page 007, 2007, doi:10.1088/1126-6708/2007/01/007, `hep-ph/0609294`

[Doo11] Samantha Dooling, *Feasibility study of HQET to QCD matching of heavy-light axial and vector currents*, Master's thesis, Institut für Theoretische Physik, Westfälische Wilhelms-Universität Münster, 2011

[Eic90] Estia Eichten, Brian Russell Hill, *An Effective Field Theory for the Calculation of Matrix Elements Involving Heavy Quarks*, Phys.Lett., volume B234: page 511, 1990, doi:10.1016/0370-2693(90)92049-O

[Fey65] Richard P. Feynman, *Nobel Lecture*, 1965, URL `http://www.nobelprize.org/nobel_prizes/physics/laureates/1965/feynman-lecture.html`, accessed on Monday, 2012-02-20 14:15 CET

[Fri09] Patrick Fritzsch, *B-meson properties from non-perturbative matching of HQET to finite-volume two-flavour QCD*, Ph.D. thesis, Institut für Theoretische Physik, Westfälische Wilhelms-Universität Münster, 2009

[Gat10] Christof Gattringer, Christian B. Lang, *Quantum Chromodynamics on the Lattice*, Springer, 2010

[Gra05] Richard L. Graham, Timothy S. Woodall, Jeffrey M. Squyres, *Open MPI: A Flexible High Performance MPI*, in *In The 6th Annual International Conference on Parallel Processing and Applied Mathematics*, 2005

[Gre04] Anthony M. Green (editor), *Hadronic Physics from Lattice QCD*, World Scientific, 2004

[Gro73] David J. Gross, Frank Wilczek, *Ultraviolet Behavior of Non-Abelian Gauge Theories*, Phys. Rev. Lett., volume 30: pages 1343–1346, Jun 1973, doi:10.1103/PhysRevLett.30.1343, URL `http://link.aps.org/doi/10.1103/PhysRevLett.30.1343`

[Gua99] Marco Guagnelli, Jochen Heitger, Rainer Sommer, Hartmut Wittig, *Hadron masses and matrix elements from the QCD Schrödinger functional*, Nucl. Phys., volume B560: pages 465–481, 1999, doi:10.1016/S0550-3213(99)00466-6, `hep-lat/9903040`

[Has01] Anna Hasenfratz, Francesco Knechtli, *Flavor symmetry and the static potential with hypercubic blocking*, Phys. Rev., volume D64: page 034 504, 2001, doi:10.1103/PhysRevD.64.034504, `hep-lat/0103029`

[Hei03] Jochen Heitger, Martin Kurth, Rainer Sommer, *Non-perturbative renormalization of the static axial current in quenched QCD*, Nucl. Phys., volume B669: pages 173–206, 2003, doi:10.1016/S0550-3213(03)00552-2, `hep-lat/0302019`

[Hei04a] Jochen Heitger, Andreas Juttner, Rainer Sommer, Jan Wennekers, *Non-perturbative tests of heavy quark effective theory*, JHEP, volume 11: page 048, 2004, doi:10.1088/1126-6708/2004/11/048, `hep-ph/0407227`

[Hei04b] Jochen Heitger, Jan Wennekers, *Effective heavy light meson energies in small volume quenched QCD*, JHEP, volume 0402: page 064, 2004, doi: 10.1088/1126-6708/2004/02/064, `hep-lat/0312016`

[Hes52] Magnus R. Hestenes, Eduard Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, volume 49, 1952

[Hof05] Roland Hoffmann, *Chiral properties of dynamical Wilson fermions*, Ph.D. thesis, Humboldt-Universitat zu Berlin, 2005

[Int] *Intel(R) 64 and IA-32 Architectures Software Developer's Manual*, URL `http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html`, accessed on Monday, 2012-02-20 17:30 CET

[Jan98] Karl Jansen, Rainer Sommer, *O(alpha) improvement of lattice QCD with two flavors of Wilson quarks*, Nucl. Phys., volume B530: pages 185–203, 1998, doi:10.1016/S0550-3213(98)00396-4,10.1016/S0550-3213(98)00396-4, `hep-lat/9803017`

[Ker88] Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language*, Prentice Hall, 2 edition, 1988

[Kö91] J. G. Körner, George Thompson, *The Heavy mass limit in field theory and the heavy quark effective theory*, Phys. Lett., volume B264: pages 185–192, 1991, doi:10.1016/0370-2693(91)90725-6

[Lus96] Ewing Lusk, Nathan Doss, Anthony Skjellum, *A high-performance, portable implementation of the MPI message passing interface standard*, Parallel Computing, volume 22: pages 789–828, 1996

[Lü92] Martin Lüscher, Rajamani Narayanan, Peter Weisz, Ulli Wolff, *The Schrödinger functional – a renormalizable probe for non-abelian gauge theories*, Nucl. Phys. B, volume 384: pages 168–228, 1992

[Lü94] Martin Lüscher, Rainer Sommer, Peter Weisz, Ulli Wolff, *A precise determination of the running coupling in the SU(3) Yang-Mills theory*, Nucl. Phys. B, volume 413: pages 481–502, 1994

[Lü96a] M. Lüscher, P. Weisz, *O(a) improvement of the axial current in lattice QCD to one loop order of perturbation theory*, Nucl. Phys., volume B479: pages 429–458, 1996, doi:10.1016/0550-3213(96)00448-8, `hep-lat/9606016`

