BERGISCHE
UNIVERSITÄT
WUPPERTAL

Adaptive Domain Decomposition Multigrid for Lattice QCD

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

an der Fakultät für Mathematik und Naturwissenschaften der
Bergischen Universität Wuppertal eingereichte

Dissertation

von

Matthias Rottmann

aus Wuppertal

Die Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20160304-125225-9
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20160304-125225-9]

Acknowledgments

I would like to thank all the people who enabled me to write this thesis and facilitated the success of this work.

First, I thank my advisor Prof. Andreas Frommer who guided me the way to multigrid solvers for lattice QCD. With his lectures he raised my interest for sparse matrices and grid structures already during my diploma studies. He always had useful advices during my graduate studies, he enabled me to present results on many conferences around the globe and to work in a very pleasant environment. Furthermore, I thank Karsten Kahl who supervised me and helped me several times with his expertise in multigrid methods and their implementation. I also thank Prof. Matthias Bolten. After my final oral diploma exam in his course, he offered me to join the work group.

I thank Björn Leder for many fruitful discussions and intensive collaborative work. Many valuable advices and input I got from Holger Arndt, Stefan Krieg, Kalman Szabo, Simon Heybrock, Wolfgang Söldner and many others in- and outside of the work group and the whole transregional collaborative research centre 55 (SFB/TRR55) "Hadron Physics from Lattice QCD".

Last but not least I thank my family and friends for mental support.

I thank BMW-c, QCDSF and CLS for providing several configurations for the numerical tests. The numerical results were obtained on Juropa at Jülich Supercomputing Centre (JSC) through NIC grant HWU12 if not stated otherwise.

My position was partially funded by Deutsche Forschungsgemeinschaft (DFG) within the SFB/TRR55.

Wuppertal, Matthias Rottmann

Contents

Chapter 1

Introduction

The main results of this thesis have already been published in the papers [24,47–49]. The physical background and the motivation for our research has been described in these papers as well, and this introduction is largely based on the introductory sections of [24, 49].

Lattice QCD simulations are among the world's most demanding computational problems, and a significant part of today's supercomputer resources is spent in these simulations [12, 58]. The focus of this thesis lies in the development and implementation of an adaptive aggregation-based multigrid method for the Wilson-Dirac operator from lattice QCD. We analyze the behavior of the method and its production code quality implementation with up-to-date physical data and show thorough comparisons with other existing successful approaches in this field. In addition we show results from a non-standard application where we apply our multigrid method as a preconditioner to the overlap Dirac operator, another discretization from lattice QCD.

The computational challenge in lattice QCD computations consists of repeatedly solving very large sparse linear systems

$$Dz = b\,,$$

where $D = D(U, m)$ is a discretization, typically the Wilson discretization, of the Dirac operator on a four-dimensional space-time lattice. The Wilson-Dirac operator depends on a gauge field U and a mass parameter m. In recent computations lattices with up to 144×64^3 lattice points have been used, involving the solution of linear systems with 452,984,832 unknowns [2,6,8,39,44]. Usually these linear systems are solved by standard Krylov subspace methods. Their iteration count increases tremendously when approaching the physically relevant parameter values (i.e., physical mass constants and lattice spacing $a \to 0$), a phenomenon referred to as "critical slowing down" in the physics literature. Thus it is of utmost importance to develop preconditioners for said methods which overcome these

scaling issues. The most commonly used preconditioners in lattice QCD computations nowadays are odd-even preconditioning [33, 74], deflation techniques [79] and domain decomposition approaches [50, 78]. While these approaches yield significant speed-ups over the unpreconditioned versions, their scaling behavior is unchanged and critical slowing down still occurs.

Multigrid methods have been considered in the lattice QCD community as well, motivated by their potential (e.g., for elliptic PDEs) of convergence independent of the lattice spacing. However, due to the random nature of the gauge fields involved, the treatment of the lattice Dirac equation by *geometric* multigrid methods, i.e., methods based solely on the underlying PDE, has been elusive for the last twenty years [11,27,68,113]. With the advent of *adaptive algebraic* multigrid methods, effective preconditioners for QCD calculations could be constructed in recent years. The pioneering work from [5, 23, 92] showed very promising results. There, an adaptive non-smoothed aggregation approach based on [25] has been proposed for the solution of the Wilson-Dirac system. An implementation is available within the QOPQDP library [91].

Within the physics community, another hierarchical technique, a domain decomposition type solver named *inexact deflation* developed in [79], is widely used. A well-optimized code for this solver is publicly available [75,76]. Inexact deflation can be regarded as an adaptive method as well. It performs a setup phase which allows the construction of a smaller system, the *little Dirac* operator, which is then used as part of an efficient preconditioner. Although there is an intimate connection with the aggregation-based multigrid approach from [25], inexact deflation seems to have been developed completely independently. As a consequence, the inexact deflation method does not resemble a typical multigrid method in the way its ingredients are arranged. In particular, it requires the little Dirac system to be solved to high accuracy in each iteration.

In this thesis we present a multigrid method that combines aspects from [79], namely a domain decomposition smoother, and from non-smoothed γ_5-respecting aggregation as in [5, 92]. Our approach elaborates on the multigrid methods from [5, 92] in that we use a domain decomposition method as the smoother instead of the previously used Krylov subspace smoother. This allows for a natural and efficient parallelization, also on hybrid architectures. Moreover, we substantially improve the adaptive setup from [5, 92] and [79] in the sense that less time is required to compute the operator hierarchy needed for an efficient multigrid method. Our approach can also be regarded as turning the domain decomposition technique from [79] into a true multigrid method. The "little Dirac" system now needs to be solved only to low accuracy. This allows, in particular, to apply the method recursively. One simply uses another two-level ansatz for the solution of the linear system on the coarse grid and applies this construction principle recursively, possibly up to a level where a direct, "exact" solution of the coarse-level system is feasible. With the inexact deflation approach, an efficient recursive application is not possible.

Furthermore, we apply our multigrid method to the overlap discretization from lattice QCD in terms of an auxiliary space preconditioning technique [85], i.e., using the Wilson-Dirac operator as a preconditioner to the overlap operator. The overlap operator is particularly attractive from a theoretical point of view since it respects an important physical property, chiral symmetry, which is violated by the Wilson discretization. However, it has the disadvantage that its computational cost can be two orders of magnitude larger than when using the Wilson discretization. We demonstrate that the technique we develop is able to reduce the computational cost for solving systems with the overlap operator substantially, reaching speed-ups of a factor of 10 or more in realistic settings. The preconditioning technique thus contributes to making the overlap operator more tractable in lattice QCD calculations.

This thesis is organized as follows. In Chapter 2 we review the Wilson discretization and its clover improved variant as a discretization of the continuum Dirac operator, and we discuss properties of the Wilson-Dirac operator. Thereafter we introduce smearing techniques and explore the connection between smearing and normality. A variety of Schwarz domain decomposition methods are introduced in Chapter 3. We also give implementation details for the latter and for the matrix-vector multiplication with the Wilson-Dirac operator followed by numerical experiments. Chapter 4 gives an introduction into aggregation-based algebraic multigrid for lattice QCD. The construction of transfer and coarse grid operators are discussed from the theoretical and implementational point of view.

In Chapter 5 we introduce the inexact deflation method from [79] and its setup procedure implemented in [75, 76] and discuss the properties of its ingredients. We also point out the differences to algebraic multigrid and why a recursive extension of the inexact deflation approach is not advisable. Chapter 6 introduces the algebraic multigrid method with Krylov subspace smoothing proposed in [4, 5, 23, 92] and implemented in [91], also including its setup procedure. Thereafter, in Chapter 7 we introduce our two-level approach and its setup procedure, followed by a thorough experimental analysis and comparisons with the methods from Chapters 5 and 6. The recursive extension of the two-level approach is introduced in Chapter 8. We explain the different multilevel cycling strategies and extend the two-level setup to a multilevel setup. Implementation details for the realization of a parallel multilevel implementation are given, followed by a thorough experimental study and several comparisons.

Finally, in Chapter 9 we develop an auxiliary space type preconditioning technique using the Wilson operator to precondition the overlap operator. A rigorous theory is developed for the hypothetical case where the Wilson operator is normal. This theory is then numerically justified for physically relevant cases.

Chapter 2

Quantum Chromodynamics

Quantum Chromodynamics (QCD) is a quantum field theory in four-dimensional space-time. It is the theory of the strong interaction of quarks (particles) and gluons (counterparts/interaction particles). In the continuum theory, this interaction is given by the Dirac operator $\mathcal{D}$ which can be written as

$$\mathcal{D} = \sum_{\mu=0}^{3} \gamma_\mu \otimes (\partial_\mu + A_\mu) \ ,$$

where $\partial_\mu = \partial/\partial x_\mu$ and $A_\mu(x)$ is the gauge field (at a point x in space-time). The anti-hermitian traceless matrices $A_\mu(x)$ are elements of $\mathfrak{su}(3)$, the Lie algebra of the special unitary group SU(3). The γ-matrices $\gamma_0, \gamma_1, \gamma_2, \gamma_3 \in \mathbb{C}^{4\times 4}$ (also known as Dirac matrices) are hermitian and unitary matrices which generate the Clifford algebra $C_{0,4}(\mathbb{R})$, cf. [73].

QCD has a high predictive power, i.e., a small number of free parameters. Predictions that can be deduced from this theory are amongst others the masses of hadrons, composite particles bound by the strong interaction (e.g., nucleon, pion; cf. [38]). The masses of hadrons and many other predictions have to be obtained non-perturbatively, i.e., via numerical simulations requiring the discretization and numerical evaluation of the theory.

In this chapter we introduce the reader to important concepts and the notation necessary to understand the Wilson discretization of the Dirac operator. We also discuss properties of the Wilson discretization with special emphasis on its non-normality and draw the connection to smearing techniques. For a more detailed introduction to QCD and lattice QCD we refer the interested reader to [32,51,82]. This chapter is largely based on the QCD related theory sections in [24, 49].

2.1 Continuum QCD

In this section we rather focus on the mathematical construction of the continuum Dirac operator than on the physical background. In these terms quarks and gluons can be seen as differentiable maps defined on $\mathbb{R}^4$.

Definition 2.1
Let $\mathcal{C} := \{1, 2, 3\}$ *be the set of* color indices, $\mathcal{S} := \{0, 1, 2, 3\}$ *the* spin indices *and*

$$\begin{array}{rccl} \psi & : & \mathbb{R}^4 & \to & \mathbb{C}^{12} \cong \mathbb{C}^{\mathcal{C}\times\mathcal{S}} \\ & & x & \mapsto & (\psi_{10}(x), \psi_{20}(x), \psi_{30}(x), \psi_{11}(x), \ldots, \psi_{33}(x))^T \end{array}$$

a differentiable function. Then ψ *defines a* quark field *or* matter field. *Let* $\mathcal{M} = \{\psi : \psi \text{ matter field}\}$. *The twelve component vector* $\psi(x)$ *is called* spinor. *Furthermore, for* $\mu = 0, 1, 2, 3$

$$\begin{array}{rccl} A_\mu & : & \mathbb{R}^4 & \to & \mathfrak{su}(3) \\ & & x & \mapsto & A_\mu(x)\,, \end{array}$$

the set $\{A_\mu : \mu = 0, 1, 2, 3\}$ *defines a* gauge field, *i.e., a gluonic counterpart of a quark field.*

A component of the spinor $\psi(x)$ is typically denoted by $\psi_{c\sigma}(x)$ where $c \in \mathcal{C}$ determines the color and $\sigma \in \mathcal{S}$ the spin index. The connection between the gauge matrices $A_\mu(x) \in \mathfrak{su}(3)$ and the matter fields $\psi(x) \in \mathbb{C}^{12}$ is established by tensorizing the gauge matrices with certain 4-by-4 matrices.

Definition 2.2
A set of hermitian, unitary matrices $\{\gamma_\mu \in \mathbb{C}^{4\times4} : \mu = 0, 1, 2, 3\}$ *is called a set of generators of the* Clifford algebra $C_{0,4}(\mathbb{R})$, *iff*

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = \begin{cases} 2 \cdot I_4 & \mu = \nu \\ 0 & \mu \neq \nu \end{cases} \quad \text{for } \mu, \nu = 0, 1, 2, 3\,. \tag{2.1}$$

The matrices γ_μ *are called* γ-matrices *or* Dirac-matrices.

The meaning behind the naming convention $C_{0,4}(\mathbb{R})$ as well as details about the role of Clifford algebras in physics can be found in [73].

Unlike the gauge fields A_μ and the matter fields ψ and η, the γ-matrices do not depend on x. The multiplication of a γ-matrix with ψ is defined by $(\gamma_\mu\psi)(x) := (\gamma_\mu \otimes I_3)\psi(x)$. In case operations act unambiguously on the color but differently on the spin degrees of freedom we use the notation ψ_σ to denote those components of the quark field belonging to the fixed spin index σ. For a given point x, $\psi_\sigma(x)$ is thus represented by a three component column vector $\psi_\sigma(x) = (\psi_{1\sigma}(x), \psi_{2\sigma}(x), \psi_{3\sigma}(x))^T$. The value of the gauge field A_μ at point x acts non-trivially on the color and trivially on the spin degrees of freedom in the sense that $(A_\mu\psi)(x) := (I_4 \otimes A_\mu(x))\psi(x)$.

Definition 2.3
Let $\mathcal{M}$ be the space of matter fields. The continuum Dirac operator *is the map*

$$\mathcal{D} \;:\; \mathcal{M} \;\to\; \mathcal{M}$$

defined by

$$\mathcal{D} := \sum_{\mu=0}^{3} \gamma_\mu \otimes (\partial_\mu + A_\mu) \tag{2.2}$$

where $\partial_\mu = \partial/\partial x_\mu$ denotes the partial derivative in direction μ. Evaluating $\mathcal{D}\psi$ at $x \in \mathbb{R}^4$, we have

$$(\mathcal{D}\psi)(x) = \sum_{\mu=0}^{3} \gamma_\mu \left((\partial_\mu + A_\mu)\psi\right)(x)\,.$$

The covariant derivative $\partial_\mu + A_\mu$ is a "minimal coupling extension" of the derivative ∂_μ, ensuring that $((\partial_\mu + A_\mu)\psi)(x)$ transforms in the same way as $\psi(x)$ under local gauge transformations, i.e., a local change of the coordinate system in color space. As part of the covariant derivative the A_μ's can be seen as connecting different (but infinitesimally close) space-time points. The combination of covariant derivatives and the γ-matrices ensures that $\mathcal{D}\psi(x)$ transforms under the space-time transformations of special relativity in the same way as $\psi(x)$. Local gauge invariance and special relativity are fundamental principles of the standard model of elementary particle physics, see, e.g., [95].

2.2 Lattice QCD and the Wilson Discretization

In order to compute predictions in QCD from first principles and non-perturbatively, the theory of QCD has to be discretized and simulated on a computer. The discretization error is then accounted for by extrapolation to the "continuum limit" based on simulations at different lattice spacings. One of the most expensive tasks in these computations is the solution of the discretized Dirac equation for a given right hand side. In this section we give a brief introduction into the principles of this discretization and discuss some properties of the arising linear operators. Since the discretization is typically formulated on an equispaced lattice, this treatment of QCD is also referred to as lattice QCD.

Definition 2.4
Consider a four-dimensional torus $\mathcal{T}$. Then a lattice $\mathcal{L}$ *with* lattice spacing a *is a subset of $\mathcal{T}$ such that for any $x, y \in \mathcal{L}$ there exists $p \in \mathbb{Z}^4$ fulfilling*

$$y = x + a \cdot p, \quad \textit{i.e.,}\ y_\mu = x_\mu + a \cdot p_\mu \textit{ for } \mu = 0, 1, 2, 3\,.$$

For shift operations on the lattice, let $\hat{\mu} \in \mathbb{R}^4$ *denote* shift vectors *defined by*

$$\hat{\mu}_\nu := \begin{cases} a & \mu = \nu \\ 0 & \textit{else.} \end{cases}$$

For a quark field ψ on the lattice, it is sufficient to be defined at each lattice point only, i.e., it is a function

$$\begin{array}{rccc} \psi: & \mathcal{L} & \to & \mathbb{C}^{12} \\ & x & \mapsto & \psi(x) \end{array}$$

which apparently does not require differentiability anymore. As in continuum QCD, the spinor $\psi(x)$ again has color and spin indices $\psi_{c\sigma}$, $c \in \mathcal{C}$, $\sigma \in \mathcal{S}$ (cf. Definition 2.1).

The gauge fields $A_\mu(x)$ connecting infinitesimally close space-time points in continuum QCD have to be replaced by objects that connect points at *finite* distances. To this purpose variables $U_\mu(x)$ are introduced. The translation from A_μ to U_μ can be defined as follows.

Definition 2.5
Given a gauge field A_μ*, the corresponding discretized gauge field* U_μ *at a point* x *is defined by the path ordered integral along the link* $(x, x+\hat{\mu})$

$$U_\mu(x) := e^{-\mathcal{P}\int_x^{x+\hat{\mu}} A_\mu(s)ds} \approx e^{-aA_\mu(x+\frac{1}{2}\hat{\mu})}\,.$$

The discretized gauge field $U = \{U_\mu(x) : x \in \mathcal{L},\ \mu = 0,1,2,3\}$ *is called* (gauge) configuration*.*

In lattice QCD calculations, the initial point is always a discrete gauge configuration U, the translation from A_μ to U_μ is "only" of theoretical interest and never performed in practice. Since $U_\mu(x)$ connects x and $x+\hat{\mu}$, we regard $U_\mu(x)$ as being associated with the *link* between x and $x+\hat{\mu}$. The link between $x+\hat{\mu}$ and x, pointing in the opposite direction, is then given by $U_\mu(x)^{-1}$ (cf. Definition 2.5). The matrices $U_\mu(x)$ satisfy

$$U_\mu(x) \in \mathrm{SU}(3),\ \text{in particular } U_\mu(x)^{-1} = U_\mu(x)^H\,.$$

Figure 2.1 illustrates the naming conventions on the lattice. $U_\mu(x)$ is also called a *gauge link*.

The covariant derivative of the continuum theory can be discretized in many ways. Here we restrict ourselves to the widely used Wilson discretization (cf. [115]).

Definition 2.6
Let A_μ *be a gauge field and* U_μ *the corresponding gauge configuration. Defining forward covariant finite differences*

$$\left(\Delta^\mu \psi_\sigma\right)(x) := \frac{U_\mu(x)\psi_\sigma(x+\hat{\mu}) - \psi_\sigma(x)}{a} \doteq (\partial_\mu + A_\mu)\psi_\sigma(x)$$

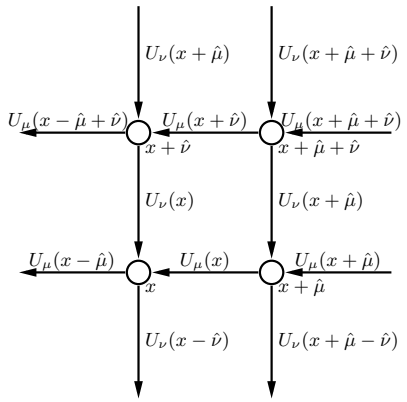

Figure 2.1: Naming conventions on the lattice.

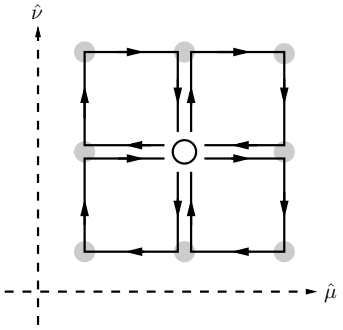

Figure 2.2: The clover term.

and backward covariant finite differences

$$\left(\Delta_\mu \psi_\sigma\right)(x) := \frac{\psi_\sigma(x) - U_\mu^H(x-\hat{\mu})\psi_\sigma(x-\hat{\mu})}{a}\,,$$

the centralized covariant finite difference discretization of the Dirac operator $\mathcal{D}$ is given by

$$D_N := \sum_{\mu=0}^{3} \gamma_\mu \otimes \left(\Delta_\mu + \Delta^\mu\right)/2\,. \tag{2.3}$$

Lemma 2.7
The forward and backward covariant finite differences satisfy

$$\left(\Delta^\mu\right)^H = -\Delta_\mu\,.$$

Proof. For arbitrary quark fields χ and ψ we have

$$\begin{aligned}
\langle \chi_\sigma, \Delta^\mu \psi_\sigma \rangle &= \tfrac{1}{a}\textstyle\sum_{x\in\mathcal{L}} \langle \chi_\sigma(x), \Delta^\mu \psi_\sigma(x)\rangle \\
&= \tfrac{1}{a}\textstyle\sum_{x\in\mathcal{L}} \langle \chi_\sigma(x), U_\mu(x)\psi_\sigma(x+\hat{\mu})\rangle - \tfrac{1}{a}\textstyle\sum_{x\in\mathcal{L}} \langle \chi_\sigma(x), \psi_\sigma(x)\rangle \\
&= (*)\,.
\end{aligned}$$

Since $\mathcal{L}$ is periodic, we can use the index transformation $x \mapsto x+\hat{\mu}$ in the first sum to obtain

$$\begin{aligned}
(*) &= \tfrac{1}{a}\textstyle\sum_{x\in\mathcal{L}} \langle \chi_\sigma(x-\hat{\mu}), U_\mu(x-\hat{\mu})\psi_\sigma(x)\rangle - \tfrac{1}{a}\textstyle\sum_{x\in\mathcal{L}} \langle \chi_\sigma(x), \psi_\sigma(x)\rangle \\
&= -\tfrac{1}{a}\textstyle\sum_{x\in\mathcal{L}} \langle \chi_\sigma(x) - U_\mu^H(x-\hat{\mu})\chi_\sigma(x-\hat{\mu}), \psi_\sigma(x)\rangle \\
&= -\langle \Delta_\mu \chi_\sigma, \psi_\sigma\rangle\,.
\end{aligned}$$

□

Consequently, the centralized covariant finite differences $(\Delta^\mu+\Delta_\mu)/2$ are anti-hermitian. The hermiticity of the γ_μ (see Definition 2.2) then implies the following statement.

Corollary 2.8
The naive discretization D_N from (2.3) *is anti-hermitian, i.e.,*

$$D_N^H = -D_N\,.$$

The naive discretization generates unphysical eigenvectors, a standard phenomenon when discretizing first order derivatives using central finite differences, cf. [105], also known as the "species doubling effect" (see [107,115]) or "red-black instability". The eigenspace for each eigenvalue of D_N has dimension 16, but only a one-dimensional subspace corresponds to an eigenfunction of the continuum operator. Wilson introduced the stabilization term $a\Delta_\mu\Delta^\mu$, a centralized second order covariant finite difference, to circumvent this problem.

Definition 2.9
Given a gauge configuration U on lattice $\mathcal{L}$ with $n_\mathcal{L}$ lattice sites, lattice spacing a and a mass parameter m_0, the Wilson discretization *of the Dirac operator (also* Wilson-Dirac operator*) is defined by*

$$D_W := \frac{m_0}{a} I_{12n_\mathcal{L}} \;+\; \frac{1}{2}\sum_{\mu=0}^{3}\Big(\gamma_\mu\otimes(\Delta_\mu+\Delta^\mu) - aI_4\otimes\Delta_\mu\Delta^\mu\Big)\,, \tag{2.4}$$

where the mass parameter m_0 sets the quark mass.

For further details regarding quark masses and m_0, see [82]. The commutativity relations (2.1) of the γ-matrices imply a non-trivial symmetry of D_W.

Lemma 2.10
Defining $\gamma_5 := \gamma_0\gamma_1\gamma_2\gamma_3$ we have $\gamma_5\gamma_\mu = -\gamma_\mu\gamma_5$ for $\mu = 0,1,2,3$ (see (2.1)*) and with $\Gamma_5 := I_{n_\mathcal{L}}\otimes\gamma_5\otimes I_3$ the Wilson-Dirac operator D_W is* Γ_5-symmetric*, i.e.,*

$$(\Gamma_5 D_W)^H = \Gamma_5 D_W\,. \tag{2.5}$$

Proof. Since γ_μ and γ_5 are hermitian we see that $\gamma_5\gamma_\mu$ is anti-hermitian. Thus the operator $(\gamma_5\gamma_\mu)\otimes(\Delta_\mu+\Delta^\mu)$, as product of two anti-hermitian operators, is hermitian. From Lemma 2.7 we obtain

$$(\Delta_\mu\Delta^\mu)^H = (\Delta^\mu)^H(\Delta_\mu)^H = (-\Delta_\mu)(-\Delta^\mu) = \Delta_\mu\Delta^\mu\,.$$

Hence, $I_4\otimes\Delta_\mu\Delta^\mu$ is hermitian and, trivially, commutes with Γ_5. This implies the Γ_5-symmetry (2.5). □

Remark 2.11 Defining the projectors

$$\pi_\mu^+ := \frac{I_4 + \gamma_\mu}{2} \quad \text{and} \quad \pi_\mu^- := \frac{I_4 - \gamma_\mu}{2}$$

we obtain an equivalent formulation for D_W as

$$D_W\psi(x) = \frac{4+m_0}{a}\psi(x) - \frac{1}{a}\sum_{\mu=0}^{3}\left(\pi_\mu^- \otimes U_\mu(x)\psi(x+\hat{\mu}) + \pi_\mu^+ \otimes U_\mu^H(x-\hat{\mu})\psi(x-\hat{\mu})\right) .$$

From this formulation it is again easy to see that D_W respects Γ_5-symmetry but also that D_W is not hermitian since $(\pi_\mu^+)^H \neq \pi_\mu^-$.

To reduce the order of the discretization error as a function of a, the Sheikholeslami-Wohlert or "clover" term (cf. [102] and Figure 2.2), depending on a parameter c_{sw}, is added to the lattice Wilson-Dirac operator. For the definition of the clover term we first define the plaquettes.

Definition 2.12
Given a configuration of gauge links $\{U_\mu(x)\}$, the plaquette $Q_x^{\mu,\nu}$ *at lattice point x is defined as*

$$Q_x^{\mu,\nu} := U_\nu(x)U_\mu(x+\hat{\nu})U_\nu^H(x+\hat{\mu})U_\mu^H(x)\,. \tag{2.6}$$

A plaquette thus is the product of all coupling matrices along a cycle of length 4 on the torus, such cycles being squares in a (μ,ν)-plane

$$Q_x^{\mu,\nu} \mathrel{\widehat{=}}$$  .