[Lü96b] Martin Lüscher, Stefan Sint, Rainer Sommer, Peter Weisz, *Chiral symmetry and O(a) improvement in lattice QCD*, Nucl. Phys. B, volume 478: pages 365–397, 1996

[Lü97a] Martin Lüscher, Stefan Sint, Rainer Sommer, Peter Weisz, Ulli Wolff, *Non-perturbative O(a) improvement of lattice QCD*, Nucl. Phys., volume B491: pages 323–343, 1997, doi:10.1016/S0550-3213(97)00080-1, `hep-lat/9609035`

[Lü97b] Martin Lüscher, Peter Weisz, Ulli Wolff, *TAO programs for the Dirac operator in O(a) improved lattice QCD*, May 1997, internal notes

[Lü03] Martin Lüscher, *Lattice QCD: From quark confinement to asymptotic freedom*, Annales Henri Poincare, volume 4: pages S197–S210, 2003, `hep-ph/0211220`

[Lü05] Martin Lüscher, *Schwarz-preconditioned HMC algorithm for two-flavour lattice QCD*, Comput. Phys. Commun., volume 165: pages 199–220, 2005, doi:10.1016/j.cpc.2004.10.004, `hep-lat/0409106`

[Mag05] Michele Maggiore, *A Modern Introduction to Quantum Field Theory*, Oxford University Press, 2005

[McG] Roland McGrath, Mark Brown, Paul Eggert, et al., *GNU C Library 2.14*, URL `http://www.gnu.org/software/libc/index.html`, accessed on Monday, 2012-02-20 15:15 CET

[Moh03] P. J. Mohr, B. N. Taylor, *Searchable Bibliography on the Constants (version 3.0)*, 2003, URL `http://physics.nist.gov/constantsbib`, accessed on Monday, 20-Feb-2012 14:00 CET

[Mon94] Istvan Montvay, Gernot Münster, *Quantum Fields on a Lattice*, Cambridge University Press, 1994

[Mor05] M. Della Morte, P. Fritzsch, N. Garron, et al., *HQET at subleading order: implementation notes*, Internal notes, January 2005

[Nak10] K Nakamura, Particle Data Group, *Review of Particle Physics*, Journal of Physics G: Nuclear and Particle Physics, volume 37(7A): page 075 021, 2010, URL `http://stacks.iop.org/0954-3899/37/i=7A/a=075021`, all data available on http://pdg.lbl.gov

[Ope] *Open MPI v1.4.5 documentation*, URL `http://www.open-mpi.org/doc/v1.4/`, accessed on Monday, 2012-02-20 15:19 CET

[Pol73] H. David Politzer, *Reliable Perturbative Results for Strong Interactions?*, Phys. Rev. Lett., volume 30: pages 1346–1349, 1973, doi:10.1103/PhysRevLett.30.1346

[Pos] *1003.1-2008 – IEEE Standard for Information Technology – Portable Operating System Interface (POSIX(R))*, URL `http://standards.ieee.org/findstds/standard/1003.1-2008.html`, accessed on Monday, 2012-02-20 15:05 CET

[Pre92] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992

[Ram89] Pierre Ramond, *Field Theory: A Modern Primer*, Addison-Wesley, 1989

[Rot05] Heinz J. Rothe, *Lattice Gauge Theories – An introduction*, World Scientific, 2005

[Sin94] Stefan Sint, *On the Schrödinger functional in QCD*, Nucl. Phys., volume B421: pages 135–158, 1994, doi:10.1016/0550-3213(94)90228-3, `hep-lat/9312079`

[Sin96] Stefan Sint, Rainer Sommer, *The running coupling from the QCD Schrödinger functional: a one-loop analysis*, Nucl. Phys. B, volume 465: pages 71–98, 1996, hep-lat/9508012

[Som95] Rainer Sommer, *Fermionic Correlation Functions in the SF*, September 1995, notes concerning an older version of the APE code

[Som10] Rainer Sommer, *Introduction to Non-perturbative Heavy Quark Effective Theory*, 2010, `1008.0710`

[Sto80] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, Springer, 1980

[Sym81] K. Symanzik, *Schrödinger Representation and Casimir Effect in Renormalizable Quantum Field Theory*, Nucl. Phys., volume B190: page 1, 1981, doi:10.1016/0550-3213(81)90482-X

[Top12] Michael Topp, *Nicht-perturbative Bestimmung von HQET-Parametern und der Quark-Massen-Abhangigkeit des Spin-Splittings im B-Meson-System*, Master's thesis, Institut für Theoretische Physik, Westfälische Wilhelms-Universität Münster, 2012

[Wil74] Kenneth G. Wilson, *Confinement of Quarks*, Phys. Rev., volume D10: pages 2445–2459, 1974, doi:10.1103/PhysRevD.10.2445

[Wil05] Kenneth G. Wilson, *The Origins of lattice gauge theory*, Nucl. Phys. Proc. Suppl., volume 140: pages 3–19, 2005, doi:10.1016/j.nuclphysbps.2004.11.271, `hep-lat/0412043`

Statement of Authorship

I confirm that this thesis is my own work and that I have only used the sources and means that I named therein, in particular, I have marked passages that were taken from the works of other authors as citations and named their sources.

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und nur die von mir angegebenen Quellen und Hilfsmittel genutzt habe, insbesondere dass ich Textstellen, die wörtlich oder sinngemäß Werken anderer Autoren entnommen wurden, unter Angabe der Quelle als Zitat kenntlich gemacht habe.

Christian Wittemeier