Similarly, the plaquettes in the other quadrants are defined as

$$Q_x^{\mu,-\nu} \mathrel{\widehat{=}}$$  , $\quad Q_x^{-\mu,\nu} \mathrel{\widehat{=}}$  , $\quad Q_x^{-\mu,-\nu} \mathrel{\widehat{=}}$  . (2.7)

Note that on each cycle of length 4 there are four plaquettes which are conjugates of each other. They are defined as the products of the gauge links along that cycle with different starting sites, so that we have, e.g., $Q_{x+\hat{\mu}}^{-\mu,\nu} = U_\mu^H(x)Q_x^{\mu,\nu}U_\mu(x)$, etc.

Definition 2.13
Defining

$$Q_{\mu\nu}(x) := Q_x^{\mu,\nu} + Q_x^{\mu,-\nu} + Q_x^{-\mu,\nu} + Q_x^{-\mu,-\nu}\,,$$

the clover term C is defined as

$$C(x) := \frac{c_{sw}}{32a}\sum_{\mu,\nu=0}^{3}(\gamma_\mu\gamma_\nu)\otimes(Q_{\mu\nu}(x) - Q_{\nu\mu}(x))$$

where $c_{sw} \in \mathbb{R}$ (for illustration see Figure 2.2). For an arbitrary quark field ψ and any lattice site x, the clover improved *Wilson-Dirac operator D is defined as*

$$D\psi(x) := D_W\psi(x) \;-\; C(x)\psi(x)\,. \tag{2.8}$$

The clover term is diagonal on the lattice $\mathcal{L}$. It removes $\mathcal{O}(a)$-discretization effects from the covariant finite difference discretization of the covariant derivative (for appropriately tuned c_{sw}; see [102] and references therein). The resulting discretized Dirac operator D thus has local discretization errors of order $\mathcal{O}(a^2)$. Clearly, putting $c_{sw} = 0$ we retrieve the "non-improved" Wilson discretization.

Lemma 2.14 *(i) The clover Wilson-Dirac operator D is Γ_5-symmetric, i.e.,*

$$(\Gamma_5 D)^H = \Gamma_5 D\,. \tag{2.9}$$

(ii) Every right eigenvector ψ_λ to an eigenvalue λ of D corresponds to a left eigenvector

$$\hat{\psi}_{\bar{\lambda}} = \Gamma_5 \psi_\lambda$$

to the eigenvalue $\bar{\lambda}$ of D and vice versa. In particular, the spectrum of D is symmetric with respect to the real axis.

(iii) The spectrum of D_W is symmetric with respect to the vertical line $\mathrm{Re}(z) = \frac{4+m_0}{a}$, i.e.,

$$\lambda \in \mathrm{spec}\big(D_W\big) \quad \Rightarrow \quad 2\tfrac{m_0+4}{a} - \lambda \in \mathrm{spec}\big(D_W\big).$$

Proof. *(i)* We recall Definition 2.2: For any μ, γ_μ is hermitian and we have

$$\gamma_\nu\gamma_\mu = -\gamma_\mu\gamma_\nu \quad \text{for} \quad \mu \neq \nu\,.$$

Thus, $\gamma_\nu\gamma_\mu$ is anti-hermitian. Further, we have

$$(Q_x^{\mu,\nu})^H = U_\mu(x)U_\nu(x+\hat{\mu})U_\mu^H(x+\hat{\nu})U_\nu^H(x) = Q_x^{\nu,\mu}\,,$$

and thus $Q_{\mu\nu}^H(x) = Q_{\nu\mu}(x)$, i.e., $Q_{\mu\nu}(x) - Q_{\nu\mu}(x)$ is anti-hermitian as well. Thus, the products $(\gamma_\mu\gamma_\nu) \otimes (Q_{\mu\nu}(x) - Q_{\nu\mu}(x))$ are hermitian and therefore the whole clover term C is hermitian.

Furthermore, C commutes with Γ_5 since for all ν, μ

$$\gamma_5(\gamma_\nu\gamma_\mu) = (\gamma_\nu\gamma_\mu)\gamma_5\,.$$

Thus, C is Γ_5-symmetric. From Lemma 2.10 we know that the Wilson-Dirac operator D_W is Γ_5-symmetric, and hence D, the difference of D_W and C, is Γ_5-symmetric as well.

(ii) Due to $D^H = \Gamma_5 D \Gamma_5$ we have

$$D\psi_\lambda = \lambda\psi_\lambda \Leftrightarrow \psi_\lambda^H D^H = \bar{\lambda}\psi_\lambda^H \Leftrightarrow (\Gamma_5\psi_\lambda)^H D = \bar{\lambda}(\Gamma_5\psi_\lambda)^H.$$

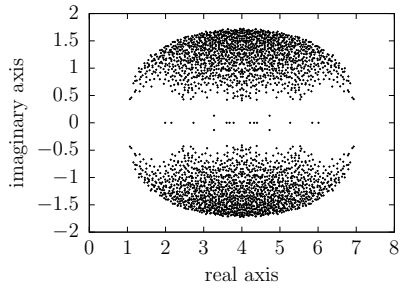

Figure 2.3: Spectrum of a 4^4 Wilson-Dirac operator with $m_0 = 0$ and $c_{sw} = 0$.

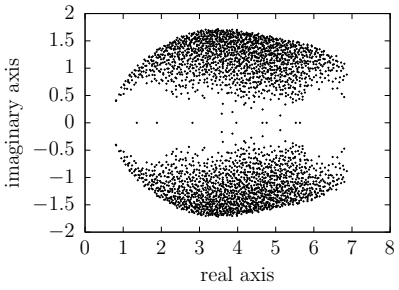

Figure 2.4: Spectrum of a 4^4 "clover improved" Wilson-Dirac operator with $m_0 = 0$ and $c_{sw} = 1$.

(iii) Consider a red-black ordering of the lattice sites, where all red sites appear before black sites. Then the matrix $D_W - \frac{4+m_0}{a} I_{12n_\mathcal{L}}$ has the block structure

$$D_W - \frac{4+m_0}{a} I_{12n_\mathcal{L}} = \begin{pmatrix} 0 & D_{rb} \\ D_{br} & 0 \end{pmatrix}.$$

Thus, if $x = (x_r, x_b)$ is an eigenvector of $D_W - \frac{4+m_0}{a} I_{12n_\mathcal{L}}$ with eigenvalue μ, then $x' = (x_r, -x_b)$ is an eigenvector of $D_W - \frac{4+m_0}{a} I_{12n_\mathcal{L}}$ with eigenvalue $-\mu$. Applying this result to D_W gives the assertion.

□

Summarizing, $D \in \mathbb{C}^{n \times n}$ is a sparse matrix which represents a nearest neighbor coupling on a periodic 4D lattice. The lattice has $n_\mathcal{L} = N_t N_s^3$ sites, each holding 12 variables, so that $n = 12 n_\mathcal{L}$. D has the symmetry property (2.9), depends on a mass parameter m_0, the Sheikholeslami-Wohlert constant c_{sw}, and a configuration $\{U_\mu(x) : x \in \mathcal{L}, \mu = 0, 1, 2, 3\}$. In practice m_0 is negative, and for physically relevant mass parameters, the spectrum of D is contained in the right half plane, cf. Figures 2.3 and 2.4.

To explicitly formulate D in matrix terms we fix a representation for the γ-matrices as

$$\gamma_0 = \begin{pmatrix} & & & i \\ & & i & \\ & -i & & \\ -i & & & \end{pmatrix}, \gamma_1 = \begin{pmatrix} & & & -1 \\ & & 1 & \\ & 1 & & \\ -1 & & & \end{pmatrix}, \gamma_2 = \begin{pmatrix} & & i & \\ & & & -i \\ -i & & & \\ & i & & \end{pmatrix}, \gamma_3 = \begin{pmatrix} & & & 1 \\ & & & 1 \\ 1 & & & \\ & 1 & & \end{pmatrix},$$

resulting in

$$\gamma_5 = \gamma_0\gamma_1\gamma_2\gamma_3 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix}.$$

Thus γ_5 acts as the identity on spins 0 and 1 and as the negative identity on spins 2 and 3. D is then given via

$$\begin{aligned}(D\psi)(x) &= \frac{1}{a}\Big((m_0+4)I_{12} - \frac{c_{sw}}{32}\sum_{\mu,\nu=0}^{3}(\gamma_\mu\gamma_\nu)\otimes\big(Q_{\mu\nu}(x)-Q_{\nu\mu}(x)\big)\Big)\psi(x)\\ &\quad - \frac{1}{2a}\sum_{\mu=0}^{3}\left((I_4-\gamma_\mu)\otimes U_\mu(x)\right)\psi(x+\hat{\mu})\\ &\quad - \frac{1}{2a}\sum_{\mu=0}^{3}\left((I_4+\gamma_\mu)\otimes U_\mu^H(x-\hat{\mu})\right)\psi(x-\hat{\mu}).\end{aligned}$$

2.3 Smearing and Normality

Definition 2.15
A linear operator A acting on a finite dimensional vector space is normal *if*

$$A^H A = A\,A^H\,.$$

Equivalently, A is normal if and only if A is unitarily similar to a diagonal matrix ([100]). Extending Definition 2.16, a continuous linear operator acting on an infinite dimensional Hilbert space is normal if it commutes with its adjoint. The continuum Dirac operator $\mathcal{D}$ is anti-self-adjoint if one restricts the matter field to an appropriate Hilbert space and thus normal which we do not discuss in detail here.

Definition 2.16
For a linear operator A acting on a finite dimensional vector space V, the field of values *of A is given by*

$$\mathcal{F}(A) = \{\psi^H A\psi : \psi^H\psi = 1,\, \psi\in V\}\,.$$

For normal operators the field of values is the convex hull of the spectrum (see, e.g., [100]). The Wilson-Dirac operator D is not normal, but it approaches normality when discretization effects become smaller. Thus, for small lattice spacing, large lattice sizes and physically relevant mass parameters we can expect that the whole field of values $\mathcal{F}(D)$ of D is in the right half plane. This has numerical advantages as, e.g., if the field of values excludes the origin, the restarted Generalized Minimal Residual (GMRES(m)) method is known to converge, see, e.g., [100].

In order to improve normality and reduce fluctuation of the spectrum, *smearing* techniques as, e.g., "stout" [83], APE [1], HYP [61] and HEX [28] smearing were introduced to lattice QCD simulations. Typically, a smearing is an iterative process modifying the gauge links by averaging them with neighboring links.

To measure the deviation from normality of D_W, the Wilson-Dirac operator without clover term, we now look at the Frobenius norm of $D_W^H D_W - D_W D_W^H$. We show that this measure can be fully expressed in terms of the pure gauge action, defined as a sum of the plaquettes (Definition 2.12). Based on this connection we then explain that "stout" smearing [83] has the effect of reducing the non-normality of D_W, among its other physical benefits. Another algorithmic benefit is discussed in Chapter 9. We published this part of the thesis (for the non-improved Wilson-Dirac operator) in [24].

For ease of notation we from now on drop the lattice spacing a, i.e., $a = 1$, so that the lattice $\mathcal{L}$ is given as

$$\mathcal{L} = \{x = (x_0, x_1, x_2, x_3) \,:\, 1 \leq x_0 \leq N_t,\, 1 \leq x_1, x_2, x_3 \leq N_s\}\,.$$

The deviation of the plaquettes from the identity is a measure for the non-normality of D as determined by the following proposition.

Proposition 2.17
The Frobenius norm of $D_W^H D_W - D_W D_W^H$ fulfills

$$\|D_W^H D_W - D_W D_W^H\|_F^2 = 16 \sum_x \sum_{\mu<\nu} \mathrm{Re}(\mathrm{tr}\,(I - Q_x^{\mu,\nu})) \tag{2.10}$$

where the first sum is to be taken over all lattice sites x and $\sum_{\mu<\nu}$ is a shorthand for $\sum_{\mu=0}^{3}\sum_{\nu=\mu+1}^{3}$.

Proof. In order to prove the proposition we inspect the entries of $D_W^H D_W - D_W D_W^H$. As in Remark 2.11 we use the notation $\pi_\mu^\pm$ for the matrices

$$\pi_\mu^\pm = \tfrac{1}{2}(I_4 \pm \gamma_\mu),\ \mu = 0, \ldots, 3\,.$$

The relations (2.1) between the γ-matrices show that each $\pi_\mu^\pm$ is a projection and that, in addition,

$$\pi_\mu^+ \pi_\mu^- = \pi_\mu^- \pi_\mu^+ = 0,\ \mu = 0, \ldots, 3\,. \tag{2.11}$$

Table 2.1 gives the nonzero entries of a (block) row in D_W and D_W^H in terms of the 12×12 matrices which couple lattice site x with the sites x and $x \pm \hat{\mu}$. We use m to denote $m_0 + 4$ with m_0 from (2.4).

The product $D_W^H D_W$ represents a coupling between nearest and next-to-nearest lattice sites; the coupling 12×12 matrices are obtained as the sum over all paths of length 2 on the torus of the product of the respective coupling matrices in D_W^H and D_W. A similar observation holds for $D_W D_W^H$. Table 2.2 reports all the entries of $D_W^H D_W$, and we now shortly discuss all the paths of length 2 which contribute to these entries of $D_W^H D_W$.

	D_W	D_W^H
(x,x)	mI_{12}	mI_{12}
$(x,x+\hat{\mu})$	$-\pi_\mu^- \otimes U_\mu(x)$	$-\pi_\mu^+ \otimes U_\mu(x)$
$(x,x-\hat{\mu})$	$-\pi_\mu^+ \otimes U_\mu^H(x-\hat{\mu})$	$-\pi_\mu^- \otimes U_\mu^H(x-\hat{\mu})$

Table 2.1: Coupling terms in D_W and D_W^H.

	$D_W D_W^H$
(x,x)	$m^2 I_{12} + \frac{1}{2}\sum_{\mu=0}^{3}(\pi_\mu^+ + \pi_\mu^-) \otimes I_3$
$(x,x+\hat{\mu})$	$-m(\pi_\mu^+ + \pi_\mu^-) \otimes U_\mu(x)$
$(x,x-\hat{\mu})$	$-m(\pi_\mu^+ + \pi_\mu^-) \otimes U_\mu(x-\hat{\mu})$
$(x,x\pm 2\hat{\mu})$	0
$\nu \neq \mu$:	
$(x,x+\hat{\mu}+\hat{\nu})$	$\pi_\mu^-\pi_\nu^+ \otimes U_\mu(x)U_\nu(x+\hat{\mu})$
	$+\pi_\nu^-\pi_\mu^+ \otimes U_\nu(x)U_\mu(x+\hat{\nu})$
$(x,x+\hat{\mu}-\hat{\nu})$	$\pi_\mu^-\pi_\nu^- \otimes U_\mu(x)U_\nu^H(x+\hat{\mu}-\hat{\nu})$
	$+\pi_\nu^+\pi_\mu^+ \otimes U_\nu^H(x-\hat{\nu})U_\mu(x-\hat{\nu})$
$(x,x-\hat{\mu}-\hat{\nu})$	$\pi_\mu^+\pi_\nu^- \otimes U_\mu^H(x-\hat{\mu})U_\nu^H(x-\hat{\mu}-\hat{\nu})$
	$+\pi_\nu^+\pi_\mu^- \otimes U_\nu^H(x-\hat{\nu})U_\mu^H(x-\hat{\nu}-\hat{\mu})$

Table 2.2: Coupling terms in $D_W^H D_W$. The coupling terms in $D_W D_W^H$ are obtained by interchanging all π_μ^+ and π_μ^- as well as all π_ν^+ and π_ν^-.

For the diagonal position (x,x) we have 9 paths of length 2, $(x,x) \to (x,x) \to (x,x)$ and $(x,x) \to (x,x\pm\hat{\mu}) \to (x,x), \mu = 0,\ldots,3$. For a nearest neighbor $(x,x+\hat{\mu})$ we have the two paths $(x,x) \to (x,x) \to (x,x+\hat{\mu})$ and $(x,x) \to (x,x+\hat{\mu}) \to (x,x+\hat{\mu})$, and similarly in the negative directions. For a position $(x,x\pm 2\hat{\mu})$ there is only one path $(x,x) \to (x,x\pm\hat{\mu}) \to (x,x\pm 2\hat{\mu})$, with the product of the couplings being 0 due to (2.11). Finally, for the other next-to-nearest neighbors we always have two paths, for example $(x,x) \to (x,x+\hat{\mu}) \to (x+\hat{\mu}-\hat{\nu})$ and $(x,x) \to (x,x-\hat{\nu}) \to (x+\hat{\mu}-\hat{\nu})$.

The coupling terms in $D_W D_W^H$ are identical to those for $D_W^H D_W$ except that we have to interchange all π_μ^+ and π_μ^- as well as all π_ν^+ and π_ν^-.

This shows that in $D_W^H D_W - D_W D_W^H$ the only non-vanishing coupling terms are those at positions $(x,x+\hat{\mu}+\hat{\nu})$, $(x,x+\hat{\mu}-\hat{\nu})$ and $(x,x-\hat{\mu}-\hat{\nu})$ for $\mu \neq \nu$.

$\mu \neq \nu$:	$D_W^H D_W - D_W D_W^H$
$(x, x+\hat{\mu}+\hat{\nu})$	$\frac{1}{2}(-\gamma_\mu + \gamma_\nu) \otimes (I_3 - Q_x^{\mu,\nu})\, U_\mu(x) U_\nu(x+\hat{\mu})$
$(x, x+\hat{\mu}-\hat{\nu})$	$\frac{1}{2}(-\gamma_\mu - \gamma_\nu) \otimes (I_3 - Q_x^{\mu,-\nu})\, U_\mu(x) U_\nu^H(x+\hat{\mu}-\hat{\nu})$
$(x, x-\hat{\mu}-\hat{\nu})$	$\frac{1}{2}(\gamma_\mu - \gamma_\nu) \otimes (I_3 - Q_x^{-\mu,-\nu})\, U_\mu^H(x-\hat{\mu}) U_\nu^H(x-\hat{\mu}-\hat{\nu})$

Table 2.3: Coupling terms in $D_W^H D_W - D_W D_W^H$.

They are given in Table 2.3, where we used the identities

$$\begin{aligned}
\pi_\mu^- \pi_\nu^- - \pi_\mu^+ \pi_\nu^+ &= \tfrac{1}{2}\,(-\gamma_\mu - \gamma_\nu)\,, \\
\pi_\mu^+ \pi_\nu^- - \pi_\mu^- \pi_\nu^+ &= \tfrac{1}{2}\,(\gamma_\mu - \gamma_\nu)\,, \\
\pi_\mu^- \pi_\nu^+ - \pi_\mu^+ \pi_\nu^- &= \tfrac{1}{2}\,(-\gamma_\mu + \gamma_\nu)\,, \\
\pi_\mu^+ \pi_\nu^+ - \pi_\mu^- \pi_\nu^- &= \tfrac{1}{2}\,(\gamma_\mu + \gamma_\nu)\ .
\end{aligned}$$

By rearranging the terms we obtain the plaquettes from (2.6) and (2.7). We exemplify this for position $(x, x+\hat{\mu}+\hat{\nu})$

$$\begin{aligned}
&\pi_\mu^- \pi_\nu^+ \otimes U_\mu(x) U_\nu(x+\hat{\mu}) + \pi_\nu^- \pi_\mu^+ \otimes U_\nu(x) U_\mu(x+\hat{\nu}) \\
-\ &\big(\pi_\mu^+ \pi_\nu^- \otimes U_\mu(x) U_\nu(x+\hat{\mu}) + \pi_\nu^+ \pi_\mu^- \otimes U_\nu(x) U_\mu(x+\hat{\nu})\big) \\
&\quad = \tfrac{1}{2}(-\gamma_\mu + \gamma_\nu) \otimes U_\mu(x) U_\nu(x+\hat{\mu}) + \tfrac{1}{2}(\gamma_\mu - \gamma_\nu) \otimes U_\nu(x) U_\mu(x+\hat{\nu}) \\
&\quad = \tfrac{1}{2}(-\gamma_\mu + \gamma_\nu) \otimes (I_3 - Q_x^{\mu,\nu})\, U_\mu(x) U_\nu(x+\hat{\mu})\,.
\end{aligned}$$

Using the fact that for the Frobenius norm we have

$$\begin{aligned}
\|AQ\|_F &= \|A\|_F \text{ whenever } Q \text{ is unitary (and } AQ \text{ is defined)}, \\
\|A \otimes B\|_F &= \|A\|_F \cdot \|B\|_F \text{ for all } A, B,
\end{aligned}$$

we obtain the following for the squares of the Frobenius norms of all the coupling matrices from Table 2.3:

$$\begin{aligned}
&2\|I - Q_x^{\mu,\nu}\|_F^2 && \text{for position } (x, x+\hat{\mu}+\hat{\nu})\,, \\
&2\|I - Q_x^{\mu,-\nu}\|_F^2 && \text{for position } (x, x+\hat{\mu}-\hat{\nu})\,, \\
&2\|I - Q_x^{-\mu,-\nu}\|_F^2 && \text{for position } (x, x-\hat{\mu}-\hat{\nu})\,.
\end{aligned}$$

Finally, for any unitary matrix Q we have

$$\|I - Q\|_F^2 = \operatorname{tr}\big((I - Q^H)(I - Q)\big) = 2 \cdot \operatorname{Re}(\operatorname{tr}(I - Q))\,.$$

Now we obtain $\|D_W^H D_W - D_W D_W^H\|_F^2$ by summing the squares of the Frobenius norms of all coupling matrices. This is a sum over $24n$ coupling matrices, n being the number of lattice sites. As discussed before, groups of four of these coupling matrices refer to the same plaquette $Q_x^{\mu,\nu}$ up to conjugation in SU(3), so $\operatorname{tr}(I - Q)$

is the same for these four plaquettes Q. We can thus "normalize" to only consider all possible "first quadrant" plaquettes $Q_x^{\mu,\nu}$ and obtain

$$\|D_W^H D_W - D_W D_W^H\|_F^2 = 4 \sum_x \sum_{\mu<\nu} 2 \cdot 2 \cdot \mathrm{Re}(\mathrm{tr}\,(I - Q_x^{\mu,\nu}))\,.$$

□

As a consequence of Proposition 2.17 we conclude that D_W is normal in the case of the *free theory*, i.e., when all links $U_\mu(x)$ are equal to the identity or when $U_\mu(x) = U(x)U^H(x+\hat{\mu})$ for a collection of $SU(3)$-matrices $U(x)$ on the lattice sites x. For physically relevant configurations, however, D_W is non-normal.

For the deviation from normality of the clover improved Wilson-Dirac operator $D = D_W + C$ we have

$$\begin{aligned} \|D^H D - D D^H\|_F &= \|D_W^H D_W - D_W D_W^H + (D_W^H - D_W)C - C(D_W^H - D_W)\|_F \\ &\leq \|D_W^H D_W - D_W D_W^H\|_F + 2\|C\|_F \|D_W^H - D_W\|_F\,. \end{aligned}$$

From Proposition 2.17 we know that $\|D_W^H D_W - D_W D_W^H\|_F \to 0$ implies $Q_x^{\mu,\nu} \to I$ for all x. From the clover term Definition 2.13 we see that then also $Q_{\mu\nu}(x) - Q_{\nu\mu}(x) \to 0$ for all x and μ, ν, i.e., $\|C\|_F \to 0$. Thus,

$$\|D_W^H D_W - D_W D_W^H\|_F \to 0 \quad \text{implies} \quad \|D^H D - D D^H\|_F \to 0\,.$$

Definition 2.18
For a given configuration $U = \{U_\mu(x)\}$, the quantity

$$S_W(U) = \sum_x \sum_{\mu<\nu} Re(\mathrm{tr}\,(I - Q_x^{\mu,\nu}))$$

is known as the Wilson gauge action[1].

Smearing techniques for averaging neighboring gauge links have been studied extensively in lattice QCD simulations. Their use in physics is motivated by the goal to reduce "cut-off effects" related to localized eigenvectors with near zero eigenvalues. We now explain why "stout" smearing [83] reduces the Wilson gauge action and thus drives the Wilson-Dirac operator towards normality. Other smearing techniques like APE [1], HYP [61] and HEX [28] have similar effects.

Given a configuration U, stout smearing modifies the gauge links according to

$$U_\mu(x) \to \tilde{U}_\mu(x) = \mathrm{e}^{\varepsilon Z_\mu^{\mathcal{U}}(x)} U_\mu(x)\,, \tag{2.12}$$

[1]To represent a physically meaningful quantity, the Wilson gauge action is usually scaled with a scalar factor. This is not relevant in the present context.

where the parameter ε is a small positive number and

$$Z_\mu^{\mathcal{U}}(x) \;=\; -\frac{1}{2}(M_\mu(x) - M_\mu^H(x)) + \frac{1}{6}\mathrm{tr}\left(M_\mu(x) - M_\mu^H(x)\right)\,, \qquad (2.13)$$

where

$$M_\mu(x) \;=\; \sum_{\nu=0,\nu\neq\mu}^{3} Q_x^{\mu,\nu} + Q_x^{\mu,-\nu}\,.$$

Note the dependence of $Z_\mu^{\mathcal{U}}(x)$ on local plaquettes associated with x.

The following result from [80, 81] relates the *Wilson flow* $\mathcal{V}(\tau) = \{V_\mu(x,\tau) : x \in \mathcal{L}, \mu = 0,\ldots,3\}$ defined as the solution of the initial value problem

$$\frac{\partial}{\partial\tau}V_\mu(x,\tau) = -\left\{\partial_{x,\mu}S_{\mathrm{W}}(\mathcal{V}(\tau))\right\}V_\mu(x,\tau)\,, \quad V_\mu(x,0) = U_\mu(x)\,, \qquad (2.14)$$

to stout smearing. Here $V_\mu(x,\tau) \in \mathrm{SU}(3)$, and $\partial_{x,\mu}$ is the canonical differential operator with respect to the link variable $V_\mu(x,\tau)$ which takes values in $\mathfrak{su}(3)$.

Theorem 2.19
Let $\mathcal{V}(\tau)$ be the solution of (2.14)*. Then*

(i) $\mathcal{V}(\tau)$ is unique for all $\mathcal{V}(0)$ and all $\tau \in (-\infty,\infty)$ and differentiable with respect to τ and $\mathcal{V}(0)$.

(ii) $S_W(\mathcal{V}(\tau))$ is monotonically decreasing as a function of τ.

(iii) One step of Lie-Euler integration with step size ε for (2.14)*, starting at $\tau = 0$, gives the approximation $\widetilde{\mathcal{V}}(\varepsilon) = \{\widetilde{V}_\mu(x,\varepsilon)\}$ for $\mathcal{V}(\varepsilon)$ with*

$$\widetilde{V}_\mu(x,\varepsilon) = \mathrm{e}^{\varepsilon Z_\mu^{\mathcal{U}}(x)}U_\mu(x)\,,$$

with $Z_\mu^{\mathcal{U}}(x)$ from (2.13)*.*

We refer to [80, 81] and also [16] for details of the proof for (i) and (ii). It is noted in [16] that the solution of (2.14) moves the gauge configuration along the steepest descent direction in configuration space and thus actually minimizes the action locally. Part (iii) follows directly by applying the Lie-Euler scheme; cf. [60].

The theorem implies that one Lie-Euler step is equivalent to a step of stout smearing, with the exception that in stout smearing links are updated sequentially instead of in parallel. And since the Wilson action decreases along the exact solution of (2.14), we can expect it to also decrease for its Lie-Euler approximation, at least when ε is sufficiently small.

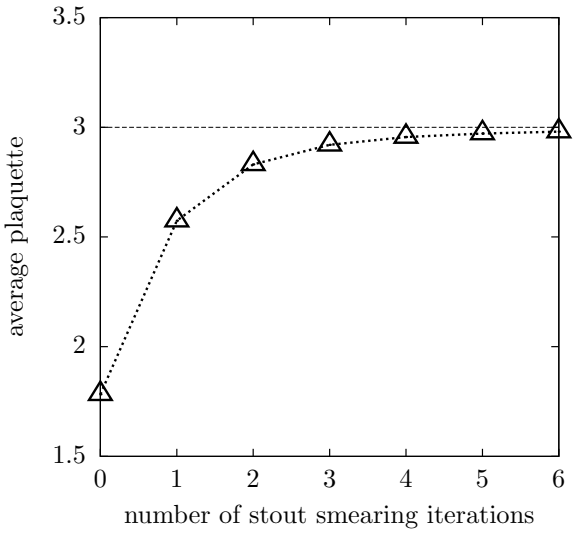

Figure 2.5: Illustration of the effect of stout smearing on the average plaquette value (2.15).

In Figure 2.5 we illustrate the relation between iterations of stout smearing and the average plaquette value Q_{avg} for a 32^4 lattice. The average plaquette value is defined by

$$Q_{avg} = N_Q^{-1} \sum_x \sum_{\mu<\nu} Re(\mathrm{tr}\,(Q_x^{\mu,\nu}))\,, \tag{2.15}$$

where N_Q denotes the total number of plaquettes. In terms of Q_{avg} (2.10) simplifies to

$$\|D_W^H D_W - D_W D_W^H\|_F = 16 N_Q (3 - Q_{avg})\,.$$

Figure 2.5 shows that the Wilson action decreases rapidly in the first iterations of stout smearing.

To conclude this section we note that there are several works relating the spectral structure and the distribution of plaquette values. For example, it has been shown in [87] that the size of the spectral gap around 0 of $\Gamma_5 D_W$ is related to $\mathrm{Re}(\mathrm{tr}\,(I - Q_x^{\mu,\nu}))$ being larger than a certain threshold for all plaquettes $Q_x^{\mu,\nu}$. Other studies consider the connection between fluctuations of the plaquette value and spacially localized near zero modes (eigenvectors with near zero eigenvalues), see [13, 84, 88], and the influence of smearing on these modes [62].

Chapter 3

Domain Decomposition Methods

Let us reserve the terminology *block decomposition* for a tensor type decomposition of $\mathcal{L}$ into lattice-blocks. The precise definition is as follows.

Definition 3.1
Assume that $\{\mathcal{T}_1^0, \ldots, \mathcal{T}_{\ell_0}^0\}$ *is a partitioning of* $\{1, \ldots, N_t\}$ *into* ℓ_0 *blocks of consecutive time points,*

$$\mathcal{T}_j^0 = \{t_{j-1} + 1, \ldots, t_j\}, \quad j = 1, \ldots, \ell_0,\, 0 = t_0 < t_1 \ldots < t_{\ell_0} = N_t\,,$$

and similarly for the spatial dimensions with partitionings $\{\mathcal{T}_1^\mu, \ldots, \mathcal{T}_{\ell_\mu}^\mu\}$, $\mu = 1, 2, 3$.

A block decomposition *of* $\mathcal{L}$ *is a partitioning of* $\mathcal{L}$ *into* $\ell = \ell_0\ell_1\ell_2\ell_3$ *lattice-blocks* $\mathcal{L}_i$*, where each lattice-block is of the form*

$$\mathcal{L}_i = \mathcal{T}_{j_0(i)}^0 \times \mathcal{T}_{j_1(i)}^1 \times \mathcal{T}_{j_2(i)}^2 \times \mathcal{T}_{j_3(i)}^3\,.$$

Accordingly we define a block decomposition of all $12n_{\mathcal{L}}$ *variables in* $\mathcal{V} = \mathcal{L} \times \mathcal{C} \times \mathcal{S}$ *into* ℓ *blocks* $\mathcal{V}_i$ *by grouping all spin and color components corresponding to the lattice-block* $\mathcal{L}_i$*, i.e.,*

$$\mathcal{V}_i = \mathcal{L}_i \times \mathcal{C} \times \mathcal{S}\,. \tag{3.1}$$

Another block decomposition $\{\mathcal{L}_i' : i = 1, \ldots, t'\}$ *is called* refinement *of* $\{\mathcal{L}_i : i = 1, \ldots, t\}$*, if for each* $\mathcal{L}_i'$ *there exists a* $\mathcal{L}_j$ *such that*

$$\mathcal{L}_i' \subseteq \mathcal{L}_j\,.$$

The computational challenge in lattice QCD computations consists of repeatedly solving linear systems involving the discretized Wilson-Dirac operator

$$D\psi = \eta. \tag{3.2}$$

Since the systems arising in lattice QCD calculations tend to have hundreds of millions of unknowns they require the use of parallel computers. Therefore, block decompositions are used for parallelization. Each process obtains a block of lattice sites from a block decomposition, administrates the corresponding variables and performs the calculations associated with these lattice sites. Figure 3.1 shows two processes from a parallel environment, reduced to two dimensions. Each process owns a lattice-block $\mathcal{P}_i$ of a block decomposition and ghost cells on the boundaries which contain the recent values from neighboring processes. Therefore, ghost cells have to be updated whenever a computation demands information from nearest neighbor lattice sites, e.g., when evaluating the Wilson-Dirac operator D on a vector ψ. Here, this updating process is represented by the arrows pointing to the ghost cells.

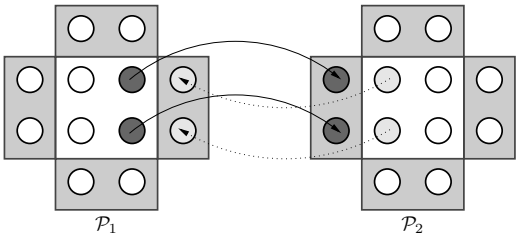

Figure 3.1: 2D example for the communication of two neighboring processes from a parallel setting.

This parallelization scheme is in a natural way compatible with domain decomposition methods as preconditioners for (3.2). These methods consist of repeatedly solving smaller block restricted linear systems. In a parallel environment they are able to improve the ratio of computation and communication if the block decomposition of the domain decomposition method is a refinement of the block decomposition $\{\mathcal{P}_i\}$ underlying the parallelization. Furthermore, due to the smaller linear system solves, the work to be done tends to be more local, hence reducing the access to local memory.

3.1 Additive and Multiplicative Schwarz

We now define the instruments needed for domain decomposition methods.

Definition 3.2
Let $\mathcal{V}_i \subset \mathcal{V}$ be a lattice-block. We define the corresponding trivial embedding

$$\mathcal{I}_{\mathcal{V}_i} : \mathcal{V}_i \to \mathcal{V}$$

as the restriction of the identity on $\mathcal{V}$ to $\mathcal{V}_i$, i.e.,

$$\mathcal{I}_{\mathcal{V}_i} := (id_{\mathcal{V}})|_{\mathcal{V}_i} .$$

The corresponding block inverse *is defined as*

$$B_i := \mathcal{I}_{\mathcal{V}_i} D_i^{-1} \mathcal{I}_{\mathcal{V}_i}^H \quad \textit{where} \quad D_i := \mathcal{I}_{\mathcal{V}_i}^H D \mathcal{I}_{\mathcal{V}_i} .$$

Lemma 3.3
Consider the iteration

$$\psi^{k+1} = H\psi^k + L\eta \quad \textit{with} \quad H, L \in \mathbb{C}^{n\times n} \textit{ and } \psi^i, \eta \in \mathbb{C}^n .$$

(i) If ψ^* is a fixed point of the iteration, i.e., ψ^* satisfies $\psi^* = H\psi^* + L\eta$, then the errors $e^k := \psi^* - \psi^k$ satisfy

$$e^{k+1} = He^k .$$

The matrix H is called error propagator.

(ii) If we have the initial guess $\psi^0 = 0$, then the k-th iterate is given by

$$\psi^k = \sum_{i=0}^{k-1} H^i L\eta .$$

Proof. *(i)* We have

$$e^{k+1} = \psi^* - \psi^{k+1} = \underbrace{H\psi^* + L\eta}_{=\psi^*} - \underbrace{(H\psi^k + L\eta)}_{=\psi^{k+1}} = H(\psi^* - \psi^k) = He^k .$$

(ii) We have $\psi^1 = H\psi^0 + L\eta = H^0 L\eta$ and inductively

$$\psi^{k+1} = H(\sum_{i=0}^{k-1} H^i L\eta) + L\eta = \sum_{i=1}^{k} H^i L\eta + H^0 L\eta = \sum_{i=0}^{k} H^i L\eta .$$

□

For a given block decomposition $\{\mathcal{V}_i : i = 1, \ldots, k\}$ of $\mathcal{V}$, one iteration of a domain decomposition method to solve the linear system (3.2) consists of solving each of the block systems

$$D_i e_i = \mathcal{I}_{\mathcal{V}_i}^H r \tag{3.3}$$

and applying the corrections

$$\psi \leftarrow \psi + B_i r \quad \text{where } B_i r = \mathcal{I}_{\mathcal{V}_i} e_i \quad \text{for } i = 1, \ldots, k , \tag{3.4}$$

possibly performing a certain number of in-between residual updates

$$r \leftarrow \eta - D\psi . \tag{3.5}$$

In case the residual update (3.5) is done only once before solving all block systems, all the corrections (3.4) are performed for the same residual r. Using the shorthand

$$M = \sum_{i=1}^{k} B_i \,,$$

one iteration of the domain decomposition method is then given by

$$\psi \leftarrow \psi + M(\eta - D\psi) = (I - MD)\psi + M\eta$$

yielding the error propagator

$$H = I - MD = I - \sum_{i=1}^{k} B_i D$$

via Lemma 3.3.

Similarly, if the residual update (3.5) is performed after each solution of a block system (3.3), we obtain the error propagator

$$H = \prod_{i=1}^{k} (I - B_i D) \ .$$

These simplest instances of domain decomposition methods were already used in the context of a theoretical analysis of PDEs by Schwarz in 1870 [101]. They are now known as the additive and the multiplicative Schwarz method (Algorithm 1 and 2), see, e.g., [104].

Algorithm 1: Additive Schwarz (one iteration)

input: ψ, η
output: ψ
1 $r \leftarrow \eta - D\psi$
2 **for** $i = 1$ to k
3 $\quad \psi \leftarrow \psi + B_i r$

Since in Algorithm 1 the residual r does not change within the whole `for`-loop, the block systems can be solved simultaneously, which lends itself to an easy parallelization.

On the other hand, in Algorithm 2, the residual changes in each cycle of the `for`-loop depending on the solution of the previous cycle. Thus, the information spreads faster than in Algorithm 1, but Algorithm 2 is inherently sequential.

Algorithm 2: Multiplicative Schwarz (one iteration)

input: ψ, η
output: ψ
for $i = 1$ to k
  $r \leftarrow \eta - D\psi$
  $\psi \leftarrow \psi + B_i r$

3.2 Red-Black Multiplicative Schwarz

This section is largely based on [49]. The red-black Schwarz method has been introduced to lattice QCD in [78] and has been used ever since in several lattice QCD implementations as a preconditioner (cf. [3, 50, 79]). In this context red-black Schwarz is also known as Schwarz Alternating Procedure (SAP). SAP is a colored version of the multiplicative Schwarz method [101, 104] that allows for parallelization. To this purpose we color the blocks such that blocks of the same color are decoupled (Figure 3.2). After a residual update all block systems of a given color can be solved (simultaneously) before computing the next residual update. Additional residual updates in-between the block solves do not show any effect. Since the Wilson-Dirac operator has only nearest neighbor couplings, only two colors are needed to decouple. This red-black Schwarz method for the solution of (3.2) is given in Algorithm 3 for a block decomposition of the lattice and variable blocks $\mathcal{V}_i$ according to (3.1). Figure 3.2 illustrates a 2D example.

With the shorthand $B_{color} = \sum_{i \in color} B_i$ and

$$K = B_{black}(I - D\, B_{red}) + B_{red}$$

we can summarize one iteration ($\nu = 1$) of Algorithm 3 as (cf. [104])

$$\psi \leftarrow (I - KD)\psi + K\eta\,. \tag{3.6}$$

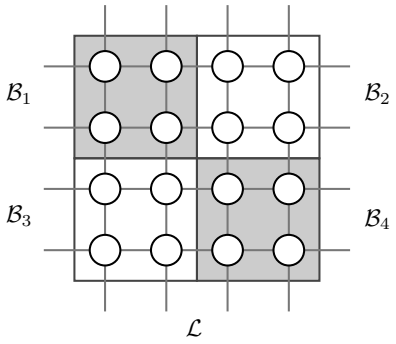

Figure 3.2: Block decomposed lattice (reduced to 2D) with 2 colors.

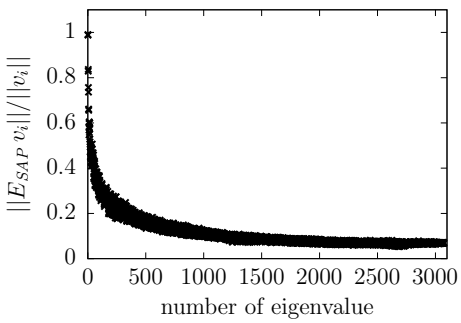

Figure 3.3: Error component reduction on a 4^4 lattice with block size 2^4.

Algorithm 3: Red-black multiplicative Schwarz

input: ψ, η, ν
output: ψ
for $i = 1$ to ν
  $r \leftarrow \eta - D\psi$
  for all *red* i **do**
    $\psi \leftarrow \psi + B_i r$
  $r \leftarrow \eta - D\psi$
  for all *black* i **do**
    $\psi \leftarrow \psi + B_i r$

According to Lemma 3.3 the error propagator is given by

$$E_{SAP} = I - KD = (I - B_{black}D)(I - B_{red}D)\,,$$

and for the initial guess $\psi = 0$, we obtain after ν iterations

$$M_{SAP}^{(\nu)}\eta = \sum_{i=0}^{\nu-1}(I - KD)^i\, K\, \eta\,.$$

We denote $E_{SAP} = I - M_{SAP}D$ where $M_{SAP} = M_{SAP}^{(1)}$.

Typically, the solution of the local systems (3.3), required when computing $B_i r$, is approximated by a few iterations of a Krylov subspace method (e.g., GMRES). When incorporating such an approximate solver, the SAP method becomes a non-stationary iterative process. Hence it is necessary to use flexible Krylov subspace methods such as FGMRES or GCR (cf. [100]) in case that SAP is used as a preconditioner (cf. [50, 78, 100]).

It turns out that SAP as a preconditioner is not able to remedy the unfavorable scaling behavior of Krylov subspace methods with respect to system size, quark mass and physical volume. When analyzing this behavior, one realizes that SAP reduces error components belonging to a large part of the spectrum very well, but a small part is almost unaffected by SAP. We illustrate this in Figure 3.3 where the horizontal axis represents the eigenvectors v of D in ascending order of the absolute value of the corresponding eigenvalue. The vertical axis gives the ratio $\|E_{SAP}v\|/\|v\|$, see Figure 3.3. The ratio is small for larger eigenvalues and becomes significantly larger for the small eigenvalues. This behavior is typical for a smoother in an algebraic multigrid method which motivated us to use SAP in this context.

3.3 Sixteen Color Multiplicative Schwarz

In this section we present the extension of a 2D four color approach [15] to the 4D case with nearest neighbor coupling. Instead of using standard two color SAP (cf. Algorithm 3) we apply a coloring scheme with 16 colors which spreads information on the lattice faster, yielding faster convergence. Furthermore, this approach can allow for more overlap of communication and computation in certain situations.

Figure 3.4 illustrates one iteration of the four color approach in 2D for nearest neighbor coupling. Each sub-figure displays one and the same processor lattice-block from a process decomposition $\mathcal{P}_i$, containing a refining block decomposition $\mathcal{B}_i$ (of 8 sub blocks) used for the Schwarz method. We call $\mathcal{P}_i$ *local lattice* or *process lattice* and $\mathcal{B}_i$ a *block*. The thin blocks surrounding $\mathcal{P}_i$ represent the *ghost cells* which contain values from the boundaries of neighboring process blocks $\mathcal{P}_j$.

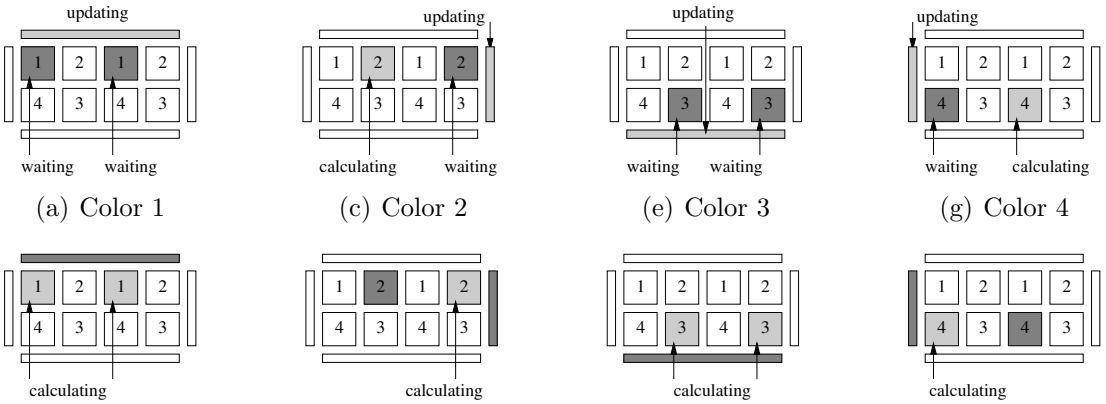

Figure 3.4: Example for one iteration of the four color approach in 2D, to be read column wise.

In the very beginning, before starting the iteration, the left ghost cells have to be updated as well, such that all ghost cells adjacent to color 1 contain the latest information. For this particular example in Figure 3.4, each Schwarz block $\mathcal{B}_i$ is adjacent to at least one ghost cell. This means, there are no *inner blocks* which could be processed by two color SAP while the ghost cells are being updated. However, while communicating, the four color approach is able to process all blocks $\mathcal{B}_i$ which are inner blocks in the current communication direction as in Figure 3.4 (c) and (g). Only one communication step per color is required to supply the blocks $\mathcal{B}_i$ with the latest information of their neighbors. This is ensured by building 2-by-2 block compounds and choosing for each compound the same circle-shaped color alignment. In Figure 3.4, each sub figure consists of two such compounds.

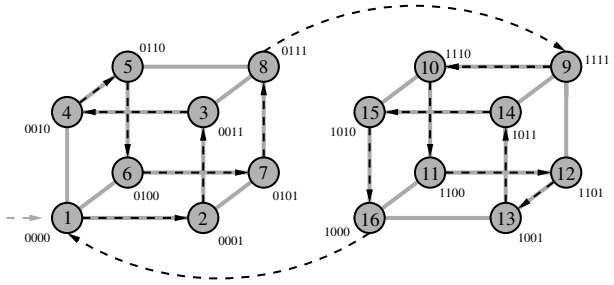

Figure 3.5: 16 color block alignment in 4D for a single 2-by-2-by-2-by-2 compound, each vertex represents a Schwarz block $\mathcal{B}_i$.

Going from 2D to 4D, the 2-by-2 compounds turn into 2-by-2-by-2-by-2 compounds of 16 blocks. Preserving the properties of the 2D approach thus requires a loop that visits all 16 blocks exactly once. This ensures that each block has the latest information with just one ghost cell update per color. One possible loop is sketched in Figure 3.5, the corresponding order is given by the decimal numbers and arrows. The number of ghost cell updates to be performed per iteration is 16 and thus the same as for two color SAP, but the 16 color approach spreads information faster on the lattice. The resulting Algorithm 4 can be formulated just by introducing 16 instead of two colors to Algorithm 3.

Algorithm 4: Sixteen color Schwarz

input: ψ, η, ν
output: ψ
for $k = 1$ to ν
  for $q = 1$ to 16
    $r \leftarrow \eta - D\psi$
    for all i with color q **do**
      $\psi \leftarrow \psi + B_i r$

The binary numbers in Figure 3.5 number the vertices in canonical order. Given a position on the loop, the binary number corresponding to the subsequent position differs only by a single digit. This type of sequence of binary numbers is called gray code (see [57, 114]), and when proceeding by one vertex on the loop, the digit that flips gives the communication direction. In a four digit binary number $d = d_3d_2d_1d_0$, d_0 corresponds to the x-direction, d_1 to y, d_2 to z and d_3 to t. A flip from 0 to 1 in a digit indicates that an update for the positive boundary in the corresponding direction is required, a flip from 1 to 0 requires the negative boundary to be updated. Thus, any loop together with its communication scheme corresponds to a 4 bit gray code.

It remains to validate the statement that any loop through the 16 blocks, that

does not visit any of them more than once, is sufficient. This can be done by understanding the following facts: Solving a block system for a block $\mathcal{B}_j$ provides an update to four neighboring blocks, exactly one neighboring block either in direction $+\mu$ or $-\mu$. Without loss of generality, let the direction be $+\mu$ for all μ. Before we can solve the block systems corresponding to these neighboring blocks, a block, say $\mathcal{B}_{j'}$, having $\mathcal{B}_j$ as neighbor in direction $-\mu$, needs a ghost cell update in direction $+\mu$. Figure 3.5 illustrates that this ghost cell update will be performed when visiting a vertex pointing into direction $+\mu$, but any path from $\mathcal{B}_j$ to $\mathcal{B}_{j'}$, not visiting any block more than once, contains at least one vertex pointing into direction $+\mu$. Thus, the required ghost cell update is performed before the block system corresponding to $\mathcal{B}_{j'}$ is solved.

These facts validate the statement for 16 blocks. Clearly it then holds for multiples of 16 blocks since all blocks of a given color are processed successively and having neighboring blocks on the same processor with up-to-date information is as good as having up-to-date ghost cells.

All of the introduced Schwarz methods solve each block system only once per iteration k. In terms of complexity, the only difference is the number of residual updates per iteration. If the lattice $\mathcal{L}$ is decomposed into s blocks $\mathcal{B}_1, \ldots, \mathcal{B}_s$, the sequential multiplicative Schwarz method can be seen as an s color multiplicative method. Thus, the number of residual updates is given by the number of colors. Though, from the implementational point of view, the complexity does not necessarily depend on the number of colors. The residual update can be performed block-wise right before solving the respective block system. Since each block system is solved exactly once per iteration, the complexity of all block-wise residual updates can be summed up to that of one global update, yielding the following proposition.

Proposition 3.4
For a fixed block size, i.e., the number of blocks s being proportional to the system size n, one iteration of any of the introduced Schwarz methods has linear complexity

$$\mathcal{O}(n)\,,$$

and this cost is independent of the number of colors and corresponding residual updates.

3.4 Implementation Details – The Wilson-Dirac Operator

Most of the computational work for iteratively solving (2.8), e.g., with Krylov subspace methods such as CG, GMRES or GCR (cf. [100]) is caused by matrix-vector multiplications (MVMs) with the Wilson-Dirac operator. Therefore, the lattice QCD community has spent large effort on optimizing the MVM with this

operator for several supercomputing platforms. We decided to use the scheme from [71] which focuses on the overlap of communication and computation. Implementation details for low level optimization like `SSE` or `AVX` are not considered in this thesis.

To explain the evaluation and communication scheme, we introduce the following definitions.

Definition 3.5
Let $\mathcal{P}_i$ be a process block decomposition. The corresponding set of all ghost cell lattice sites *is given by*

$$\partial\mathcal{P}_i = \{x \in \mathcal{L} - \mathcal{P}_i : \exists \mu \in \{0,1,2,3\} \text{ such that } x + \hat{\mu} \in \mathcal{P}_i \text{ or } x - \hat{\mu} \in \mathcal{P}_i\}$$

and can also be seen as the outer boundary *of $\mathcal{P}_i$. More precisely, we define the particular* ghost cells/outer boundaries

$$\partial_\mu^\pm \mathcal{P}_i := \{x \in \partial\mathcal{P}_i \,:\, x \mp \hat{\mu} \in \mathcal{P}_i\}\,.$$

Similarly we define inner boundaries

$$d_\mu^\pm \mathcal{P}_i := \{x \in \mathcal{P}_i \,:\, x \pm \hat{\mu} \in \partial_\mu^\pm \mathcal{P}_i\}\,.$$

For a given quark field ψ we define the inner and outer boundaries of ψ by

$$\partial_\mu^\pm \mathcal{P}_i(\psi) = \{\psi(x) : x \in \partial_\mu^\pm \mathcal{P}_i\} \quad \text{and} \quad d_\mu^\pm \mathcal{P}_i(\psi) = \{\psi(x) : x \in d_\mu^\pm \mathcal{P}_i\}\,.$$

If the index i of $\mathcal{P}_i$ is not of particular interest, we use the shorthands

$$\partial_\mu^\pm \psi \quad \text{and} \quad d_\mu^\pm \psi\,.$$

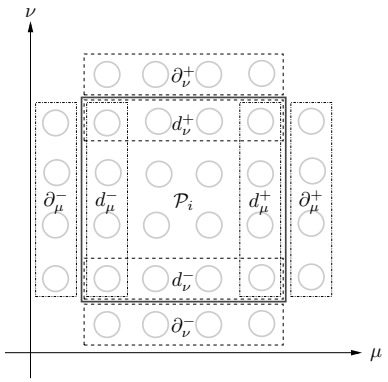

Figure 3.6: Illustration of outer and inner boundaries in Definition 3.5.

For an illustration of Definition 3.5, see Figure 3.6. If $\mathcal{P}_j$ is the neighbor of $\mathcal{P}_i$ in positive μ direction, an update of $\partial_\mu^+ \mathcal{P}_i(\psi)$ ensures

$$\partial_\mu^+ \mathcal{P}_i(\psi) = d_\mu^- \mathcal{P}_j(\psi)\,,$$

Algorithm 5: Evaluation of the Wilson-Dirac operator D_W

input: ψ, η
output: $\eta = \eta + D_W\psi$

```
1  for μ = 0 to 3
2      for all x ∈ P_i do
3          "λ_μ(x) ← [(I_4 − γ_μ)/2 ⊗ I_3]ψ(x)"
4  for μ = 0 to 3
5      start communicating d_μ^− λ_μ in negative μ direction
6  for μ = 0 to 3
7      for all x ∈ P_i do
8          "χ_μ(x + μ̂) ← [(I_4 + γ_μ)/2 ⊗ U_μ^H(x)]ψ(x)"
9  for μ = 0 to 3
10     start communicating ∂_μ^+ χ_μ in positive μ direction
11 for μ = 0 to 3
12     wait for receiving ∂_μ^+ λ_μ
13 for μ = 0 to 3
14     for all x ∈ P_i do
15         "ψ(x) ← ψ(x) + [I_4 ⊗ U_μ(x)]λ_μ(x + μ̂)"
16 for μ = 0 to 3
17     wait for receiving d_μ^− χ_μ
18 for μ = 0 to 3
19     for all x ∈ P_i do
20         "ψ(x) ← ψ(x) + χ_μ(x)"
```

and similarly for the negative ghost cells of ψ.

Now, recalling the notation introduced in Section 2.2, we can formulate the evaluation scheme for D_W from [71] in Algorithm 5. Besides hiding communication, this approach avoids the need to store gauge links for the ghost cells. Hence, the set of gauge links required on process $\mathcal{P}_i$ is given by

$$\{U_\mu(x) \,:\, x \in \mathcal{P}_i,\ \mu = 0, 1, 2, 3\}$$

and is thus scaling with the process lattice size.

In a parallel environment, neighbor coupling causes the need of communication. For communicating in different directions and computing the couplings in different directions at the same time, we introduce buffers λ_μ and χ_μ. On process $\mathcal{P}_i$, line 3 computes $d_\mu^- \lambda_\mu$ which is needed by the neighbor in negative μ direction, say $\mathcal{P}_j$, to compute

$$(I_4 - \gamma_\mu)/2 \otimes U_\mu(x)\psi(x + \hat{\mu}) \quad \text{for} \quad x + \hat{\mu} \in \partial_\mu^+ \mathcal{P}_j$$

in line 15. Hence, every process communicates $d_\mu^- \lambda_\mu$ in negative μ direction and in return obtains the contributions to $d_\mu^+ \psi$ coming from $\partial_\mu^+ \psi$. Afterwards, all contributions from positive μ directions are available in all lattice sites x on any process $\mathcal{P}_i$, and the gauge matrices $U_\mu(x)$ can be multiplied with $\lambda_\mu(x)$ and added to the result as shown in line 15. Since the matrices $(I_4 \pm \gamma_\mu)/2$ are projections on $\mathbb{C}^4$ with rank 2, we only need to perform two multiplications with $U_\mu(x)$. The four component contribution to $\psi(x)$ in terms of spin variables is then computed as linear combinations of the two results from the multiplications with $U_\mu(x)$. Therefore, the buffers λ_μ and χ_μ can be computed and stored in projected form, only requiring 6 complex numbers per lattice site. This is why computations involving λ_μ or χ_μ are displayed in quotation marks since they are carried out in a compressed form and to be understood symbolically.

In the same manner, line 8 computes $\partial_\mu^+ \chi_\mu(x)$, but the contributions to $d_\mu^- \psi$ coming from $\partial_\mu^- \psi$ are missing. Therefore, $\partial_\mu^+ \chi_\mu(x)$ is communicated in positive μ direction. As the only difference, the gauge matrices $U_\mu^H(x)$ are multiplied with $d_\mu^+ \psi$ before communicating since $\mathcal{P}_i$ owns the gauge matrices pointing to the ghost cells $\partial_\mu^+ \mathcal{P}_i$.

The procedures for both directions are independent from each other and merged together in order to overlap communication and computation.

Since D_W incorporates only the neighbor coupling, the clover term inducing the self coupling can be treated separately. For the γ-matrix representation from Section 2.2, the clover term consists of two hermitian 6-by-6 matrices per lattice site which can be computed once in the very beginning. Due to hermiticity, only the upper triangle of each matrix needs to be stored. The application of the clover term then just consists of two multiplications with 6-by-6 matrices per lattice site.

3.4.1 Odd-even preconditioning

A commonly used technique in lattice QCD computations is odd-even preconditioning. A lattice site $x = (x_1, x_2, x_3, x_4)$ is called *even* if $x_1 + x_2 + x_3 + x_4$ is even, else it is called *odd*. The Wilson-Dirac operator has the form

$$D = \begin{pmatrix} D_{ee} & D_{eo} \\ D_{oe} & D_{oo} \end{pmatrix},$$

if we order all even sites first. Due to the nearest neighbor coupling, D_{ee} and D_{oo} are block diagonal with 6×6 diagonal blocks, they only contain the clover term. The inverse of D is then given by

$$D^{-1} = \begin{pmatrix} I & 0 \\ -D_{oo}^{-1} D_{oe} & I \end{pmatrix} \begin{pmatrix} D_S^{-1} & 0 \\ 0 & D_{oo}^{-1} \end{pmatrix} \begin{pmatrix} I & -D_{eo} D_{oo}^{-1} \\ 0 & I \end{pmatrix} \tag{3.7}$$

with the Schur complement

$$D_S = D_{ee} - D_{eo} D_{oo}^{-1} D_{oe}\,.$$

Instead of solving a system with D we can solve the corresponding system for the even lattice sites given by the Schur complement D_S and then retrieve the solution at the odd lattice sites. This is done by applying D^{-1} in the form of (3.7) to a (odd-even ordered) right hand side

$$\eta = \begin{pmatrix} \eta_e \\ \eta_o \end{pmatrix} ,$$

i.e., by computing

$$\psi_e = D_S^{-1}(\eta_e - D_{eo}D_{oo}^{-1}\eta_o)$$

with an iterative solver for D_S and thereafter computing

$$\psi_o = D_{oo}^{-1}\eta_o - D_{oo}^{-1}D_{oe}\psi_e \,. \tag{3.8}$$

Since D_{oo} is 6×6 block diagonal, the inverse D_{oo}^{-1} is pre-computed once, and the operator D_S is applied in factorized form. A matrix-vector multiplication with D_S thus requires the same work as one with D while the condition of D_S improves over that of D. Typically, this results in a gain of 2–3 in the number of iterations and execution time. Odd-even preconditioning is well-established in lattice QCD, see, e.g., [78, 92].

We implemented odd-even preconditioning in all unpreconditioned Krylov subspace methods and in the solver for the Schwarz blocks.

3.5 Implementation Details – Schwarz Methods

Based on the basic concepts of Schwarz methods and their parallelization introduced in the previous sections we now discuss implementation details. Therein we mostly focus on how to avoid redundant computations and, as a side aspect, how to overlap communication and computation. Details of deeper matter as computation of index arrays for block orders or data layouts are not discussed here. Similar concepts for Schwarz methods have already been implemented in [75, 76].

For the notation used in the following definition, recall Definition 3.1.

Definition 3.6
Consider a block decomposition $\mathcal{B}_i$ of the lattice $\mathcal{L}$ with corresponding variable blocks $\mathcal{V}_i = \mathcal{B}_i \times \mathcal{C} \times \mathcal{S}$. For any vector $v \in \mathcal{V} = \mathcal{L} \times \mathcal{C} \times \mathcal{S}$, $v|_{\mathcal{V}_i}$ denotes the restriction of v to $\mathcal{V}_i$. For a lattice-block $\mathcal{B}_i$, the outer boundary $\partial\mathcal{B}_i \subset \mathcal{L} - \mathcal{B}_i$ *is the set of all lattice sites which are* linked *but not belonging to $\mathcal{B}_i$, i.e.,*

$$\partial\mathcal{B}_i := \{x \notin \mathcal{B}_i \,:\, \exists y \in \mathcal{B}_i, \exists \mu \in \{0,1,2,3\} \text{ such that } x = y + \hat{\mu} \text{ or } x = y - \hat{\mu}\} \,.$$

Accordingly we define the outer boundary of a variable block

$$\partial\mathcal{V}_i := \partial\mathcal{B}_i \times \mathcal{C} \times \mathcal{S} \,.$$

The restriction of the clover improved Wilson-Dirac operator D to the links interconnecting $\partial\mathcal{V}_i$ and $\mathcal{V}_i$ is denoted by $D_{\partial\mathcal{V}_i}^{\mathcal{V}_i}$, i.e., $D_{\partial\mathcal{V}_i}^{\mathcal{V}_i} = \mathcal{I}_{V_i}^H D \mathcal{I}_{\partial V_i}$ with the respective trivial embeddings $\mathcal{I}_{V_i}^H$, $\mathcal{I}_{\partial V_i}$ from Definition 3.2.

The restriction of D acting only within the block $\mathcal{V}_i$, i.e., D_i (from Definition 3.2), is a map from $\mathcal{V}_i$ to $\mathcal{V}_i$ itself while $D_{\partial\mathcal{V}_i}^{\mathcal{V}_i}$ contains all contributions of D from outside of $\mathcal{V}_i$ to $\mathcal{V}_i$. Having these two operators, a local variant of a residual update $r \leftarrow r - D\phi$, where ϕ denotes the increment for an update for the current iterate (i.e., $\psi \leftarrow \psi + \phi$), can be performed via

$$r|_{\mathcal{V}_i} \leftarrow r|_{\mathcal{V}_i} - (D_i\phi|_{\mathcal{V}_i} + D_{\partial\mathcal{V}_i}^{\mathcal{V}_i}\phi|_{\partial\mathcal{V}_i}) \,. \tag{3.9}$$

For Schwarz methods it is highly desirable to do all residual updates locally on demand before solving a block system $D_i\phi|_{\mathcal{V}_i} = r|_{\mathcal{V}_i}$ since D, r and ϕ are then streamed through the cache only once per Schwarz iteration. On top of that, when performing the residual update globally on $\mathcal{V}$, c colors would cause c global residual updates which from the block-based point of view means $c-1$ redundant computation of all block residuals $r|_{\mathcal{V}_i}$.

As far as local residual updates are concerned, any of the previously introduced multiplicative Schwarz methods can be formulated as follows: For all blocks, compute the current block residual and solve the block system. The only difference now lies in the order of the blocks in which the Schwarz iteration proceeds. E.g., for the red-black approach, the red blocks are ordered first and the black blocks thereafter (or vice versa). Then computations on blocks of the same color do not influence each other since D does not couple variables from different blocks. Thus, for the outcome it makes no difference whether we compute the whole red residual first or block-wise on demand.

Our numerical testing showed that the fastest way of approximating a block system solution is the minimal residual (MR) iteration [100] which is mathematically equivalent to restarted GMRES with restart length 1. For a fast computation, a block-wise data layout, such that each block is coherent in storage, is advisable. This applies to the vectors r, ψ, ϕ and also to the configuration U. The ghost cells for the communication may be added, e.g., after all the blocks at the rear end of the vectors. Algorithm 6 describes the MR iteration for the block system case.

One Schwarz iteration calls the MR iteration for each block $\mathcal{V}_i$ with the current residual $r|_{\mathcal{V}_i}$. On the block $\mathcal{V}_i$, the residual part of the block restricted system is kept up to date by line 6 during the whole MR iteration. Summarizing, it provides an update $\phi|_{\mathcal{V}_i}$ for the current iterate ψ and performs a residual update

$$r|_{\mathcal{V}_i} \leftarrow r|_{\mathcal{V}_i} - D_i\phi|_{\mathcal{V}_i} \,. \tag{3.10}$$

Hence, "half" of the residual update from (3.9) is already performed within the MR iteration, and it is sufficient to compute just the contributions to the block

Algorithm 6: Minimal residual iteration (MR) on an SAP block $\mathcal{V}_i$

input: $r|_{\mathcal{V}_i}$
output: $\phi|_{\mathcal{V}_i}$, $r|_{\mathcal{V}_i}$
1 $\phi|_{\mathcal{V}_i} \leftarrow 0$
2 **for** $k = 1$ to *maxiter*
3 $\quad z \leftarrow D_i r|_{\mathcal{V}_i}$
4 $\quad \alpha \leftarrow z^H r|_{\mathcal{V}_i} / (z^H z)$
5 $\quad \phi|_{\mathcal{V}_i} \leftarrow \phi|_{\mathcal{V}_i} + \alpha r|_{\mathcal{V}_i}$
6 $\quad r|_{\mathcal{V}_i} \leftarrow r|_{\mathcal{V}_i} - \alpha z$

residual $r|_{\mathcal{V}_i}$ induced by the outer boundaries of the update ϕ via

$$r|_{\mathcal{V}_i} \leftarrow r|_{\mathcal{V}_i} - D^{\mathcal{V}_i}_{\partial\mathcal{V}_i} \phi|_{\partial\mathcal{V}_i} \tag{3.11}$$

before starting the MR iteration on this block again. This is particularly interesting when performing more than one Schwarz iteration consecutively, or when the initial residual r is globally up to date, e.g., when the initial guess ψ is zero. A description for the case of the initial guess being zero can be found in Algorithm 7.

In the first iteration $k = 1$ for the first *color* of Algorithm 7, the residual update could be omitted completely, since the entire residual is already known. For the subsequent colors within the first iteration $k = 1$, the residual contributions of the blocks themselves via D_i are still up to date. Thus, a residual update via (3.11) is sufficient. Using a few `if`-cascades and computations of (3.10), Algorithm 7 can be generalized for the case of $\psi \neq 0$, still avoiding global residual updates.

Algorithm 7: SAP in parallel (for one process holding several blocks $\mathcal{V}_i$)

input: η
output: ψ
1 $\psi \leftarrow 0,\ \phi \leftarrow 0,\ r \leftarrow \eta$
2 **for** $k = 1$ to *maxcycles*
3 $\quad$ **for** *color* $= 1$ to *numcolors*
4 $\quad\quad$ start update of ghost cells of ϕ {`communication only`}
5 $\quad\quad$ **for all** blocks $\mathcal{V}_i$ of current color *not adjacent* to ghost cells **do**
6 $\quad\quad\quad$ update residual via (3.11)
7 $\quad\quad\quad$ $[\phi|_{\mathcal{V}_i}, r|_{\mathcal{V}_i}] \leftarrow \mathrm{MR}(r|_{\mathcal{V}_i})$
8 $\quad\quad\quad$ $\psi \leftarrow \psi + \phi|_{\mathcal{V}_i}$
9 $\quad\quad$ wait for update of ghost cells of ϕ to be finished
10 $\quad\quad$ **for all** blocks $\mathcal{V}_i$ of current color *adjacent* to ghost cells **do**
11 $\quad\quad\quad$ perform lines 6 to 8

Algorithm 7 also describes an ansatz for implementing the parallelization. For every color the ghost cells are updated once in the beginning (cf. Figure 3.1). Note that the ghost update can be restricted to the contributions from the previous color in order to reduce the amount of data to be communicated. While the communication proceeds, the inner block systems of the current color which are not adjacent to ghost cells on the lattice can be computed. After completion of the communication, all block systems of the current color adjacent to ghost cells can be computed.

This implementation of a communication scheme can also be used for the additive approach. Therein, the recent block restricted residual updates coming from the MR iteration have to be buffered until the current iteration is finished, thus enforcing that all initial local residuals in line 6 of Algorithm 7 for the MR iteration are computed from information of previous iterations. For the 16 color approach the algorithm is also valid, but in order to benefit from all its advantages, one has to specify for every color which part of the ghost cells has to be updated (cf. Figure 3.4 and Section 3.3). Then, lines 6 to 8 can be performed for all blocks which are not adjacent to the ghost cells currently updated, i.e., some of the work from line 11 can already be done in lines 6 to 8 during the communication.

When SAP is used as a right preconditioner for, e.g., flexible GMRES (FGMRES, cf. [100]), the corresponding Krylov subspace is built for

$$DM_{SAP}\,.$$

In case that the multiplications with D and M_{SAP} are performed either both in single or both in double precision, the result of the multiplication with D can be obtained at a lower computational cost. After the SAP iteration, the current residual r can be computed by performing (3.11) for $color = 1$ to $numcolors - 1$. For red-black SAP this corresponds to performing (3.11) only for half of the blocks. Then, the multiplication of D with the solution ψ from the SAP iteration can be obtained at a lower computational cost by using the relation

$$D\psi = \eta - r\,.$$

In Section 8.4.6 we make use of this feature within a mixed precision FGMRES where the matrix-vector multiplications are performed in single precision except when computing the residual for the restart which is performed in double precision. All inner products are performed in double precision without exception. A similar mixed precision variant of (restarted) GCR is used in [75]. All other results were computed with an implementation that performs the domain decomposition method in single precision but the preconditioned FGMRES in double precision. This means that we do not use mixed precision FGMRES outside of Section 8.4.6.

3.5.1 Odd-even preconditioning

When using Schwarz methods it is common to solve the block systems

$$\eta|_{\mathcal{V}_i} = D_i\psi|_{\mathcal{V}_i}$$

with an odd-even preconditioned (cf. Section 3.4.1) version of the MR iteration (see Algorithm 6). Even though the odd-even preconditioned MR iteration only provides a residual update of half length we are still able to retrieve an entire residual update on the whole block from this information. Thus, all computational advantages from combining Algorithms 6 and 7 are preserved.

For the sake of simplicity we drop the block subscript $\mathcal{V}_i$ in this section. Considering a residual r on a whole block, i.e., for the even and the odd sites

$$r = \eta - D\psi = \begin{pmatrix} \eta_e - D_{ee}\psi_e - D_{eo}\psi_o \\ \eta_o - D_{oe}\psi_e - D_{oo}\psi_o \end{pmatrix}$$

we obtain $r_o = 0$ if we take

$$\psi_o = D_{oo}^{-1}(\eta_o - D_{oe}\psi_e)$$

as in (3.8). This is independent of the accuracy of ψ_e. Hence, the residual r on the odd lattice sites never needs to be updated. On the even sites the MR iteration approximates a solution ψ_e of

$$\eta_e - D_{eo}D_{oo}^{-1}\eta_o = (D_{ee} - D_{eo}D_{oo}^{-1}D_{oe})\psi_e$$

with a corresponding residual $\tilde{r}_e$. Therefore, using (3.7) we have

$$\begin{aligned} \tilde{r}_e &= \eta_e - D_{eo}D_{oo}^{-1}\eta_o - (D_{ee} - D_{eo}D_{oo}^{-1}D_{oe})\psi_e \\ &= \eta_e - D_{ee}\psi_e - D_{eo}D_{oo}^{-1}(\eta_o - D_{oe}\psi_e) \\ &= \eta_e - D_{ee}\psi_e - D_{eo}\psi_o \\ &= r_e\,. \end{aligned}$$

Thus, the residual update obtained by the MR iteration yields the correct residual update for the whole Schwarz block.

3.6 Numerical Results

In this section we show numerical results for three types of parallel Schwarz methods, the additive, the red-black and the 16 color Schwarz method previously discussed in the Sections 3.1, 3.2 and 3.3, respectively. Typically, Schwarz methods are used as preconditioners to Krylov subspace methods [50, 78].

Within our implementation, all Schwarz methods solve the block systems iteratively using an odd-even preconditioned MR iteration (see Section 3.5). This

yields a non-stationary preconditioner and thus the preconditioned Krylov subspace method has to be flexible. We therefore employed the Schwarz methods as preconditioners to FGMRES.

We implemented the Schwarz methods and Krylov subspace methods in the programming language `C` using the parallelization interface of `MPI`. The MVM with D is realized as in Section 3.4. Our code is optimized to a certain extent, but certainly not to the extreme: As is customary in lattice QCD computations, we use a mixed precision approach where the preconditioning is done in single precision. Low level optimization (e.g., making use of the `SSE`-registers on Intel/AMD architectures) has not been considered in this section. All Krylov subspace methods (FGMRES, BiCGStab, CG) have been implemented in a common framework with the same degree of optimization to allow for a standardized comparison of compute times.

In each Schwarz method we used block size 4^4, solving each block system with 4 iterations of odd-even MR. The relative residual tolerance for FGMRES is 10^{-10} and we denote the number of Schwarz iterations in each FGMRES iteration with ν.

ID	lattice size $N_t \times N_s^3$	pion mass m_π [MeV]	CGNR iterations	shift m_0	clover term c_{sw}	provided by
1	64×32^3	300	11,370	-0.04630	1.00000	BMW-c [39, 40]

Table 3.1: Configurations used together with their parameters. For details about their generation we refer to the references. Pion mass rounded to steps of 5 MeV.

In our numerical experiments we observed that the ratios between the different Schwarz methods and Krylov subspace methods are almost independent of the configurations used. Thus, we decided that using a single configuration is sufficient for the purpose of this section. This configuration and its parameters are listed in Table 3.1. In principle, the pion mass m_π and the lattice spacing (not listed) determine the condition of the respective operator, e.g., the smaller m_π, the more ill-conditioned the respective operator is. The physical pion mass is $m_{\pi_{phys}} = 135$ MeV.

All results were obtained on the Juropa machine at Jülich Supercomputing Centre (JSC), a cluster with 2,208 compute nodes, each with two Intel Xeon X5570 (Nehalem-EP) quad-core processors [65, 66]. This machine provides a maximum of 8,192 cores for a single job. Unless stated otherwise, we used the `icc`-compiler with the optimization flags `-O3`, `-ipo`, `-axSSE4.2` and `-m64`.

3.6.1 Comparison of Schwarz Methods

First, we compare different Schwarz methods with different numbers of cycles ν per FGMRES iteration and two different parallelizations on configuration ID 1,

i.e., a rather well conditioned medium sized 64×32^3 lattice.

	FGMRES + Schwarz		
	additive	red-black	16 colors
	iteration counts		
$\nu = 1$	1,229	635	609
$\nu = 3$	536	272	250
$\nu = 5$	345	173	164
	timings for 32 processes		
$\nu = 1$	523s	287s	293s
$\nu = 3$	454s	242s	233s
$\nu = 5$	438s	230s	226s
	timings for 256 processes		
$\nu = 1$	67.3s	37.3s	38.4s
$\nu = 3$	58.7s	31.9s	30.9s
$\nu = 5$	56.8s	30.4s	30.0s

Table 3.2: Comparison of FGMRES preconditioned with additive, red-black and 16 color Schwarz for two different parallelizations on a 64×32^3 lattice (Table 3.1: ID 1).

Table 3.2 shows that the red-black approach requires almost a factor of two less iterations per solve, independently of the number of cycles ν. This carries over to the runtimes for both parallelizations. In terms of iteration counts, the 16 color approach yields a small advantage over the red black approach. Up to minor losses this also carries over to the runtimes, except for $\nu = 1$. For both parallelizations the 16 color approach performs worse in this case since in the first cycle of a Schwarz method, work can be saved for the first color if the initial guess is zero. This is exactly the case since in every FGMRES iteration k the respective Schwarz method is applied to the k-th Krylov subspace basis vector v_k as right hand side with initial guess zero. Therefore, in Algorithm 7 the residual updates and the communication for the first color can be omitted. For the two color approach this means that half of the communication and half of the residual calculations can be skipped while for the 16 color approach skipping communication and residual calculation for the first color saves 1/16 only. This is reflected by the runtimes.

For the cases $\nu = 3$ and $\nu = 5$, the gain of the 16 color approach over SAP is about 1–2% in runtime which does not necessarily justify the additional coding effort. Thus, we do not consider using the 16 color approach in other numerical tests.

3.6.2 Comparison with Krylov Subspace Methods

We now compare the most effective red-black Schwarz methods of the previous tests with different Krylov subspace methods, i.e., GMRES(20), CG on the normal equation with the residual $r = \eta - D\psi$ (CNGR) and ultimately with an odd-even preconditioned mixed precision approach of BiCGStab (cf. [100]) which in the physics community is considered to perform best when no pre-computed information is used as in deflation or multigrid methods. In our experiments, larger restart values for GMRES did not lead to significantly different solve times.

FGMRES+SAP(5)	GMRES	CGNR	mp-oe-BiCGStab
iteration counts			
173	4,175	11,370	1,074
timings for 32 processes			
230s	882s	2,928s	189s
timings for 256 processes			
30.4s	113s	392s	27.2s

Table 3.3: Comparison of FGMRES preconditioned with red-black Schwarz with different Krylov subspace methods for two different parallelizations on a 64×32^3 lattice (Table 3.1: configuration ID 1).

The results reported in Table 3.3 show that preconditioning GMRES yields a factor of more than 3 in runtime for both degrees of parallelizations. CGNR performs worst of all stated methods and BiCGStab performs best although there is only a small difference between BiCGStab and SAP-preconditioned FGMRES (FGMRES+SAP). Depending on architecture and parallelization, FGMRES+SAP performs better when all-to-all communication is expensive. FGMRES+SAP, GMRES and CGNR require two all-to-all communications (inner products) and BiCGStab even requires 6 all-to-all communications per iteration. We profiled the all-to-all communication and observed that for BiCGStab it consumed already 5.3 out of 27.2 seconds when using 256 processes but only 1.9 out of 189 seconds for 32 processes. Thus, the speed-up factor for 8 times more cores is only 6.9 whereas we obtain a factor of 7.6 when using FGMRES+SAP. For GMRES we observe even a factor of 7.8 which might due to cache effects as we will see in the strong scaling plot Figure 3.8 in the next section.

3.6.3 Scaling Tests

Compared to Krylov subspace methods, Schwarz methods do not require all-to-all communications and they have a higher arithmetic intensity, i.e., the flop count per lattice site and iteration (and thus per communication step) is much higher. While Krylov subspace methods are working with vector operations and matrix-vector multiplications possibly streaming large vectors through the cache, Schwarz

methods are dealing with small block systems and small vector operations. Thus, we can expect that Schwarz methods perform better for large parallelizations when in particular all-to-all communications are expensive, but also for small parallelizations, i.e., large local lattices, when vectors and configurations do not fit into cache anymore.

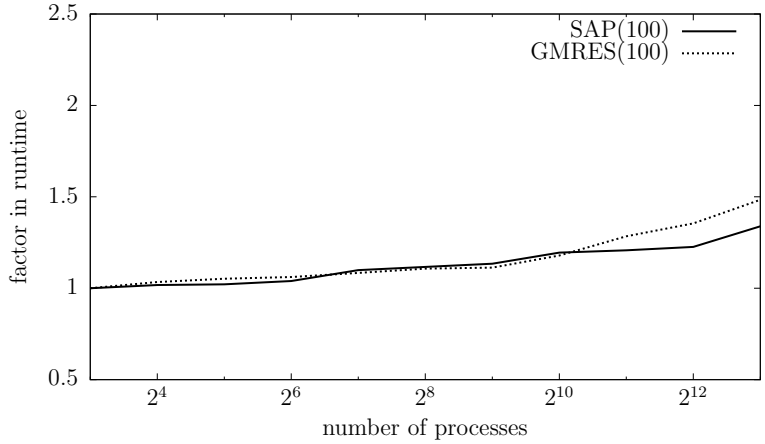

Figure 3.7: Weak scaling of 100 iterations of SAP and of 100 iterations of GMRES for a 8×4^3 local lattice.

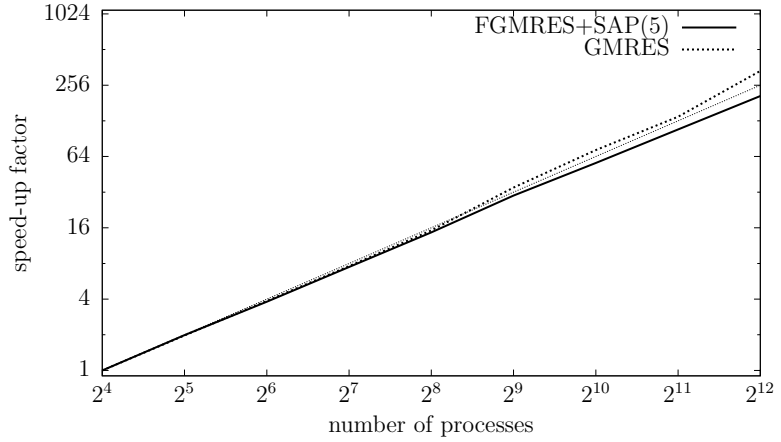

Figure 3.8: Strong scaling of FGMRES preconditioned with 5 iterations of SAP and of pure GMRES using a 64×32^3 lattice (Table 3.1: configuration ID 1). The thin, dashed, diagonal line displays optimal scaling.

For a comparison of the weak scaling of SAP and GMRES, we employed 100 iterations of either method for lattice sizes ranging from 8^4 to $64^2 \times 32^2$ with a process lattice size of 8×4^3. We used configurations where each gauge link was taken as the QR factor of a 3-by-3 matrix with uniformly distributed random

entries in

$$\{a + \mathrm{i}b : -1 \leq a, b \leq 1\}\,.$$

This may produce matrices with a negative determinant which is not an issue here since we apply a fixed number of iterations instead of solving to any chosen accuracy. Furthermore, we took the best timings out of 10 runs for every measurement, respectively. Figure 3.7 displays the gain in runtime as a function of the number of processes. Until 2^{10} processes, both methods scale equally. Profiling shows that beyond 2^{10} processes the time spent in all-to-all communications starts influencing the overall runtime in GMRES, so that SAP scales better. For this weak scaling test our SAP implementation, in contrast to our GMRES implementation, is not able to hide nearest neighbor communication since a process owns only one 4^4 Schwarz block of each color. Even when we employ employing 4,096 processes on Juropa, nearest neighbor communication is not an issue, neither for SAP nor for GMRES. Only when using 8,192 cores we see a slight dependence on nearest neighbor communication.

Figure 3.8 shows the strong scaling of FGMRES+SAP(5) compared to the strong scaling of GMRES, again taking the best timings out of 10 runs for every measurement. The thin dotted diagonal line represents the desirable optimal scaling. The number of processes ranges from 16 to 4,096. Both methods show almost perfect scaling up to 512 processes. Beyond this number, the runtime of GMRES decreases by a bit more than a factor of two when doubling the number of processes. This indicates that GMRES is sensitive to cache effects, i.e., for smaller parallelizations the large vectors cannot be streamed through the cache efficiently. The increased communication cost (cf. Figure 3.7) are more than balanced by these cache effects. For FGMRES+SAP(5) the runtime is decreased by a bit less than a factor of two when doubling the number of processes. Cache effects do not seem to be an issue as SAP benefits from its higher arithmetic intensity. We also profiled the communication in this case and found that this minor slow down is caused by all-to-all communication and nearest neighbor communication to the same extent. For the range of processes on the Juropa machine, the difference between both methods is only marginal and it cannot be said that one method scales better than the other one. Additional and also more promising results for SAP using a different implementation in combination with a different machine can be found in [78].

Chapter 4

Algebraic Multigrid

Any multigrid method consists of two components, namely a smoother and a coarse grid correction [25, 59, 98, 110]. Typically, the smoother can be chosen as a simple iterative method. This can be a relaxation scheme like Jacobi, Gauss-Seidel or their block variants (which are equivalent to additive and multiplicative Schwarz, respectively) as well as Krylov subspace methods. Given the properties of SAP presented in the previous chapter, we choose SAP as our smoothing scheme in the QCD context.

Except for implementation details given in Section 4.4, this chapter is largely based on [49].

Let us reserve the term *near kernel* for the space spanned by the eigenvectors belonging to small (in modulus) eigenvalues of D. Since SAP is not able to sufficiently remove error components belonging to the near kernel (cf. Figure 3.3), the multigrid method treats these persistent error components separately in a smaller subspace with n_c variables. Thus, this subspace should approximate the near kernel well. The typical algebraic multigrid setup is then as follows: We have to find an operator D_c which resembles D on that subspace both in the sense that it acts on the near kernel in a similar manner as D does, but also in terms of the connection structure and sparsity. The latter allows to work on D_c recursively using the same approach, thus going from two-grid to true multigrid.

Definition 4.1
Let $n = 12n_{\mathcal{L}}$ and $n_c < n$. Considering full rank linear restriction *and* prolongation/interpolation *operators*

$$\begin{aligned} R: \ \mathbb{C}^n &\rightarrow \mathbb{C}^{n_c}, \\ P: \ \mathbb{C}^{n_c} &\rightarrow \mathbb{C}^n \end{aligned}$$

we define a Petrov-Galerkin projection of D, i.e., the coarse grid operator

$$D_c := RDP$$

and the corresponding coarse grid correction

$$\psi \leftarrow \psi + PD_c^{-1}Rr$$

with $r = \eta - D\psi$.

The map R restricts information from the original space to the subspace and P transports information back to the original space. The coarse grid correction for a current iterate ψ restricts the current residual r via R to the subspace, there solving

$$D_c e_c = Rr \tag{4.1}$$

and transporting the coarse error e_c via P back to the original space as a correction for ψ. As in (3.6), one step of coarse grid correction can be summarized as

$$\psi \leftarrow (I - PD_c^{-1}RD)\psi + PD_c^{-1}R\eta$$

and the corresponding error propagator is given by

$$I - PD_c^{-1}RD$$

which is a projection onto $\text{range}(RD)^\perp$ along $\text{range}(P)$. The action of the coarse grid correction is thus complementary to that of the smoother if $\text{range}(P)$ approximately contains the near kernel and $\text{range}(RD)^\perp$ approximately contains the remaining complementary eigenvectors (which are then efficiently reduced by the smoother). The latter condition is satisfied if $\text{range}(R) = \text{range}(RD)$ approximately contains the *left* eigenvectors corresponding to the small eigenvalues. This can be seen by looking at exact eigenvectors: Since left and right eigenvectors are mutually orthogonal, if $\text{range}(R) = \text{range}(RD)$ is spanned by left eigenvectors of D, then $\text{range}(R)^\perp$ is spanned by the complementary right eigenvectors of D.

Once D_c is found, a basic two-level algorithm consists of alternating the application of the smoother and the coarse grid correction.

Algorithm 8: Two-grid method (two-level V-cycle with post-smoothing)

input: ψ, η, ν
output: ψ

1 $r \leftarrow \eta - D\psi$
2 $\psi \leftarrow \psi + PD_c^{-1}Rr$
3 $r \leftarrow \eta - D\psi$
4 $\psi \leftarrow \psi + M_{SAP}^{(\nu)}r$

Algorithm 8 represents one coarse grid correction with ν steps of post-smoothing, also termed two-level *V-cycle* (cf. Figure 4.1) in the multigrid literature. Presmoothing can also be considered but does not yield any advantage in a two-level setting. This is due to

$$\text{spec}\left((I - M_{SAP}D)(I - PD_c^{-1}RD)\right) = \text{spec}\left((I - PD_c^{-1}RD)(I - M_{SAP}D)\right) .$$

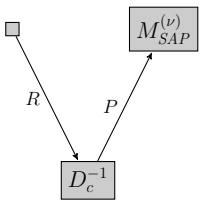

Figure 4.1: Illustration of a two-level V-cycle with post-smoothing only.

Typically, this method is applied as a preconditioner to a Krylov subspace method in the same manner as SAP in the previous chapter.

The update provided to the current iterate ψ is a so-called *multiplicative* update since coarse grid correction and smoother are applied sequentially to the latest residual $r = \eta - D\psi$. Performing both steps for a common residual, i.e., omitting the second residual update in line 3, is called an *additive* update. The current iterate ψ is then updated via

$$\psi \leftarrow \psi + \left(P D_c^{-1} R + M_{SAP}^{(\nu)} \right) (\eta - D\psi) \,.$$

This allows for applying both, smoother and coarse grid correction at the same time, but it might harm the efficiency per iteration of the outer Krylov solver. For a more detailed description, see, e.g., [104].

Algorithm 8 can be recursively extended to true multigrid by formulating a two-level algorithm of this kind for the solution of (4.1) required in line 2 until we obtain an operator which is small enough to solve (4.1) directly.

To be computationally beneficial, solving (4.1) has to be much cheaper than solving the original equation $D\psi = \eta$. For this purpose D_c has to be very small and/or sparse. As the number of eigenvectors that are not sufficiently reduced by the SAP smoother grows with n, cf. [9], one should not aim at fixing n_c (as in deflation methods), but at finding sparse matrices R and P whose ranges approximate the left and right near kernel of D well, respectively.

4.1 Aggregation-based Intergrid Transfer Operators

Consider a block decomposition $\mathcal{L}_i$ of the lattice $\mathcal{L}$. It has been observed in [79] that eigenvectors belonging to small eigenvalues of D tend to (approximately) coincide on a large number of lattice-blocks $\mathcal{L}_i$, a phenomenon which was termed "local coherence". Local coherence means in particular that we can represent many eigenvectors belonging to small eigenvalues from just a few by decomposing them into the parts belonging to each of the lattice-blocks. We refer to [79] for a detailed qualitative analysis of this observation. Local coherence is the philosophy behind the aggregation-based intergrid transfer operators introduced in a general setting in [20, 25] and applied to QCD problems in [5, 23, 92]. Similarly to a block decomposition we define an aggregation.

Definition 4.2
An aggregation $\{\mathcal{A}_1, \ldots, \mathcal{A}_s\}$ *is a partitioning of the set* $\mathcal{V} = \mathcal{L} \times \mathcal{C} \times \mathcal{S}$ *of all variables (recall Definition* 2.1*). It is termed a* lattice-block based *aggregation if each* $\mathcal{A}_i$ *is of the form*

$$\mathcal{A}_i := \mathcal{L}_{j(i)} \times \mathcal{W}_i,$$

where $\mathcal{L}_j$ *are the lattice-blocks of a block decomposition of the lattice* $\mathcal{L}$ *and* $\mathcal{W}_i \subseteq \mathcal{C} \times \mathcal{S}$*.*

Aggregates for the lattice Wilson-Dirac operator (3.2) will typically be realized as lattice-block based aggregates. The difference to a block decomposition purely based on a lattice decomposition is that an aggregate does not have to contain all spin and color variables. Thus, a lattice block $\mathcal{L}_i$ can be associated with more than one aggregate. Note that, however, the SAP smoother on the one hand and interpolation and restriction on the other hand do not have to be based on a *common* block decomposition of $\mathcal{L}$.

Definition 4.3
Consider a set $\{v_1, \ldots, v_N\} \subseteq \mathbb{C}^n$ *of so-called* test vectors *representing the near kernel and a set of aggregates* $\{\mathcal{A}_1, \ldots, \mathcal{A}_s\}$*. The interpolation operator* P *is then defined by decomposing the test vectors over the aggregates*

$$(v_1 \mid \ldots \mid v_N) = \begin{pmatrix} \square \\ \square \\ \square \\ \square \end{pmatrix} \longrightarrow P = \begin{pmatrix} \square & & & \\ & \square & & \\ & & \ddots & \\ & & & \square \end{pmatrix} \begin{matrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \vdots \\ \mathcal{A}_s \end{matrix}. \tag{4.2}$$

Formally, each aggregate $\mathcal{A}_i$ *induces* N *variables* $(i-1)N+1, \ldots, iN$ *on the coarse system, and we define*

$$Pe_{(i-1)N+j} := \mathcal{I}_{\mathcal{A}_i} \mathcal{I}^H_{\mathcal{A}_i} v_j, \quad \textit{for} \quad i = 1, \ldots, s,\ j = 1, \ldots, N. \tag{4.3}$$

Herein, $\mathcal{I}^H_{\mathcal{A}_i}$ (recall Definition 3.2) represents the trivial restriction operator for the aggregate $\mathcal{A}_i$, i.e., $\mathcal{I}_{\mathcal{A}_i} \mathcal{I}^H_{\mathcal{A}_i} v_j$ leaves the components of v_j from $\mathcal{A}_i$ unchanged while zeroing all others, and $e_{(i-1)N+j}$ denotes the $(i-1)N + j$-th unit vector. For the sake of stability, the test vectors are orthonormalized locally, i.e., for each i we replace $\mathcal{I}^H_{\mathcal{A}_i} v_j$ in (4.3) by the j-th basis vector of an orthonormal basis of $\operatorname{span}(\mathcal{I}^H_{\mathcal{A}_i} v_1, \ldots, \mathcal{I}^H_{\mathcal{A}_i} v_N)$. This neither alters the range of P nor changes the coarse grid correction operator $I - P(RDP)^{-1}RD$, and it ensures $P^H P = I$.

The restriction R is obtained in an analogous manner by using a set of test vectors $\{\hat{v}_1, \ldots, \hat{v}_N\}$ and the same aggregates to build R^H. Figure 4.2 illustrates

a lattice-block based aggregation from a lattice point of view—reduced to two dimensions—where in each aggregate $\mathcal{A}_i$ we take $\mathcal{W}_i$ as the whole set $\mathcal{C} \times \mathcal{S}$. Then, the aggregates can be viewed as forming a new, coarse lattice and the sparsity and connection structure of $D_c = RDP$ resembles the one of D, i.e., we have again a nearest neighbor coupling. Each lattice point of the coarse grid, i.e., each aggregate, holds N variables.

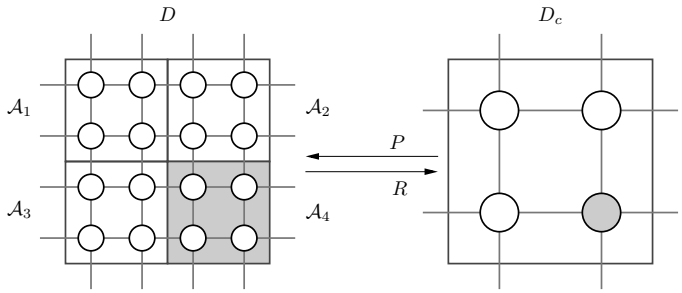

Figure 4.2: Aggregation-based interpolation (geometrical point of view reduced to 2D).

With respect to parallelization it is advisable to choose the aggregates $\mathcal{A}_i$ as a refinement of the parallelization block decomposition $\mathcal{P}_i$ (see Chapter 3). This allows one to perform interpolation and restriction without communication.

4.2 Petrov-Galerkin Approach in Lattice QCD

The structure and the spectral properties of the Wilson-Dirac operator D suggest to explicitly tie the restriction R to the interpolation P. The following construction of P—and thus R—is similar to constructions found in [5, 23, 79, 92] in the sense that the structure of all these interpolation operators is similar while the test vectors v_i upon which the interpolation is built—and therefore the action of the operators—are different.

Due to Lemma 2.14 it is natural to choose

$$R = (\Gamma_5 P)^H$$

in the aggregation-based intergrid operators: if P is built from vectors $v_1, \ldots, v_N$ which approximate right eigenvectors with small eigenvalues of D, then $R = (\Gamma_5 P)^H$ is built from vectors $\hat{v}_i = \Gamma_5 v_i$ which approximate left eigenvectors with small eigenvalues.

As was pointed out in [5], it is furthermore possible to even have $R = P^H$ by taking the special spin-structure of the Dirac operator into account when defining the aggregates. To be specific, we introduce the following definition.

Definition 4.4
The aggregation $\{\mathcal{A}_i : i = 1, \ldots, s\}$ *is termed* Γ_5-compatible *if any given aggregate* $\mathcal{A}_i$ *is composed exclusively of fine variables with spin 0 and 1 or of fine variables with spin 2 and 3.*

Assume that we have a Γ_5-compatible aggregation and consider the interpolation operator P from (4.2). Since Γ_5 acts as the identity on spins 0 and 1 and as the negative identity on spins 2 and 3, when going from P to $\Gamma_5 P$, each of the nonzero blocks in P belonging to a specific aggregate is either multiplied by $+1$ or by -1. This gives

$$\Gamma_5 P = P\Gamma_5^c \tag{4.4}$$

with Γ_5^c acting as the identity on the spin-0-1-aggregates and as the negative identity on the spin-2-3-aggregates.

Lemma 4.5
Let the aggregation be Γ_5*-compatible and* P *a corresponding aggregation-based prolongation as in* (4.2) *and* $R = (\Gamma_5 P)^H$*. Consider the two coarse grid operators*

$$D_c^{PG} = RDP \quad \text{and} \quad D_c = P^H DP.$$

Then

(i) $D_c = \Gamma_5^c D_c^{PG}$.

(ii) $I - PD_c^{-1}P^H D = I - P(D_c^{PG})^{-1}RD$.

(iii) D_c^{PG} *is hermitian,* D_c *is* Γ_5^c*-symmetric.*

(iv) For the field of values $\mathcal{F}(D) := \{\psi^H D\psi : \psi^H\psi = 1\}$*, we have* $\mathcal{F}(D_c) \subseteq \mathcal{F}(D)$.

Proof. We first observe that just as Γ_5 the operator Γ_5^c is diagonal with diagonal entries $+1$ or -1, so $\Gamma_5^c = (\Gamma_5^c)^H = (\Gamma_5^c)^{-1}$. Part *(i)* now follows from

$$D_c^{PG} = RDP = (\Gamma_5 P)^H DP = (P\Gamma_5^c)^H DP = \Gamma_5^c P^H DP = \Gamma_5^c D_c.$$

Consequently,

$$P(D_c^{PG})^{-1}RD = PD_c^{-1}\Gamma_5^c P^H \Gamma_5 D = PD_c^{-1}\Gamma_5^c\Gamma_5^c P^H D = PD_c^{-1}P^H D,$$

which gives *(ii)*. For part *(iii)* we observe that

$$(D_c^{PG})^H = P^H D^H R^H = P^H D^H \Gamma_5 P = P^H \Gamma_5 DP = RDP = D_c^{PG}.$$

This shows that D_c^{PG} is hermitian, which is equivalent to $D_c = \Gamma_5^c D_c^{PG}$ being Γ_5^c-symmetric. Finally, since $P^H P = I$, we have

$$\begin{aligned} \mathcal{F}(D_c) = \{\psi_c^H D_c \psi_c : \psi_c^H \psi_c = 1\} &= \{(P\psi_c)^H D(P\psi_c) : (P\psi_c)^H (P\psi_c) = 1\} \\ &\subseteq \{\psi^H D\psi : \psi^H\psi = 1\} = \mathcal{F}(D), \end{aligned}$$

which gives *(iv)*. □

Lemma 4.5 has some remarkable consequences. Part *(ii)* shows that we end up with the same coarse grid correction, irrespectively of whether we pursue a Petrov-Galerkin approach (operator D_c^{PG} with $R = \Gamma_5 P$) or a Galerkin approach (operator D_c, restriction is the adjoint of the prolongation). The Petrov-Galerkin operator D_c^{PG} inherits the hermiticity of the operator $\Gamma_5 D$, whereas the Galerkin operator D_c inherits the Γ_5-symmetry (and thus the symmetry of the spectrum, see Lemma 2.14) of D. Moreover, if $\mathcal{F}(D)$ lies in the right half plane, then so does $\mathcal{F}(D_c)$ and thus the spectrum of D_c. It is known that the "symmetrized" Wilson-Dirac operator $\Gamma_5 D$ is close to maximally indefinite [55], i.e., the number of negative eigenvalues is about the same as the positive ones. Numerical experiments indicate that this property seems to be inherited by $\Gamma_5^c D_c = D_c^{PG}$.

Γ_5-symmetry implies an interesting connection between the eigensystem of $\Gamma_5 D$ and the singular values and vectors of D.

Corollary 4.6
Let

$$\Gamma_5 D = V \Lambda V^H, \;\; \Lambda \text{ diagonal}, \;\; V^H V = I$$

be the eigendecomposition of the hermitian operator $\Gamma_5 D$*, then*

$$D = (\Gamma_5 V \operatorname{sign}(\Lambda)) \; |\Lambda| \; V^H = U \Sigma V^H \tag{4.5}$$

is the singular value decomposition of D *with* $U := \Gamma_5 V \operatorname{sign}(\Lambda)$ *unitary and* $\Sigma := |\Lambda|$.

The theory of algebraic multigrid methods for non-hermitian problems recently developed in [26] suggests to base interpolation and restriction on the right and left singular vectors corresponding to small singular values rather than on eigenvectors, so we could in principle use the relation (4.5). However, obtaining good approximations for the singular vectors belonging to small singular values is now much harder than obtaining good approximations to eigenvectors belonging to small eigenvalues, since the small singular values lie right in the middle of the spectrum of $\Gamma_5 D$, whereas the small eigenvalues of D lie at the "border" of its spectrum (and in the right half plane $\mathbb{C}^+$ if $\mathcal{F}(D) \subset \mathbb{C}^+$). Numerically we did not find that going after the singular values payed off with respect to the solver performance and it significantly increased the setup timing. These observations led us to the eigenvector based adaptive multigrid approach presented here; it also motivates that we consider D_c rather than D_c^{PG} as the "correct" coarse grid system to work with recursively in a true multigrid method.

In our computations, we take special Γ_5-compatible, lattice-block based aggregations.

Definition 4.7
Let $\{\mathcal{L}_j \; : \; j = 1, \ldots, n_{\mathcal{L}_c}\}$ *be a block decomposition of the lattice* $\mathcal{L}$*. Then the*

standard aggregation $\{\mathcal{A}_{j,\tau} : j = 1, \ldots, n_{\mathcal{L}_c},\ \tau = 0, 1\}$ *with respect to this block decomposition is given by*

$$\mathcal{A}_{j,0} := \mathcal{L}_j \times \{0, 1\} \times \mathcal{C} \quad \textit{and} \quad \mathcal{A}_{j,1} := \mathcal{L}_j \times \{2, 3\} \times \mathcal{C}\,.$$

Aggregates of the standard aggregation always combine two spin degrees of freedom in a Γ_5-compatible manner and all three color degrees of freedom. For any given j, the two aggregates $\mathcal{A}_{j,0}$ and $\mathcal{A}_{j,1}$ are the only two aggregates associated with the lattice-block $\mathcal{L}_j$. The standard aggregates thus induce a coarse lattice $\mathcal{L}_c$ with $n_{\mathcal{L}_c}$ sites where each coarse lattice site corresponds to one lattice-block $\mathcal{L}_j$ and holds $2N$ variables with N the number of test vectors. N variables correspond to spin 0 and 1 (and aggregate $\mathcal{A}_{j,0}$); another N variables to spin 2 and 3 (and aggregate $\mathcal{A}_{j,1}$). Thus, the overall system size of the coarse system is $n_c = 2Nn_{\mathcal{L}_c}$.

With standard aggregation, in addition to the properties listed in Lemma 4.5, the coarse system $D_c = P^H D P$ also preserves the property that coarse lattice points can be arranged as a 4D periodic lattice such that the system represents a nearest neighbor coupling on this torus. Each coarse lattice point now carries $2N$ variables.

4.3 Adaptive Test Vector Computation

If no *a priori* information about the near kernel is available, the test vectors $v_1, \ldots, v_N$ to be used in an aggregation-based multigrid method have to be obtained computationally during a *setup phase*.

4.3.1 Adaptivity in Aggregation-based AMG

We briefly review the setup concept of adaptive (smoothed) aggregation as described in [25]. We do so in the Galerkin context, i.e., we take

$$R = P^H\,.$$

The first fundamental idea of adaptivity in algebraic multigrid methods is to use the smoother to find error components not effectively reduced by the smoother, i.e., belonging to the near kernel. Starting with an initial random vector u, some iterations with the smoothing scheme on the homogeneous equations

$$Du = 0$$

yield a vector $\tilde{v}$ rich in components that are not effectively reduced. The first set of test vectors then is the singleton $\{v\}$, and one constructs the corresponding aggregation-based interpolation P from (4.2). This construction guarantees that v is in range(P) and thus is treated on the coarse grid.

Once a first two-grid or multigrid method is constructed in this way, one can use it to generate an additional vector not effectively reduced by the current method by again iterating on the homogeneous system. This newly found vector is added to the set of test vectors upon which we build new interpolation and coarse grid operators. Continuing in this manner we ultimately end up with a multigrid method which converges rapidly, but possibly at a high computational cost for the setup if many vectors need to be generated and incorporated in the interpolation operator. To remedy this issue, already in [25], some sophisticated ideas to filter the best information out of the produced vectors, are proposed which have been partly used in the implementations of adaptive algebraic multigrid for QCD described in [5, 23, 92].

The use of the homogeneous equation $Du = 0$ as the anvil to shape useful information by working on it with the most advanced method at hand, is the core idea of early adaptivity in algebraic multigrid methods.

4.3.2 Adaptivity in Bootstrap AMG

It is possible to use the current multigrid method in an adaptive setup in more ways than just to test it for deficiencies by applying it to the homogeneous equation $Du = 0$. This is exploited in the *bootstrap* approach developed in [21, 22] which we sketch now. Details will be discussed in connection with the inexact deflation method in Section 5.3 and Chapter 8. The following observation is crucial.

Lemma 4.8
Given an eigenpair (v_c, λ_c) of the generalized eigenvalue problem on the coarse grid

$$D_c v_c = \lambda_c P^H P v_c \,,$$

(Pv_c, λ_c) solves the constrained eigenvalue problem

$$\textit{find } (v, \lambda) \textit{ with } v \in \textit{range}(P) \textit{ s.t. } P^H (Dv - \lambda v) = 0$$

on the fine grid.

Proof.

$$P^H (DPv_c - \lambda_c P v_c) = D_c v_c - \lambda_c P^H P v_c = 0.$$

□

This observation allows to use the coarse grid system as a source of information about the eigenvectors with small eigenvalues of the fine grid system. Computing eigenvectors with small λ_c on the coarse grid is cheaper than on the fine grid, and applying a few iterations of the smoother to the lifted vectors Pv_c yields useful test vectors rich in components belonging to the near kernel of the fine

grid system. As we will see, the setup process used in the "inexact deflation" approach from [79], explained in the next chapter, can also be interpreted as a bootstrap-type setup, where instead of using an exact solution to the coarse grid eigenproblem only approximations are calculated.

4.4 Implementation Details

In this section we give implementation details for the Galerkin approach using the Γ_5-compatible standard aggregation from Definition 4.7, proposed in [5, 23, 92]. The prolongation P and restriction P^H do not require any communication as long as the aggregates $\mathcal{A}_{i,\tau}$ are a refinement of the process decomposition $\mathcal{P}_k$ introduced in Section 3.3. Besides the application of P and P^H we also state the properties and the construction of the coarse grid operator D_c in detail.

4.4.1 Prolongation and Restriction

Considering a standard aggregation $\{\mathcal{A}_{i,0}, \mathcal{A}_{i,1} : i = 1, \dots, s\}$ and N test vectors $v_1, \dots, v_N \in \mathbb{C}^n$, a coarse grid vector $c \in \mathbb{C}^{n_c}$ consists of $2N$ variables per coarse lattice site. On the i-th lattice site, the first N variables, i.e., $c_{2N(i-1)+j}$ for $j = 1, \dots, N$, correspond to $\mathcal{A}_{i,0}$, the second N variables, i.e., $c_{2N(i-1)+j}$ for $j = N+1, \dots, 2N$, correspond to $\mathcal{A}_{i,1}$. The information from the variables $c_{2N(i-1)+j}$ and $c_{2N(i-1)+N+j}$ is obtained from and transferred back to the fine grid via the j-th test vector. The prolongation Pc of a coarse grid vector c to the fine grid is realized as

$$Pc = \sum_{i=1}^{s} \sum_{j=1}^{N} \left(c_{2N(i-1)+j} \, \mathcal{I}_{\mathcal{A}_{i,0}} \mathcal{I}^H_{\mathcal{A}_{i,0}} v_j + c_{2N(i-1)+N+j} \, \mathcal{I}_{\mathcal{A}_{i,1}} \mathcal{I}^H_{\mathcal{A}_{i,1}} v_j \right) .$$

Herein, $\mathcal{I}^H_{\mathcal{A}_{i,\tau}}$ restricts a vector to an aggregate and $\mathcal{I}_{\mathcal{A}_{i,\tau}}$ re-embeds it into the whole fine grid space. Hence, the operation $c_{2N(i-1)+j} \, \mathcal{I}_{\mathcal{A}_{i,0}} \mathcal{I}^H_{\mathcal{A}_{i,0}} v_j$ is realized as multiplying the components v_j belonging to $\mathcal{A}_{i,0}$ by $c_{2N(i-1)+j}$. Thus, the outer sum over i, the aggregates, is rather of symbolic character, it does not introduce any additions. The total cost of a prolongation is equivalent to N AXPY operations of size n and therefore dominated by the cost for memory access rather than arithmetic cost. For a fast application of P, the prolongation has to be coherent in memory. If the test vectors v_j are stored one after the other, we run N times through the result vector $w = Pc$, i.e., once for adding each aggregate-wise scaled test vector v_j. Together with the v_j's this means running through $2N$ vectors. This number can be reduced to $N + 1$ by glueing the components $v_{j,k}$ for $j = 1, \dots, N$, i.e., storing the test vectors in the order $([(v_1)_1, \dots, (v_N)_1], \dots, [(v_1)_n, \dots, (v_N)_n])$ or, in other words, storing P row wise.

Then, every component w_k is computed once, yielding the final result

$$w_k = \sum_{j=1}^{N} c_{\alpha(k)+j} \, (v_j)_k \quad \text{with} \quad \alpha(k) = 2N(i-1) + N\tau \text{ for } k \in \mathcal{A}_{i,\tau} \, .$$

In the literature this approach is also referred to as "GAXPY" (generalized AXPY), see [37].

Considering a fine grid vector $w \in \mathbb{C}^n$, the restriction $P^H w$ is computed via

$$\begin{aligned} \left(P^H w\right)_{2N(i-1)+j} &= \left(\mathcal{I}^H_{\mathcal{A}_{i,0}} v_j\right)^H \mathcal{I}^H_{\mathcal{A}_{i,0}} w \\ \left(P^H w\right)_{2N(i-1)+N+j} &= \left(\mathcal{I}^H_{\mathcal{A}_{i,1}} v_j\right)^H \mathcal{I}^H_{\mathcal{A}_{i,1}} w \end{aligned}$$

for $i = 1, \ldots, s$ and $j = 1, \ldots, N$. The expression $\left(\mathcal{I}^H_{\mathcal{A}_{i,\tau}} v_j\right)^H \mathcal{I}^H_{\mathcal{A}_{i,\tau}} w$ evaluates that part of the inner product $v_j^H w$ with degrees of freedom from the aggregate $\mathcal{A}_{i,\tau}$. Hence, the total computational cost is equivalent to N inner products of size n and therefore also dominated by the cost of memory access. The latter is the same as for the prolongation for both options of storing the vectors. Thus, it is advisable to use the component-wise glued data layout from the prolongation for the restriction as well. To obtain $c \leftarrow P^H w \in \mathbb{C}^{n_c}$ being initially zero, every component w_k $(k = 1, \ldots, n)$ induces $j = 1, \ldots, N$ AXPY operations

$$c_{\alpha(k)+j} \leftarrow c_{\alpha(k)+j} + (v_j^H)_k \, w_k \quad \text{with} \quad \alpha(k) = 2N(i-1) + N\tau \text{ for } k \in \mathcal{A}_{i,\tau}$$

while running through a data amount of $N + 1$ fine grid vectors of size n.

4.4.2 The Coarse Grid Operator

First, we study the structure of the coarse grid operator before we explain how to construct it efficiently. As in the previous section, we have a standard aggregation $\{\mathcal{A}_{i,0}, \mathcal{A}_{i,1} \, : \, i = 1, \ldots, s\}$ and N test vectors $v_1, \ldots, v_N \in \mathbb{C}^n$. An entry of the coarse grid operator

$$(D_c)_{2N(p-1)+N\tau+k, 2N(q-1)+N\kappa+j} = \left(\left(\mathcal{I}_{\mathcal{A}_{p,\tau}} \mathcal{I}^H_{\mathcal{A}_{p,\tau}}\right) v_k\right)^H D \left(\mathcal{I}_{\mathcal{A}_{q,\kappa}} \mathcal{I}^H_{\mathcal{A}_{q,\kappa}}\right) v_j \qquad (4.6)$$

is nonzero if and only if $\mathcal{A}_{p,\tau} = \mathcal{A}_{q,\kappa}$ or $\mathcal{A}_{p,\tau}$ is a neighbor of $\mathcal{A}_{q,\kappa}$, i.e., their corresponding lattice-blocks are adjacent. The first case with $(p, \tau) = (q, \kappa)$ yields the self coupling part of the operator, the second yields the neighbor coupling part. Let

$$D_{\mathcal{A}_{p,\tau}, \mathcal{A}_{q,\kappa}} := \mathcal{I}_{\mathcal{A}_{p,\tau}} \mathcal{I}^H_{\mathcal{A}_{p,\tau}} D \, \mathcal{I}_{\mathcal{A}_{q,\kappa}} \mathcal{I}^H_{\mathcal{A}_{q,\kappa}} \, .$$

Due to the Γ_5-compatibility of the aggregates we have

$$\begin{aligned} D^H_{\mathcal{A}_{p,\tau},\mathcal{A}_{q,\kappa}} &= \left(\mathcal{I}_{\mathcal{A}_{p,\tau}}\mathcal{I}^H_{\mathcal{A}_{p,\tau}} D\, \mathcal{I}_{\mathcal{A}_{q,\kappa}}\mathcal{I}^H_{\mathcal{A}_{q,\kappa}}\right)^H \\ &= \mathcal{I}_{\mathcal{A}_{q,\kappa}}\mathcal{I}^H_{\mathcal{A}_{q,\kappa}} D^H \mathcal{I}_{\mathcal{A}_{p,\tau}}\mathcal{I}^H_{\mathcal{A}_{p,\tau}} \\ &= \mathcal{I}_{\mathcal{A}_{q,\kappa}}\mathcal{I}^H_{\mathcal{A}_{q,\kappa}} \Gamma_5 D \Gamma_5 \mathcal{I}_{\mathcal{A}_{p,\tau}}\mathcal{I}^H_{\mathcal{A}_{p,\tau}} \\ &= (-1)^\tau \mathcal{I}_{\mathcal{A}_{q,\kappa}}\mathcal{I}^H_{\mathcal{A}_{q,\kappa}} D\, \mathcal{I}_{\mathcal{A}_{p,\tau}}\mathcal{I}^H_{\mathcal{A}_{p,\tau}} (-1)^\kappa \\ &= (-1)^{\tau+\kappa} D_{\mathcal{A}_{q,\kappa},\mathcal{A}_{p,\tau}} \,. \end{aligned}$$

In particular, this implies

$$D^H_{\mathcal{A}_{p,\tau},\mathcal{A}_{q,\tau}} = D_{\mathcal{A}_{q,\tau},\mathcal{A}_{p,\tau}}$$

and

$$D^H_{\mathcal{A}_{p,\tau},\mathcal{A}_{q,\kappa}} = -D_{\mathcal{A}_{q,\kappa},\mathcal{A}_{p,\tau}} \quad \text{for} \quad \tau \neq \kappa \,.$$

Defining $\mathcal{A}_p := \mathcal{A}_{p,0} \cup \mathcal{A}_{p,1}$ we observe for the lattice-site-wise neighbor coupling matrices

$$D_{\mathcal{A}_p,\mathcal{A}_q} = \begin{pmatrix} D_{\mathcal{A}_{p,0},\mathcal{A}_{q,0}} & D_{\mathcal{A}_{p,0},\mathcal{A}_{q,1}} \\ D_{\mathcal{A}_{p,1},\mathcal{A}_{q,0}} & D_{\mathcal{A}_{p,1},\mathcal{A}_{q,1}} \end{pmatrix}$$

that the reversed coupling is given by

$$D_{\mathcal{A}_q,\mathcal{A}_p} = \begin{pmatrix} D^H_{\mathcal{A}_{p,0},\mathcal{A}_{q,0}} & -D^H_{\mathcal{A}_{p,1},\mathcal{A}_{q,0}} \\ -D^H_{\mathcal{A}_{p,0},\mathcal{A}_{q,1}} & D^H_{\mathcal{A}_{p,1},\mathcal{A}_{q,1}} \end{pmatrix} .$$

Therefore, only the coupling matrices in, e.g., positive directions need to be stored. Let $D_{\mathcal{A}_p} := D_{\mathcal{A}_p,\mathcal{A}_p}$. The self coupling matrices

$$D_{\mathcal{A}_p} = \begin{pmatrix} D_{\mathcal{A}_{p,0},\mathcal{A}_{p,0}} & D_{\mathcal{A}_{p,0},\mathcal{A}_{p,1}} \\ D_{\mathcal{A}_{p,1},\mathcal{A}_{p,0}} & D_{\mathcal{A}_{p,1},\mathcal{A}_{p,1}} \end{pmatrix}$$

fulfill $D_{\mathcal{A}_{p,\tau},\mathcal{A}_{p,\tau}} = D^H_{\mathcal{A}_{p,\tau},\mathcal{A}_{p,\tau}}$ and $D_{\mathcal{A}_{p,0},\mathcal{A}_{p,1}} = -D^H_{\mathcal{A}_{p,1},\mathcal{A}_{p,0}}$ such that only, say, the upper triangle of each matrix $D_{\mathcal{A}_p}$ needs to be stored.

Now, knowing the structure of the coarse grid operator we can formulate construction routines. Since the coarse grid operator is given by $D_c = P^H D P$, the simplest approach could be to calculate $u_{p,\tau,j} := D\mathcal{I}_{\mathcal{A}_{p,\tau}}\mathcal{I}^H_{\mathcal{A}_{p,\tau}} v_j$ for all aggregates $\mathcal{A}_{p,\tau}$ and all test vectors v_j and then compute $v_i^H \mathcal{I}_{\mathcal{A}_{q,\kappa}}\mathcal{I}^H_{\mathcal{A}_{q,\kappa}} u_{p,\tau,j}$ for all aggregates $\mathcal{A}_{q,\kappa}$ that are neighbors of $\mathcal{A}_{p,\tau}$. This approach requires $2sN$ applications of D. Since for a fixed aggregate size the number of aggregates s is of order $\mathcal{O}(n)$, the complexity of this approach is $\mathcal{O}(N \cdot n^2)$.

However, it is possible to eliminate the factor s, i.e., the dependence on the number of aggregates. First, we proceed with the computation of the self coupling matrices. Algorithm 9 describes the computation of the columns j and $N + j$ for all self coupling matrices of D_c, restricted to the upper triangle. When using

Algorithm 9: Compute columns j and $N+j$ of all self coupling matrices

input: j

output: columns j and $N+j$ for all self coupling matrices of D_c

1 $u_j^0 = \sum_{p=1}^{s} \left(D_{\mathcal{A}_{p,0},\mathcal{A}_{p,0}}\right) v_j$

2 $u_j^1 = \sum_{p=1}^{s} \left(D_{\mathcal{A}_{p,0},\mathcal{A}_{p,1}} + D_{\mathcal{A}_{p,1},\mathcal{A}_{p,1}}\right) v_j$

3 **for** $p = 1$ to s

4 **for** $i = 1$ to j {diagonal blocks}

5 **for** $\tau = 0$ to 1

6 $(D_c)_{2N(p-1)+N\tau+i,2N(p-1)+N\tau+j} = \left(\mathcal{I}^H_{\mathcal{A}_{p,\tau}} v_i\right)^H \mathcal{I}^H_{\mathcal{A}_{p,\tau}} u_j^\tau$

7 **for** $i = 1$ to N {top right block}

8 $(D_c)_{2N(p-1)+i,2N(p-1)+N+j} = \left(\mathcal{I}^H_{\mathcal{A}_{p,0}} v_i\right)^H \mathcal{I}^H_{\mathcal{A}_{p,1}} u_j^1$

aggregate restricted operators in lines 1 and 2 (as used in Section 3.5) both sums together require the cost of 3/4 of all aggregate restricted operators. Performing Algorithm 9 for $j = 1, \ldots, N$ we obtain the upper triangles of all self coupling matrices.

For the neighbor coupling, let

$$f_\mu : \{1, \ldots, s\} \to \{1, \ldots, s\}, \quad \mu = 0, 1, 2, 3$$

be functions with

$$\mathcal{A}_{f_\mu(p)} \text{ neighbor of } \mathcal{A}_p \text{ in positive direction } \mu\,.$$

For computing the neighbor couplings in positive μ direction, we have to compute the products $D_{\mathcal{A}_{p,\tau},\mathcal{A}_{f_\mu(p),\kappa}} v_j$. The coupling between neighboring aggregates $\mathcal{A}_{p,\tau}$ and $\mathcal{A}_{f_\mu(p),\kappa}$ involves only the boundaries $d_\mu^+ \mathcal{A}_{p,\tau}$ and $d_\mu^- \mathcal{A}_{f_\mu(p),\kappa} = \partial_\mu^+ \mathcal{A}_{p,\kappa}$ (cf. Definition 3.5). Therefore, it is sufficient to compute the products $D_{d_\mu^+ \mathcal{A}_{p,\tau},\partial_\mu^+ \mathcal{A}_{p,\kappa}} v_j$ on the boundaries only. Algorithm 10 shows how to compute the neighbor coupling matrices.

Summing up the costs of Algorithms 9 and 10 for $j = 1 \ldots, N$ we end up with $\mathcal{O}(N)$ applications of D and thus a complexity of

$$\mathcal{O}(N \cdot n)\,.$$

For computing the neighbor coupling in parallel we need the information of all ghost cells in positive μ direction for all test vectors v_j. During the communication for each test vector v_j we can compute the self coupling term corresponding to v_j. Hence, we can hide communication, and the entire computation of the coarse grid operator can be performed as illustrated in Algorithm 11.

Algorithm 10: Compute columns j and $N+j$ of neighbor couplings $(D_c)_\mu$

input: j, μ
output: columns j and $N+j$ for all neighbor coupling matrices of D_c for positive μ direction

$u_j^0 = \sum_{p=1}^{s} \left(D_{d_\mu^+ \mathcal{A}_{p,1}, \partial_\mu^+ \mathcal{A}_{p,0}} + D_{d_\mu^+ \mathcal{A}_{p,0}, \partial_\mu^+ \mathcal{A}_{p,0}} \right) v_j$

$u_j^1 = \sum_{p=1}^{s} \left(D_{d_\mu^+ \mathcal{A}_{p,0}, \partial_\mu^+ \mathcal{A}_{p,1}} + D_{d_\mu^+ \mathcal{A}_{p,1}, \partial_\mu^+ \mathcal{A}_{p,1}} \right) v_j$

for $p = 1$ to s
  for $i = 1$ to N
    for $\tau = 0$ to 1
      for $\kappa = 0$ to 1
        $(D_c)_{2N(p-1)+N\tau+i, 2N(f_\mu(p)-1)+N\kappa+j} = \left(\mathcal{I}^H_{d_\mu^+ \mathcal{A}_{p,\tau}} v_i \right)^H \mathcal{I}^H_{d_\mu^+ \mathcal{A}_{p,\tau}} u_j^\kappa$

Algorithm 11: Compute D_c

input: $v_1, \ldots, v_N$
output: D_c

for $j = 1$ to N
  for $\mu = 0$ to 3
    start updating $\partial_\mu^+ v_j$
  perform Algorithm 9 for index j
  for $\mu = 0$ to 3
    wait for update of $\partial_\mu^+ v_j$ being finished
    perform Algorithm 10 for j and μ

Once the coarse grid operator D_c has been constructed, the evaluation of a product $D_c \psi_c$ works similarly to what we explained in Algorithm 5 in Section 3.4 with the only difference that the projections $I_4 \pm \gamma_\mu$ are omitted. On the coarse grid equation we also use odd-even preconditioning which is implemented in a fashion analogous to that described in Section 3.4.1.

Chapter 5

Inexact Deflation

A hierarchical approach for solving the Wilson-Dirac equation

$$D\psi = \eta\,, \tag{5.1}$$

which lately received attention in the lattice QCD community, was proposed in [79]. It is a combination of what is called "inexact deflation" with an SAP preconditioned generalized conjugate residuals (GCR) method. The paper [79] does not relate its approach to the existing multigrid literature. The purpose of this chapter is to recast the formulations from [79] into established terminology from algebraic multigrid theory and to explain the limitations of the overall method from [79] which composes its multigrid ingredients in a non-optimal manner. We also explain how the setup employed in [79] to construct the "inexact deflation subspace" (i.e., the test vectors) can be viewed and used as a bootstrap setup in the sense of Section 4.3.2. This chapter is largely based on [49].

5.1 Deflation Subspace

The inexact deflation subspace constructed in [79] is the range of a linear operator P which resembles the definition of aggregation-based interpolation from (4.2). As in the aggregation-based construction it uses a set of test vectors $v_1, \ldots, v_N$ which are "chopped" up over aggregates $\mathcal{L}_i \times \mathcal{W}_i$ (called subdomains in [79]) to obtain the locally supported columns of P. These aggregates are not Γ_5-compatible since [79] takes

$$\mathcal{W}_i = \mathcal{C} \times \mathcal{S}\,.$$

Hence, the Γ_5-symmetry is not preserved on the coarse grid operator D_c which is obtained as $D_c = P^H D P$. Since the inexact deflation approach is not meant to be recursively extended to a true multilevel method, preserving important properties of the fine system on the coarse system is of lesser concern. However, within its

two-level framework, the projected test vectors $P^H v_1, \ldots, P^H v_N$ are used on the coarse grid to deflate the coarse system solve in a sense that $P^H v_1, \ldots, P^H v_N$ are used to augment the Krylov subspace for the corresponding coarse grid solver (see, e.g., [99]).

Two projections π_L, π_R are defined in [79] as follows.

Definition 5.1
Considering a lattice block decomposition $\mathcal{L}_i$ and corresponding aggregates $\mathcal{A}_i = \mathcal{L}_i \times \mathcal{C} \times \mathcal{S}$ and P the corresponding aggregation-based interpolation, we define a left projection π_L and a right projection π_R via

$$\pi_L = I - DPD_c^{-1}P^H \quad \textit{and} \quad \pi_R = I - PD_c^{-1}P^H D\,. \tag{5.2}$$

Clearly, π_R is the coarse grid correction introduced at the beginning of the previous chapter; cf. Lemma 4.5(ii).

Lemma 5.2
The projections π_L and π_R fulfill the following properties:

(i) $D\pi_R = \pi_L D$

(ii) $P^H\pi_L = \pi_R P = 0 = \pi_L DP = P^H D\pi_R$

(iii) $\pi_L(I - PP^H) = (I - PP^H)\pi_R = (I - PP^H)$

Proof. All three properties follow by direct computation.

(i) We have

$$D\pi_R = D - DPD_c^{-1}P^H D = \pi_L D\,.$$

(ii) Furthermore, we have

$$\pi_R P = P - PD_c^{-1}\underbrace{P^H DP}_{=D_c} = 0 \quad \text{and} \quad P^H\pi_L = P^H - \underbrace{P^H DP}_{=D_c}D_c^{-1}P^H = 0\,.$$

Using *(i)* we obtain

$$\pi_L DP = D\underbrace{\pi_R P}_{=0} = 0 \quad \text{and analogously} \quad P^H D\pi_R = \underbrace{P^H\pi_L}_{=0}D = 0\,.$$

(iii) Finally, we have

$$\pi_L(I - PP^H) = I - DPD_c^{-1}P^H - PP^H + DPD_c^{-1}\underbrace{P^H P}_{=I}P^H = I - PP^H$$

and

$$(I - PP^H)\pi_R = I - PD_c^{-1}P^H D - PP^H + P\underbrace{P^H P}_{=I}D_c^{-1}P^H D = I - PP^H\,.$$

□

Note that, while $I - PP^H$ is an orthogonal projection onto range$(P)^\perp$, π_L is an oblique projection onto range$(P)^\perp$ along range(DP), and π_R is an oblique projection onto range$(D^H P^\perp)$ along range(P).

In the context of inexact deflation these projections and the relation from Lemma 5.2*(i)* are used to decompose the linear system of equations

$$D\psi = \eta$$

as

$$D\pi_R\psi = \pi_L\eta \quad \text{and} \quad D(I - \pi_R)\psi = (I - \pi_L)\eta\,.$$

The second equation can be simplified to $(I - \pi_R)\psi = PD_c^{-1}P^H\eta$. Thus, the solution ψ can be computed as

$$\psi = \pi_R\psi + (I - \pi_R)\psi = \chi + \chi'\,,$$

where

$$\chi' = PD_c^{-1}P^H\eta \tag{5.3}$$

only requires the solution of the coarse grid system D_c and

$$D\chi = D\pi_R\psi = \pi_L\eta \tag{5.4}$$

is the "inexactly deflated" system which in [79] is solved by a right preconditioned Krylov subspace method. To be specific, the Krylov subspace is built for the operator

$$D\pi_R M_{SAP}^{(\nu)}$$

and the right hand side $\pi_L\eta$. The Krylov subspace method is GCR (general conjugate residuals, cf. [100]), a minimum residual approach which automatically adapts itself to the fact that the preconditioner $M_{SAP}^{(\nu)}$ is non-stationary, see the discussion in Section 3.2. We summarize all these steps in Algorithm 12.

Algorithm 12: Inexact deflation

input: η, ν
output: ψ
1 $\chi' \leftarrow PD_c^{-1}P^H\eta$
2 solve $D\pi_R M_{SAP}^{(\nu)} z = \pi_L\eta$ via GCR with $\chi = M_{SAP}^{(\nu)} z$
3 $\psi \leftarrow \chi + \chi'$

The inexact deflation method can also be interpreted as a one step multigrid method where (5.3) yields the coarse grid correction and the GCR solver for (5.4) yields the complementary post-smoother (cf. [110]). This can be seen by rewriting Algorithm 12 as shown in Algorithm 13. However, the post-smoother is not cheap at all since the GCR solver requires multiplications with π_R and $M_{SAP}^{(\nu)}$ in every iteration.

Algorithm 13: Inexact deflation as one step multigrid method

input: η, ν
output: ψ

1 $\psi = 0$
2 $\psi \leftarrow PD_c^{-1}P^H\eta = \psi + PD_c^{-1}P^H(\eta - D\psi)$ {coarse grid correction}
3 $r \leftarrow \pi_L\eta = \eta - D(PD_cP^H\eta) = \eta - D\psi$ {residual update}
4 solve $D\pi_R z = r$ right preconditioned with $M_{SAP}^{(\nu)}$ {post-smoother}
5 $\psi \leftarrow \psi + z$

5.2 Comparison of Multigrid and Inexact Deflation

Although the ingredients of an aggregation-based algebraic multigrid method as described in Chapter 4 and of "inexact deflation" as described in the previous section are the same, their composition makes the difference. In the multigrid context we combine the SAP smoothing iteration with the coarse grid correction such that it gives rise to the error propagator of a V-cycle with ν post-smoothing steps

$$E = (I - M_{SAP}^{(\nu)}D)(I - PD_c^{-1}P^HD)\,.$$

Hence, we obtain for one iteration of the V-cycle

$$\psi \leftarrow \psi + C^{(\nu)}r$$

where ψ denotes the current iterate and r the current residual $\eta - D\psi$, and

$$C^{(\nu)} = M_{SAP}^{(\nu)} + PD_c^{-1}P^H - M_{SAP}^{(\nu)}DPD_c^{-1}P^H\,. \tag{5.5}$$

In terms of the projectors (5.2) this can be written as

$$C^{(\nu)} = M_{SAP}^{(\nu)}\pi_L + PD_c^{-1}P^H\,.$$

Using the multigrid method as a right preconditioner in the context of a Krylov subspace method, the preconditioner is given by $C^{(\nu)}$, and the subspace is built for $DC^{(\nu)}$. We again should use a flexible Krylov subspace method such as flexible GMRES or GCR, since the smoother M_{SAP} is non-stationary and, moreover, we will solve the coarse system D_c only with low accuracy using some "inner iteration" in every step. The important point is that a rough approximation of the coarse grid correction in (5.5), i.e., the solution of systems with the matrix D_c at only low accuracy, will typically have only a negligible effect on the quality of the preconditioner, and it will certainly not hamper the convergence of the iterates towards the solution of the system since multiplications with the matrix D are done exactly. However, in the "inexact deflation" context the exact splitting of the solution $\psi = \chi' + \chi$ with

$$\chi' = PD_c^{-1}P^H\eta \quad \text{and} \quad D\pi_R\chi = \pi_L\eta$$

requires the same final accuracy for both χ' and χ. Therefore, when computing χ', the coarse grid system has to be solved with high accuracy. More importantly, D_c^{-1} also appears in π_R which is part of the "deflated" matrix $D\pi_R$ in the system for χ. In the inexact deflation context, this system is solved using SAP as a preconditioner. While we can allow for a flexible and possibly inexact evaluation of the preconditioner, the accuracy with which we evaluate the non-preconditioned matrix $D\pi_R$ in every step will inevitably affect the accuracy attainable for χ. As a consequence, in each iteration we have to solve the system with the matrix D_c arising in π_R with an accuracy comparable with the accuracy at which we want to obtain χ (although the accuracy requirements could, in principle, be somewhat relaxed as the iteration proceeds due to results from [103, 112]).

The difference of the two approaches is now apparent. In the multigrid context we are allowed to solve the coarse system with low accuracy, in inexact deflation we are not. Since the coarse grid system is still a large system, the work to solve it accurately will by far dominate the computational cost in each iteration in inexact deflation. In the multigrid context we can solve at only low accuracy without noticeably affecting the quality of the preconditioner, thus substantially reducing the computational cost of each iteration. Moreover, such a low accuracy solution can be obtained even more efficiently by a recursive application of the two-grid approach, resulting in a true multigrid method.

For further reading we refer to [67, 97, 108].

5.3 Adaptivity in the Setup of Inexact Deflation

To set up the inexact deflation method we need a way to obtain test vectors to construct the matrix P used to define the inexact deflation operators π_L and π_R. Once these vectors are found the method is completely defined (see Section 5.1). In analogy to the discussion of adaptive algebraic multigrid methods in Sections 4.3.1 and 4.3.2, these test vectors should contain information about the eigenvectors corresponding to small eigenvalues of the operator $DM_{SAP}^{(\nu)}$, the preconditioned system.

Though the setup proposed in [79] is similar in nature to the one described in Section 4.3.1, it differs in one important way. Instead of working on the homogeneous equation $D\psi = 0$ with a random initial guess to obtain the test vectors, it starts with a set of random test vectors v_j and approximately computes $D^{-1}v_j$ using SAP. The (approximate) multiplication with D^{-1} will amplify the components of the v_j's belonging to the near kernel. These new vectors are now used to define P (and D_c), yielding an inexact deflation method which can again be used to approximately compute $D^{-1}v_j$ giving new vectors for P. The whole process is repeated several times; see Algorithm 14 for a detailed description where a total of n_{inv} of these cycles is performed.

The update $v_j \leftarrow (M_{SAP}^{(\nu)}\pi_L + PD_c^{-1}P^H)v_j$ in line 8 of the algorithm is equiv-

Algorithm 14: Inexact deflation setup – IDsetup(n_{inv},ν) as used in [79]

input: n_{inv}, ν
output: $v_1, \ldots, v_N$, P, D_c
1 Choose $v_1, \ldots, v_N \in \mathbb{C}^n$ as random test vectors
2 **for** $\tau = 1$ to 3
3 **for** $j = 1$ to N
4 $v_j \leftarrow M_{SAP}^{(\tau)} v_j$
5 **for** $\tau = 1$ to n_{inv}
6 (re-)construct P and D_c from current $v_1, \ldots, v_N$
7 **for** $j = 1$ to N
8 $v_j \leftarrow (M_{SAP}^{(\nu)} \pi_L + P D_c^{-1} P^H) v_j$
9 $v_j \leftarrow \frac{v_j}{||v_j||}$

alent to the application of the V-cycle iteration matrix $C^{(\nu)}$ (cf. (5.5)). It can be interpreted as one step of an iteration to solve $Dv = v_j$ with initial guess 0 and iteration matrix $C^{(\nu)}$.

The update of the test vectors in line 8 can also be viewed in terms of the bootstrap AMG setup outlined in Section 4.3.2. While the first part of the update, $M_{SAP}^{(\nu)} \pi_L v_j$, is the application of a coarse grid correction followed by smoothing, i.e., a test to gauge the effectiveness of the method (cf. Section 4.3.1), the second part of the update, $P D_c^{-1} P^H v_j$ is in range(P). In contrast to the bootstrap methodology where an update in range(P) is obtained by interpolating eigenvectors with small eigenvalues of D_c, in the inexact deflation variant these "optimal" vectors are only approximated.

Chapter 6

GMRES-smoothed AMG

The general applicability of algebraic multigrid methods to lattice QCD systems was first shown in [4,5,23,92] where the resulting method is simply called "AMG", a terminology that we keep for this section and also for later discussion. A multi-level implementation of AMG for the clover improved Wilson-Dirac operator (2.8) is publicly available as part of the QOPQDP library [91]. In this chapter we specify the components which compose the AMG method and we describe its setup procedure that we found in the implementation within the QOPQDP library. This chapter is mainly based on [49].

In the AMG method the smoother consists of a few iterations, ν, of the generalized conjugate residual (GCR) method (cf. [100]). GCR is a Krylov subspace method which is mathematically equivalent to the GMRES method. It minimizes the residual within the Krylov subspace $\text{span}\{D^k r \,:\, k = 0, \ldots, \nu\}$ and its restarted version converges as long as the field of values of D resides in the right half plane, see [100]. For ν iterations of GCR and a given residual r, we denote the smoother iteration matrix with $M^{(\nu)}$ which is a polynomial of degree ν in D, i.e., $M^{(\nu)} = P_\nu(D, r)$. The corresponding error propagator is then given by

$$I - M^{(\nu)} D \,.$$

The coarse operator is given by $D_c = P^H D P$, where P is an aggregation-based prolongation obtained during a setup phase which produces the test vectors $v_1, \ldots, v_N$. The setup will be described in detail in Section 6.1. The aggregates are from a standard aggregation according to Definition 4.7, implying that it is, in particular, lattice-block based and Γ_5-compatible. Parameters of the aggregation are the underlying block decomposition of $\mathcal{L}$. The coarse grid matrix D_c inherits many important properties of D, cf. Lemma 4.5.

For the two-level setting, smoother and coarse grid correction are combined into a standard V-cycle with no pre- and ν steps of post-smoothing. Therefore, the error propagator is simply given by

$$(I - M^{(\nu)} D)(I - P D_c^{-1} P^H D) \,.$$

The V-cycle is used as a preconditioner for an outer right preconditioned GCR iteration. In the multilevel case, a K-Cycle is used. This rather sophisticated cycling strategy will be explained in Section 8.1.2.

The set of ingredients for AMG is summarized in Table 6.1.

pre-smoother	none
post-smoother	ν iterations of GCR
coarse grid correction	Γ_5-compatible aggregation, Galerkin type (cf. Section 4.2)
cycling strategy	K-cycle (cf. Section 8.1.2)
Krylov subspace wrapper	right-preconditioned GCR
setup procedure	Algorithm 15

Table 6.1: Ingredients for the AMG method.

6.1 Setup Procedure

It remains to specify the setup phase yielding the test vectors $v_1, \ldots, v_N$ for the prolongation operator P. In [23] the authors use the original adaptive setup of [25], sketched in Section 4.3.1. The numerical study in [23] was done only for the normal equations of the 2d Schwinger model. In [5], the setup was improved to one which—instead of having to consider one test vector a time—now works with 20 test vectors simultaneously on which 10 "relaxations" are performed as in the first loop of Algorithm 14 (with SAP replaced by GMRES and $\eta = 1, \ldots, 10$). In a second phase, groups of five test vectors each are then refined by iterating with the multigrid method in which prolongation is defined using only the complementary 15 test vectors. In the follow-up paper [92], the setup procedure was apparently changed. It is sketched without giving details, saying that the smoother is applied to random vectors and the interpolation is built from the resulting vectors.

The AMG implementation within the QOPQDP library [91] in spirit contains the setup from [92] combined with the idea of using inverse iteration from [79] (also cf. Section 5.3), but without making use of the bootstrap aspect. The code uses an inverse iteration scheme which computes test vectors using the BiCGStab method [100]. The test vectors are computed successively instead of simultaneously. More precisely, the i-th test vector $v_i^{(1)}$ is initialized as a random vector. Then, $v_i^{(2)} = D^{-1}v_i^{(1)}$ is approximated by a fixed number *bsi* of BiCGStab iterations. This is repeated as long as the quotient of two consecutive Rayleigh quotients for $D^H D$ is below a certain change level $cl < 1$, i.e.,

$$\frac{||Dv_i^{(k)}||_2}{||v_i^{(k)}||_2} < cl \cdot \frac{||Dv_i^{(k-1)}||_2}{||v_i^{(k-1)}||_2}\,, \tag{6.1}$$

Algorithm 15: AMG-setup(bsi, cl, mi)

input: bsi, cl, mi
output: $v_1, \ldots, v_N$, P, D_c

```
for i = 1 to N
    choose v_i^(1) as a random test vector
    for k = 1 to mi(i)
        orthonormalize v_i^(k) against v_1, ..., v_{i-1}.
        calculate v_i^(k+1) via bsi BiCGStab iterations for D v_i^(k+1) = v_i^(k) and
        initial guess zero
        if inequality (6.1) is not satisfied then
            break                                   {stop the k loop}
    v_i = v_i^(k+1)
construct P and D_c
```

or a maximum of $mi(i)$ repeats is exceeded. This particular bound depends on the index i of the test vector. For each test vector v_i the code implicitly assumes that the Rayleigh quotients monotonically decrease during the setup. For the computation of v_1 the code allows more iterations than for the subsequent ones. During this process, the test vector v_i is kept orthonormal to all previous $v_j,\ j < i$. Algorithm 15 gives the precise formulation of the entire process.

Chapter 7

Two-Level DD-αAMG

Chapters 3, 4 and 5 made all the ingredients available to describe our domain decomposition aggregation-based adaptive algebraic multigrid (DD-αAMG) method for the (clover improved) Wilson-Dirac operator (2.8). We now do so, mainly elaborating on our publication [49].

As the smoother we take $M_{SAP}^{(\nu)}$, i.e., we perform ν iterations of red-black Schwarz as formulated in Algorithm 3. Like ν, the underlying block decomposition of the lattice $\mathcal{L}$ is a parameter to the method which we will specify in the experiments.

The coarse system D_c is obtained as $D_c = P^H D P$, where P is an aggregation-based prolongation obtained during a setup phase. In contrast to the setup employed in AMG (see Chapter 6) we use an adaptive setup that makes use of the currently available coarse grid correction which is improved in each iteration of the setup. The aggregates are from a Γ_5-compatible standard aggregation (Definition 4.7, Lemma 4.5) and thus identical to those used in AMG.

We combine the smoothing iteration and the coarse grid correction into a standard V-cycle (cf. Chapter 4) with no pre- and ν steps of post-smoothing so that the iteration matrix of one V-cycle is given by $C^{(\nu)}$ from (5.5). Instead of using the V-cycle as a stand-alone solver, we run FGMRES, the flexible GMRES method (cf. [100]) as a wrapper for the V-cycle, i.e., the V-cycle is used as a (right) preconditioner for FGMRES.

The set of ingredients is summarized in Table 7.1.

7.1 Adaptive Two-Level Setup Procedure

It remains to specify how we perform the adaptive setup yielding the test vectors $v_1, \ldots, v_N$. Extensive testing showed that a modification of the inexact deflation setup (Algorithm 14) is most efficient. The modification is a change in the update of the vectors v_j in the second half of the algorithm. Instead of doing one iteration

pre-smoother	none
post-smoother	$M_{SAP}^{(\nu)}$ (cf. Section 3.2)
coarse grid correction	Γ_5-compatible aggregation, Galerkin type (cf. Section 4.2)
Krylov subspace wrapper	right-preconditioned flexible GMRES
setup procedure	Algorithm 16

Table 7.1: Ingredients for the DD-αAMG two-level method.

with $C^{(\nu)}$ and initial guess 0 to approximately solve $Dv = v_j$, we use the currently available vector v_j as our initial guess, see Algorithm 16.

Algorithm 16: DD-αAMG-setup(n_{inv}, ν)

input: n_{inv}, ν
output: $v_1, \ldots, v_N$, P, D_c
1 perform Algorithm 14 with line 8 replaced by $v_j \leftarrow v_j + C^{(\nu)}(v_j - Dv_j)$ $\{= C^{(\nu)}v_j + (I - C^{(\nu)}D)v_j\}$

7.2 Numerical Two-Level Results

In this section we exclusively show two-level results. This includes an elaborate study of the two-level case as well as comparisons with other existing two-level approaches. Studying and understanding the two-level case is essential for developing a multilevel approach.

We implemented the DD-αAMG method in the programming language `C` using the parallelization interface of `MPI`. Within BiCGStab we use odd-even preconditioning for D. In all multigrid approaches we use odd-even preconditioned restarted GMRES with a restart length of 30 when we solve the coarse system involving D_c. We implemented all odd-even preconditioned operators as explained in Sections 3.4 and 3.5, and the coarse grid correction is realized as in Section 4.4.

We use a mixed precision approach where we perform the V-cycle of the preconditioner in single precision. Low level optimization (e.g., making use of the `SSE`-registers on Intel/AMD architectures) was not implemented when the results for this section were produced and published. Results from an `SSE`-optimized version of this code are given in Section 8.4.6. All Krylov subspace methods (FGMRES, BiCGStab, GCR, CG) have been implemented in a common framework with the same degree of optimization to allow for a standardized comparison of computing times. This is particularly relevant when we compare timings with BiCGStab. We also include comparisons with the inexact deflation approach

(cf. Chapter 5) and AMG (cf. Chapter 6), where we used the efficient implementations which are publicly available [75, 91].

	parameter		default
setup	number of iterations	n_{inv}	6
	number of test vectors	N	20
	size of lattice-blocks for aggregates		4^4
	coarse system relative residual tolerance (stopping criterion for the coarse system)$^{(*)}$		$5 \cdot 10^{-2}$
solver	restart length of FGMRES	n_{kv}	25
	relative residual tolerance (stopping criterion)	tol	10^{-10}
smoother	number of post-smoothing steps$^{(*)}$	ν	2
	size of lattice-blocks in SAP$^{(*)}$		4^4
	number of minimal residual (MR) iterations to solve the local systems (3.3) in SAP$^{(*)}$		4

Table 7.2: Parameters for the DD-αAMG two-level method. $(*)$: same in solver and setup.

Table 7.2 summarizes the default parameters used for DD-αAMG in our experiments. Besides those discussed previously in this chapter, the table also gives parameter values for the stopping criterion used for the solves with the coarse system D_c (the initial residual is to be decreased by a factor of 20) and the stopping criterion for the entire FGMRES iteration (residual to be decreased by a factor of 10^{10}). In each SAP iteration we have to (approximately) solve the local systems (3.3). Instead of requiring a given decrease in the residual we here fix the number of iterative steps (to 4). The iterative method we use here is the odd-even preconditioned MR method (cf. Section 3.5). In our numerical experiments we found that $\nu = 2$ yields a good default parameter for the number of smoother iterations. In Section 3.6 we saw that FGMRES+SAP was performing even better for $\nu = 5$ since the SAP smoother reduces the number of all-to-all communications and improves data locality. Now, however, the optimal value for ν is rather a matter of balancing between coarse grid correction and smoother than of reducing communication.

For the various configurations and respective operators we found that this default set of parameters yields a well performing solver, with only little room for further tuning. The size of the lattice-blocks (4^4) fits well with all lattice sizes occurring in practice, where N_t and N_s are multiples of 4. The number of setup iterations, n_{inv}, is the only one of these parameters which should be adapted depending on the given situation. It will depend on how many systems we have to solve, i.e., how many right hand sides we have to treat. When n_{inv} is increased, the setup becomes more costly, while, at the same time, the solver becomes faster.

ID	lattice size $N_t \times N_s^3$	pion mass m_π [MeV]	CGNR iterations	shift m_0	clover term c_{sw}	provided by
2	48×16^3	250	7,055	−0.095300	1.00000	BMW-c [39, 40]
3	48×24^3	250	11,664	−0.095300	1.00000	BMW-c [39, 40]
4	48×32^3	250	15,872	−0.095300	1.00000	BMW-c [39, 40]
5	48×48^3	135	53,932	−0.099330	1.00000	BMW-c [39, 40]
6	64×64^3	135	84,207	−0.052940	1.00000	BMW-c [39, 40]
7	128×64^3	270	45,804	−0.342623	1.75150	CLS [29, 44]

Table 7.3: Configurations used together with their parameters. For details about their generation we refer to the references. Pion masses rounded to steps of 5 MeV. All BMW configurations were smeared with 3 steps of HEX [28] smearing, the CLS configuration was not smeared.

Thus, the time spent in the setup has to be balanced with the number of right hand sides, and we will discuss this in some detail in Section 7.2.2. The default $n_{inv} = 6$ given in Table 7.2 should be regarded as a good compromise.

We ran DD-αAMG on the various configurations, smeared and non-smeared ones, listed in Table 7.3, analyzed the behavior of the setup routine and performed different scaling tests. As in Section 3.6 all results were obtained on Juropa [66] at JSC.

7.2.1 Comparison with BiCGStab

First we compare a mixed precision[2], odd-even preconditioned implementation of BiCGStab with the DD-αAMG method using the standard parameter set for a 64^4 configuration at physical pion mass which represents an ill-conditioned linear system with $n = 201{,}326{,}592$.

	BiCGStab	DD-αAMG	speed-up factor	coarse grid
setup time		22.9s		
solve iter	13,450	21		$3{,}716^{(*)}$
solve time	91.2s	3.15s	29.0	2.43s
total time	91.2s	26.1s	3.50	

Table 7.4: BiCGStab vs. DD-αAMG with default parameters (Table 7.2) on an ill-conditioned 64^4 lattice (Table 7.3, configuration ID 6), 8,192 cores, $(*)$: coarse grid iterations summed up over all iterations on the fine grid.

The results reported in Table 7.4 show that we obtain a speed-up factor of 3.5 over BiCGStab with respect to the total timing. Excluding the setup time,

[2]The mixed precision implementation uses double precision flexible GMRES(25), preconditioned by 50 steps of single precision, odd-even preconditioned BiCGStab.

we gain a factor of 29. The right most column of Table 7.4 shows that in this ill-conditioned case about 77% of the solve time of DD-αAMG goes into computations on the coarse grid.

7.2.2 Setup Evaluation

Lattice QCD computations are dominated by two major tasks: generating configurations within the Hybrid Monte-Carlo (HMC) algorithm [70] and evaluating these configurations, i.e., calculating observables. Both tasks require solutions of the lattice Dirac equation.

The HMC generates a sequence of stochastically independent configurations. The configuration is changed in every step, and the Wilson-Dirac equation has to be solved only twice per configuration. Thus, HMC requires a new setup—or at least an update—for the interpolation and coarse grid operator in every step. Therefore, the costs of setup/update and solve have to be well-balanced.

The calculation of observables requires several, typically 12 or more, solves for a single configuration. Therefore, one would be willing to invest more time into the setup in order to obtain a better solver.

number of setup steps n_{inv}	average setup timing	average iteration count	lowest iteration count	highest iteration count	average solver timing	average total timing
1	2.08	149	144	154	6.42	8.50
2	3.06	59.5	58	61	3.42	6.48
3	4.69	34.5	33	36	2.37	**7.06**
4	7.39	27.2	27	28	1.95	9.34
5	10.8	24.1	24	25	**1.82**	12.6
6	14.1	23.0	23	23	1.89	16.0
7	16.8	**22.0**	22	22	1.88	18.7
8	19.5	**22.0**	22	22	2.02	21.5
9	21.8	**22.0**	22	22	2.15	24.0
10	24.3	22.5	22	23	2.31	26.6
11	26.6	23.0	23	23	2.38	29.0

Table 7.5: Evaluation of DD-αAMG-setup($n_{inv}, 2$) cf. Algorithm 16, 48^4 lattice, ill-conditioned configuration (Table 7.3, configuration ID 5), 2,592 cores, averaged over 20 runs. The bold numbers display the minima of the respective columns.

Table 7.5 illustrates how the ratio between setup and solve can be balanced depending on the amount of right hand sides. In this particular case, two steps in the setup might be the best choice if only a single solution of the system is needed (minimal time for setup + 1 solve). For many right hand sides, where the time spent in the solver dominates, 5 steps in the setup might be the best choice.

Doing up to 7 steps can lower the iteration count of the solver even further, but the better the test vectors approximate the near kernel, the more ill-conditioned the coarse system becomes, i.e., lowering the iteration count of the solver means increasing the iteration count on the coarse system. The iteration counts for $n_{inv} = 10, 11$ indicate that it might be possible to overdo the setup.

The numbers shown have been averaged over 20 runs, because the measurements vary due to the choice of random initial test vectors. The fourth and the fifth column of Table 7.5 show that the fluctuations in the iteration count of the solver are modest. For $n_{inv} \geq 4$ the fluctuations almost vanish completely.

	BiCGStab iteration counts					
	conf 1	conf 2	conf 3	conf 4	conf 5	conf 6
	7,950	8,350	9,550	8,600	8,100	9,950
	DD-αAMG iteration counts					
n_{inv}	conf 1	conf 2	conf 3	conf 4	conf 5	conf 6
1	161	208	175	183	181	272
2	62	75	67	67	64	85
3	34	37	36	37	35	39
4	27	28	28	29	27	29
5	24	25	25	25	24	25
6	23	23	23	24	23	23

Table 7.6: Configuration dependence study of BiCGStab and DD-αAMG with DD-αAMG-setup($n_{inv}, 2$) for 6 different, ill-conditioned configurations on 48^4 lattices, (Table 7.3, configuration ID 5), 2,592 cores.

Table 7.6 gives the iteration count of BiCGStab and DD-αAMG for a set of 6 stochastically independent configurations from a single HMC simulation. The BiCGStab iteration count shows a clear dependence on the gauge fields just as DD-αAMG for small values of n_{inv}. However, for $n_{inv} \geq 4$ the iteration count varies only marginally.

7.2.3 Scaling Tests

We now study the scaling behavior of the solver as a function of the mass parameter and the system size. While the former determines the condition number of the Wilson-Dirac operator, the latter has an effect on the density of the eigenvalues. In particular, increasing the volume leads to a higher density of small eigenvalues [9]. In a weak parallel scaling test we also analyze the performance as a function of the number of processors used.

Mass Scaling

For this study we used a 48^4 lattice configuration. We ran the setup once for the mass parameter $m_0 = -0.09933$ in the clover improved Wilson-Dirac operator (2.8). This represents an ill-conditioned system where the pion mass with 135 MeV is at its physical value. We then used the interpolation operator obtained for this system for a variety of other mass parameters, where we then ran the DD-αAMG solver without any further setup.

	BiCGStab		DD-αAMG		coarse system	
m_0	iteration count	solver timing	iteration count	solver timing	⌀iteration count	timing (% solve time)
−0.03933	350	2.27s	17	0.58s	9.94	0.12s (20.7)
−0.04933	400	2.60s	17	0.59s	11.2	0.13s (22.0)
−0.05933	450	2.97s	18	0.64s	12.7	0.16s (25.0)
−0.06933	600	4.10s	19	0.72s	15.4	0.20s (27.8)
−0.07933	850	5.45s	19	0.76s	19.4	0.25s (32.9)
−0.08933	1,550	9.82s	20	0.92s	28.6	0.37s (40.2)
−0.09033	1,700	10.3s	21	1.00s	30.8	0.43s (43.0)
−0.09133	1,700	10.6s	21	1.04s	33.4	0.47s (45.2)
−0.09233	1,800	11.7s	21	1.08s	36.4	0.51s (47.2)
−0.09333	2,250	13.7s	21	1.13s	39.7	0.55s (48.7)
−0.09433	2,650	16.7s	22	1.23s	43.2	0.62s (50.4)
−0.09533	2,850	17.4s	22	1.28s	46.9	0.68s (53.1)
−0.09633	3,450	21.8s	22	1.36s	51.3	0.75s (55.1)
−0.09733	3,750	23.7s	23	1.48s	56.5	0.84s (56.8)
−0.09833	4,700	28.5s	23	1.63s	65.9	0.99s (60.7)
−0.09933	6,250	42.0s	24	1.89s	79.3	1.22s (64.5)

Table 7.7: Mass scaling behavior of DD-αAMG for $n_{inv} = 5$, 48^4 lattice (Table 7.3, configuration ID 5), 2,592 cores.

In Table 7.7 we compare BiCGStab and DD-αAMG with respect to the effort for one right hand side and the scaling with the mass parameter m_0. For the smallest m_0, DD-αAMG is 22 times faster than BiCGStab and even for the largest value of m_0 there remains a factor of 3.9. We also see that the two methods scale in a completely different manner. The BiCGStab solve for the smallest m_0 is about 18 times more expensive (in terms of solve time and iteration count) than the solve for the largest one. On the other hand, the DD-αAMG timings just increase by a factor of 3.2, the iteration count even only by a factor of 1.4. The coarse grid iteration count, however, increases by a factor of 8.0 which explains the difference between the increase in iteration count and the increase in solver timing.

System Size Scaling

In Table 7.8 we report tests on the scaling with the system size for constant mass parameter and (physical) lattice spacing. We again compare DD-αAMG with BiCGStab. The iteration count of BiCGStab for $N_t \times N_s^3$ lattices appears to scale with N_s and thus almost doubles from $N_s = 16$ to $N_s = 32$ whereas for DD-αAMG we observe an almost constant iteration count and time to solution.

	BiCGStab		DD-αAMG		
lattice size $N_t \times N_s^3$	iteration count	solver timing	setup timing	iteration count	solver timing
48×16^3	1,550	7.03s	6.59s	20	0.89s
48×24^3	2,150	10.7s	7.29s	20	0.83s
48×32^3	2,600	13.1s	7.15s	21	0.92s

Table 7.8: Lattice size scaling of DD-αAMG, $n_{inv} = 6$ setup iterations, lattices generated with the same mass parameter and lattice spacing (Table 7.3, configuration ID 2, 3 and 4), local lattice size 4×8^3.

Weak Scaling

For a weak scaling test we ran 100 iterations of DD-αAMG with $n_{inv} = 5$ in the setup on lattices ranging from size 16^4 on a single node (8 cores/node) to $128^2 \times 64^2$ on 1,024 nodes with 16×8^3 local lattice size on each core, cf. Figure 7.1.

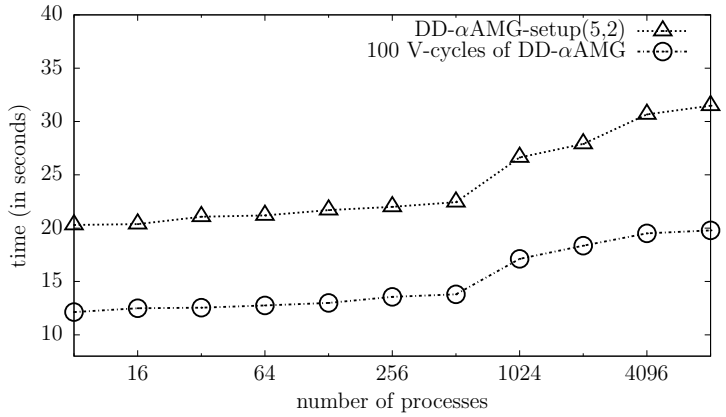

Figure 7.1: Weak scaling test of DD-αAMG. The lattice size is increased with the number of processes, keeping the local lattice size per process fixed to 16×8^3.

For the scaling study we fixed the number of iterations on the coarse grid to be exactly 50 steps of odd-even preconditioned GMRES so that we always have the same number of 100 `MPI_Allreduce` operations. In Figure 7.1 we see the usual

$\log(p)$ dependence, p the number of processes, caused by global communication, together with an exceptional increase when going from 512 to 1,024 processes. Additional measurements show that this is due to the fact that the `MPI_Allreduce` operations take substantially longer for 1,024 processors, a machine-specific feature of Juropa. Apart from this, our method scales well up to 8,192 processes.

7.2.4 Comparison with the Inexact Deflation Method

The inexact deflation code of [79] is publicly available [75]. We now compare its performance with DD-αAMG. Based on a preprint of [49], one of the papers containing results of this thesis, the inexact deflation method has been upgraded in the spirit of DD-αAMG (cf. [76]). The new version is termed "with inaccurate projection". We here compare with the older, "exact projection" version.

We have chosen the parameters of both methods equally except for the number of post-smoothing steps ν. For the inexact deflation method $\nu = 5$ and for DD-αAMG $\nu = 2$ turned out to provide the fastest solver, respectively. We used the `gcc` compiler with the `-O3` flag and hand coded low-level `SSE`-optimization for the inexact deflation method and the `icc` compiler with the optimization flags `-O3 -ipo -axSSE4.2 -m64` for the DD-αAMG method. These compiler options provide the optimal choices for the respective codes. Since our focus is on algorithmic improvements we did not employ customized `SSE`-optimization in this section. Results for DD-αAMG with `SSE`-optimization will be given in Section 8.4.6.

The following results were produced on the same 48^4 lattice as in Sections 7.2.2 and 7.2.3 and on a 128×64^3 lattice (Table 7.3, configurations ID 5 and 7).

Table 7.9 compares inexact deflation and DD-αAMG for a whole range for n_{inv}. We see that $n_{inv} = 5$ provides the fastest DD-αAMG solver which is two times faster than the fastest inexact deflation solver which requires $n_{inv} = 10$. Note that in this case the setup cost for both methods are about the same. For the calculation of observables where the same system has to be solved for several right hand sides, this factor of two directly carries over to the total computation time since the setup cost then is negligible.

When looking at combined times for setup and solve for one right hand side, $n_{inv} = 2$ is best for DD-αAMG, where it takes 6.48 seconds. The best choice for inexact deflation is $n_{inv} = 6$ requiring 10.89 seconds.

We also see that except for very small values for n_{inv}, the number of iterations required in DD-αAMG is less than half of that in inexact deflation. The numbers in parenthesis denote the average number of coarse solver iterations in each iteration of the respective method. For DD-αAMG the number of iterations on the coarse grid increases with the work spent in the setup. Hence, the lowest DD-αAMG-iteration count does not necessarily provide the fastest solver in the two grid setting. In inexact deflation the number of iterations on the coarse grid is not that clearly tied to n_{inv}. Since in inexact deflation the coarse system

	Inexact deflation			DD-αAMG		
setup steps n_{inv}	setup timing	iteration count (coarse)	solver timing	setup timing	iteration count (coarse)	solver timing
1	1.01s	233 (82)	10.1s	2.08s	149 (24)	6.42s
2	1.87s	155 (145)	10.2s	3.06s	59 (46)	3.42s
3	2.69s	108 (224)	9.96s	4.69s	35 (63)	2.37s
4	3.43s	84 (301)	9.25s	7.39s	27 (68)	1.95s
5	6.14s	70 (320)	7.50s	10.8s	24 (75)	**1.82**s
6	5.68s	63 (282)	5.21s	14.1s	23 (84)	1.89s
7	6.93s	58 (277)	4.67s	16.8s	**22** (90)	1.88s
8	7.71s	54 (267)	4.12s	19.5s	**22** (99)	2.02s
9	8.74s	51 (263)	3.89s	21.8s	**22** (109)	2.15s
10	10.1s	**49** (265)	**3.62**s	24.3s	**22** (116)	2.31s
11	11.3s	50 (266)	3.77s	26.6s	23 (119)	2.38s

Table 7.9: Comparison of DD-αAMG and inexact deflation, coarse system solver tolerance 10^{-12} and $\nu = 5$ in inexact deflation, ill-conditioned system on a 48^4 lattice (Table 7.3, configuration ID 5), 2,592 cores. The bold numbers display the minima of the respective columns.

has to be solved very accurately, the number of iterations needed to solve the coarse grid system is higher than in DD-αAMG. However, it is only moderately (a factor of 2 to 4) higher, though, because the projected test vectors are deflated on the coarse grid within the coarse system GCR solves (cf. Chapter 5). Additionally, for the same number of test vectors, DD-αAMG produces a coarse system which is twice as large (and contains four times as many nonzeros in the coarse grid operator) as that of inexact deflation with the benefit of preserving the Γ_5-structure on the coarse grid. The DD-αAMG coarse grid system seems to be more ill-conditioned—an indication that the important aspects of the fine grid system are represented on the coarse grid—and the resulting coarse grid correction clearly lowers the total iteration count more efficiently and thus speeds up the whole method. Both methods, DD-αAMG and inexact deflation, perform best when using 20 test vectors.

Finalizing our discussion of the two methods, we compare in Table 7.10 inexact deflation and DD-αAMG for an unsmeared configuration (configuration ID 7, Table 7.3), typical for many recent lattice QCD computations. Again we took the default parameter set, but now with relatively cheap setup phases. For inexact deflation we used 5 setup iterations and for DD-αAMG we used three. This yields the minimal total time for both methods and results in a speed-up factor of more than 1.4 for setup and solve in DD-αAMG against inexact deflation which shows that a rougher configuration does not harm the efficiency of DD-αAMG. Still 55% of the execution time is spent in solves of the coarse system in DD-αAMG.

	Inexact deflation	DD-αAMG	speed-up factor
smooth iter	5	2	
setup iter	5	3	
setup time	10.9s	7.85s	1.39
solve iter	31	45	
solve time	8.63s	5.81s	1.49
total time	19.5s	13.6s	1.43

Table 7.10: Comparison of DD-αAMG with inexact deflation on an ill-conditioned system on a 128×64^3 lattice (Table 7.3, configuration ID 7), same parameters as in Table 7.9, 8,192 cores.

7.2.5 GMRES/GCR Smoothing and "AMG"

As explained in Sections 4.2, 4.3.1 and 4.3.2, the AMG method (see Chapter 6) motivated many of the choices in DD-αAMG, particularly preservation of Γ_5-symmetry and aggregation-based interpolation. An important difference of the two methods is the choice of the smoothing iteration. While DD-αAMG uses some steps of SAP, which can be regarded as a "block smoothing", AMG uses "point smoothing", namely some steps of standard (odd-even preconditioned) GCR. Another difference is the setup which was introduced in Section 6.1 and compared with the setup from [79] introduced in Section 5.3. The latter is basically the setup used by DD-αAMG (cf. Section 7.1).

To study the influence of the different smoothers on the multigrid method, we implemented a GMRES smoother in our framework as well. In numerical tests with state-of-the-art lattices we found that GMRES and GCR always needed the same number of iterations to converge, i.e., we did not observe any stability issues. The difference in runtime was negligible as well. Hence, a comparison of GMRES and SAP as smoother iterations amounts to a comparison of "emulated" AMG and DD-αAMG. Using the same framework enables us to compare these approaches with the same degree of optimization and almost independently of the setup procedure being used.

Scanning the parameter range we found that 5 or 6 iterations of odd-even preconditioned GMRES smoothing gave the best results for AMG in our framework. In Table 7.11 we compare AMG with DD-αAMG and see a moderate speed-up of DD-αAMG over AMG in the setup and the solve phase. If we look just at the time spent in smoothing on the fine grid, we see a clear advantage of SAP over GMRES by a factor of 1.5. The impact of this factor on the overall runtime of the solver is expected to grow when employing additional levels due to the smoother being applied on coarser levels as well.

Note that moving from GMRES to SAP improves the arithmetic intensity in a parallel environment, i.e., the computation-to-communication and the computation-to-bandwidth ratio.

	AMG	DD-αAMG	speed-up factor
smoother iter	6	2	
setup iter	5	5	
setup time	19.7s	17.7s	1.11
smoother time	0.60s	0.40s	1.50
solve iter	23	23	
solve time	3.76s	3.07s	1.22
total time	23.5s	20.8s	1.13

Table 7.11: Comparison of DD-αAMG with AMG on an ill-conditioned system on a 64×64^3 lattice (Table 7.3, configuration ID 6), parameters from Table 7.2, 8,192 cores.

The AMG method has been implemented as part of the QOPQDP library which is publicly available, see [91]. Table 7.12 reports a comparison of DD-αAMG with AMG in QOPQDP for two of our configurations. We compared different choices of parameters with our standard parameter settings for DD-αAMG. We stopped the iterations when the initial residual was decreased by a factor of 10^{-5} (instead of 10^{-10}), the reason for this being that in QOPQDP configurations are represented in single precision by default. For the default choice of parameters in AMG we see that the setup is substantially more costly (factors between 2 and 4 in time), while the number of iterations for each system solve is slightly higher than for DD-αAMG. We can make the effort in the AMG setup comparable to that of DD-αAMG by reducing the limit on the maximum number of BiCGStab iterations to be performed in each inverse iteration step on each test

	ID 6, 128 cores			ID 7, 256 cores		
	AMG-d	AMG-20	DD-αAMG	AMG-d	AMG-10	DD-αAMG
setup time	2424s	826s	896s	2464s	607s	656s
solve iter	14	22	10	13	21	11
solve time	45.4s	66.0s	57.1s	36.5s	50.4s	37.3s
	ID 6, 8192 cores			ID 7, 8192 cores		
	AMG-d	AMG-40	DD-αAMG	AMG-d	AMG-20	DD-αAMG
setup time	52.3s	24.6s	27.7s	89.9s	29.1s	32.3s
solve iter	14	16	10	13	16	11
solve time	4.75s	5.51s	1.82s	3.49s	3.43s	1.86s

Table 7.12: Comparison of DD-αAMG and AMG as available from [91]. AMG-d uses default parameter settings, AMG-k sets $msi = k$ so that setup time is comparable to DD-αAMG. `SSE`-optimization switched off in AMG.

vector (*msi*), but then the number of iterations for each solve increases in AMG and solve times become always larger than with DD-αAMG. For large numbers of cores the solve times are 2 to 3 times smaller than in AMG. We believe that this is due to the different implementations.

Chapter 8

Multilevel DD-αAMG

Except for implementation details given in Section 8.3, this chapter is largely based on [48,49]. Our multilevel extension of the two-level approach from the previous chapter, to be presented here, combines the two same components, namely SAP as the smoother and a Γ_5-preserving aggregation-based interpolation, on every level. This means that the smoother as well as the interpolation together with the associated coarse-level correction are of the same type on all levels of the hierarchy. As in the two-level case, and for the same reasons, the multilevel method is used as a preconditioner to FGMRES.

To be more specific, we introduce the following definitions.

Definition 8.1
Let L be the number of levels employed and denote $D_1 := D$. Furthermore, let $n_\ell,\ \ell = 1, \dots, L$ be the dimension of the underlying vector space on each level ℓ. In the same manner as in Chapter 4 we define interpolation operators

$$P_\ell : \mathbb{C}^{n_{\ell+1}} \to \mathbb{C}^{n_\ell}, \quad \ell = 1, \dots, L-1$$

which transfer information from level $\ell+1$ to ℓ. Accordingly, the operators P_ℓ^H transfer information from ℓ to $\ell+1$. Using these operators we recursively define coarse-level operators

$$D_\ell : \mathbb{C}^{n_\ell} \to \mathbb{C}^{n_\ell}, \quad D_\ell := P_{\ell-1}^H D_{\ell-1} P_{\ell-1}$$

for $\ell = 2, \dots, L$. The complementary smoothers on each level are denoted by

$$M_\ell,\ \ell = 1, \dots, L-1\,.$$

Having all these ingredients, we call

$$\{(P_\ell, D_{\ell+1}, M_\ell)\ :\ \ell = 1, \dots, L-1\}$$

a multigrid hierarchy.

8.1 Multilevel Cycling Strategies

Considering a multigrid hierarchy with more than two levels, several different cycling strategies, i.e., recursive schemes for multigrid algorithms can be thought of. In Chapter 4 we introduced the two-level V-cycle which is extended to a multilevel V-cycle by recursive application to D_c^{-1}. For the sake of completeness we now introduce the multilevel V-cycle and also the W-cycle since they are the most common cycling strategies, and we introduce the K-cycle since it shows the best convergence behavior of all tested variants.

8.1.1 V-Cycles and W-Cycles

The simplest multilevel cycling strategy is the *V-cycle* illustrated in Algorithm 17. In a V-cycle, on each level only one recursive call is used in-between the pre- and post-smoothing iterations.

Algorithm 17: $\psi_\ell = \text{V-Cycle}(\ell, \eta_\ell)$

input: ℓ, η_ℓ
output: ψ_ℓ
if $\ell = L$ **then**
  $\psi_\ell \leftarrow D_\ell^{-1}\eta_\ell$
else
  $\psi_\ell = 0$
  for $i = 1$ to μ
    $\psi_\ell \leftarrow \psi_\ell + M_\ell(\eta_\ell - D_\ell\psi_\ell)$
  $\eta_{\ell+1} \leftarrow P_\ell^H(\eta_\ell - D_\ell\psi_\ell)$
  $\psi_{\ell+1} \leftarrow \text{V-Cycle}(\ell+1, \eta_{\ell+1})$
  $\psi_\ell \leftarrow \psi_\ell + P_\ell\psi_{\ell+1}$
  for $i = 1$ to ν
    $\psi_\ell \leftarrow \psi_\ell + M_\ell(\eta_\ell - D_\ell\psi_\ell)$

Oftentimes, an ordinary V-cycle does not provide a satisfactory convergence rate. As a typical phenomenon, the approximate solutions $\psi_{\ell+1}$ for $D_{\ell+1}^{-1}\eta_{\ell+1}$ on the intermediate levels are not accurate enough to provide an efficient coarse grid correction $P_\ell\psi_{\ell+1}$ on the next finer level. Considering more than just one recursion yields a possible remedy, the W-cycling strategy, see Algorithm 18. Obviously, for $k = 1$ the W-cycle turns into the ordinary V-cycle. It is also possible to consider different values of k on each level or even choose k adaptively, e.g., according to some residual tolerance. Both algorithms, the V- and the W-cycle are named after the shape of their respective recursion schemes as illustrated in Figure 8.1.

Algorithm 18: $\psi_\ell = \text{W-Cycle}(\ell, \eta_\ell)$

input: ℓ, η_ℓ
output: ψ_ℓ
if $\ell = L$ **then**
  $\psi_\ell \leftarrow D_\ell^{-1}\eta_\ell$
else
  $\psi_\ell = 0$
  for $j = 1$ to k
    for $i = 1$ to μ
      $\psi_\ell \leftarrow \psi_\ell + M_\ell(\eta_\ell - D_\ell\psi_\ell)$
    $\eta_{\ell+1} \leftarrow P_\ell^H(\eta_\ell - D_\ell\psi_\ell)$
    $\psi_{\ell+1} \leftarrow \text{W-Cycle}(\ell+1, \eta_{\ell+1})$
    $\psi_\ell \leftarrow \psi_\ell + P_\ell\psi_{\ell+1}$
    for $i = 1$ to ν
      $\psi_\ell \leftarrow \psi_\ell + M_\ell(\eta_\ell - D_\ell\psi_\ell)$

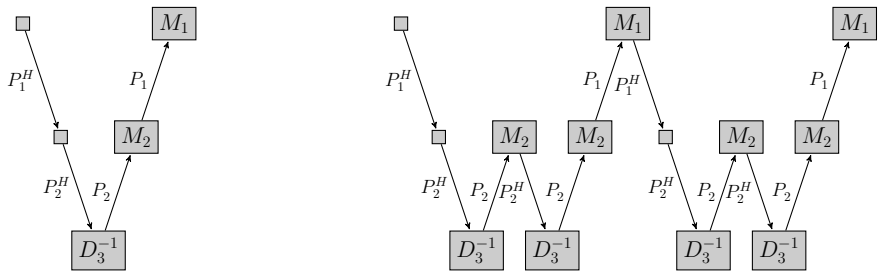

Figure 8.1: Illustration of a three-level V-cycle (left) and a three-level W-cycle of length $k = 2$ (right), both with post-smoothing only.

8.1.2 K-Cycles

Numerical tests with very large configurations and small quark masses have shown that simple V- or W-cycles are often not the ideal choice in terms of solver performance. We thus use a more elaborate cycling strategy, the K-cycle suggested in [90], in the multi-level method: A K-cycle optimally recombines several coarse-level solves, again in a recursive manner. More precisely, on every level ℓ we obtain an approximate solution of the coarse-level system by a few iterations of a flexible Krylov subspace method, preconditioned by the K-cycle multilevel method from level $\ell + 1$ to L. We deviate from the original approach in [90] in that we use a stopping criterion based on the reduction of the residual rather than a fixed number of iterations. We will give the specific choice for the stopping criterion used in our implementation in Section 8.4.

The K-cycle is stated in Algorithm 19. For a fixed number of iterations it can

Algorithm 19: $\psi_\ell = \text{K-Cycle}(\ell, \eta_\ell)$

input: ℓ, η_ℓ
output: ψ_ℓ
if $\ell = L$ **then**
  $\psi_\ell \leftarrow D_\ell^{-1}\eta_\ell$
else
  $\psi_\ell = 0$
  for $i = 1$ to μ
    $\psi_\ell \leftarrow \psi_\ell + M_\ell(\eta_\ell - D_\ell\psi_\ell)$
  $\eta_{\ell+1} \leftarrow P_\ell^H(\eta_\ell - D_\ell\psi_\ell)$
  $\psi_{\ell+1} \leftarrow \text{FGMRES}(D_{\ell+1}, \eta_{\ell+1}) + \text{K-Cycle}(\ell+1, \cdot)$ {`FGMRES for matrix` $D_{\ell+1}$ `and r.h.s.` $\eta_{\ell+1}$`, preconditioned in each iteration with K-Cycle`$(\ell+1, \cdot)$`}`
  $\psi_\ell \leftarrow \psi_\ell + P_\ell\psi_{\ell+1}$
  for $i = 1$ to ν
    $\psi_\ell \leftarrow \psi_\ell + M_\ell(\eta_\ell - D_\ell\psi_\ell)$

be regarded as a standard W-cycle with adaptive re-combination of the approximate solutions after each recursion.

8.2 Adaptive Multilevel Setup Procedure

For the construction of the aggregation-based Γ_5-preserving interpolation operators P_ℓ, and with them the coarse-level operators $D_{\ell+1} = P_\ell^H D_\ell P_\ell$, the two-level setup from Section 7.1 has to be extended to a multilevel setup. In order to preserve the Γ_5-symmetry on all levels, we use a block-spin structure for the interpolation operators on all levels. The number N_ℓ of test vectors used on each level is allowed to vary, and typically $N_\ell \leq N_{\ell+1}$ for all levels ℓ. The setup process that we found to work best in practice is divided into two phases:

1. An initial phase given as Algorithm 20 which constructs an initial multilevel hierarchy solely based on the smoothing iteration starting with random test vectors.

2. An iterative phase which imitates inverse iteration, illustrated in Fig. 8.2. In each step, we improve the test vectors by approximately applying D^{-1} using the current multilevel method and then update the multilevel hierarchy using the improved test vectors. This means that we perform n_{inv} calls to Algorithm 21 with $k_1 = 1$.

Algorithm 20: initial_setup_phase(ℓ), we assume $N_{\ell-1} \leq N_\ell$

input: ℓ, N_j for $j = \ell, \ldots, L-1$
output: $v_j^{(1)}, \ldots, v_j^{(N_\ell)}$, P_j, D_{j+1} for $j = \ell, \ldots, L-1$

1 **if** $\ell = 1$ **then**
2 choose N_ℓ random test vectors $v_\ell^{(1)}, \ldots, v_\ell^{(N_\ell)}$
3 **else**
4 **for** $j = 1$ to $N_{\ell-1}$
5 $v_\ell^{(j)} \leftarrow P_{\ell-1}^H v_{\ell-1}^{(j)}$ {restricted test vectors from previous level}
6 **for** $j = N_{\ell-1} + 1$ to N_ℓ
7 choose $v_\ell^{(j)}$ as a random test vector
8 **for** $\eta = 1$ to 3
9 **for** $j = 1$ to N_ℓ
10 $x = 0$
11 **for** $i = 1$ to η
12 $x \leftarrow x + M_\ell(v_\ell^{(j)} - D_\ell x)$ {M_ℓ smoother for system with matrix D_ℓ}
13 $v^{(j)} = x$
14 construct P_ℓ and set $D_{\ell+1} = P_\ell^H D_\ell P_\ell$
15 **if** $\ell < L - 1$ **then**
16 initial_setup_phase($\ell + 1$) {perform Algorithm 20 on next level}

During the development of our multilevel setup, we also tried simpler setup strategies like Algorithm 21 without the recursive call in line 9, i.e., just replacing the test vectors by the results from the K-cycles on each level. For this approach we observed that the interpolation operators P_ℓ on the intermediate levels $\ell < L - 1$ were not of satisfactory quality, in a sense that $D_{\ell+1}$ was not ill-conditioned enough since spec($D_{\ell+1}$) did not resemble the crucial part of spec(D_ℓ). However, when using additional setup iterations on these levels, we obtain efficient interpolation operators P_ℓ on each level.

Again we will give our specific choices for the various parameters in the setup in Section 8.4 which contains the numerical results.

8.3 Implementation Details – Idling Processes

In Sections 3.5 and 4.4 we discussed implementation details for the construction of the coarse grid correction and the smoother, respectively. These ingredients

Algorithm 21: iterative_setup_phase(ℓ)

input: ℓ, N_j, k_j, $v_j^{(1)}, \ldots, v_j^{(N_\ell)}$, P_j, D_{j+1} for $j = \ell, \ldots, L-1$
output: $v_j^{(1)}, \ldots, v_j^{(N_\ell)}$, P_j, D_{j+1} for $j = \ell, \ldots, L-1$

```
1 if ℓ < L then
2     for i = 1 to k_ℓ
3         for j = 1 to N_ℓ
4             {ψ_m, m = ℓ, ..., L−1} ← K-Cycle(ℓ, v_{ℓ−1}^{(j)})   {results ψ_m of
                                                the last recursive call
                                                    on all levels m}
5             for m = ℓ to L − 1
6                 v_m^{(j)} = ψ_m/||ψ_m||          {update test vectors with
                                                the iterates of each level}
7         for m = ℓ to L − 1
8             construct P_m and D_{m+1}
9     iterative_setup_phase(ℓ + 1)                  {perform Algorithm 21
                                                        on next level}
```

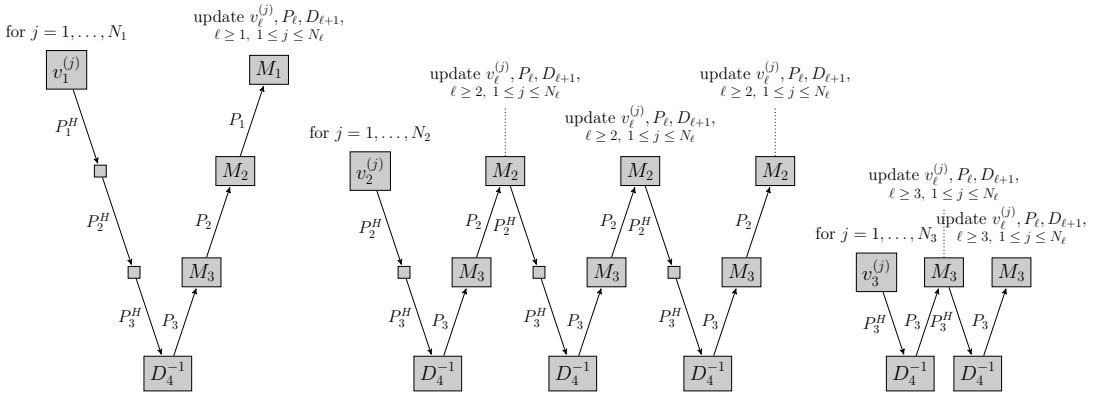

Figure 8.2: Illustration of Algorithm 21 as one setup iteration on level 1 with $L = 4$, $k_2 = 3$, $k_3 = 2$ and K-cycle length 1, i.e., a V-cycle.

forming the whole two-grid hierarchy can be implemented on coarser levels $\ell > 1$ requiring only minor adjustments. The Γ_5-preserving standard aggregation on the coarser levels again provides pairs of aggregates such that each pair shares a common lattice block and Γ_5 is constant on each aggregate.

As a new feature of the implementation and the only significant difference to the two-level approach, processes are now allowed to idle on the coarser levels. This is necessary for a parallel multigrid method since a high degree of paralleliza-

tion can cause a lack of lattice sites on the coarser levels. In our implementation we assume that sites which belong to a common aggregate always share the same process. This allows to perform the computational part of interpolation and restriction without communication, similarly to what was done in the two-level approach. Furthermore, we assume to have on each process at least one block of each color from the SAP block decomposition on the finest level (cf. Section 3.2). This ensures that no idle times occur when applying the smoother. As soon as on the coarser levels the processes do not own at least one SAP block of each color, or not an entire aggregate, dedicated processes (master processes) obtain the data belonging to neighboring processes (assistant processes), such that the previous assumptions hold for all master processes (illustrated in Figure 8.3). This ensures that the smoother, the restriction and the interpolation can be applied in the same manner as on the finer levels. During computations done by the master processes on the current level or on coarser ones, the assistant processes idle. In summary, we have a set of dedicated master processes, each of them owning a set of assistant processes. Each set of assistants belonging to a common master process is geometrically connected, and together with the master the union of their lattice blocks forms again a lattice block on the current level. The set of all masters joined with their assistants yields a coarser block decomposition of the current level lattice.

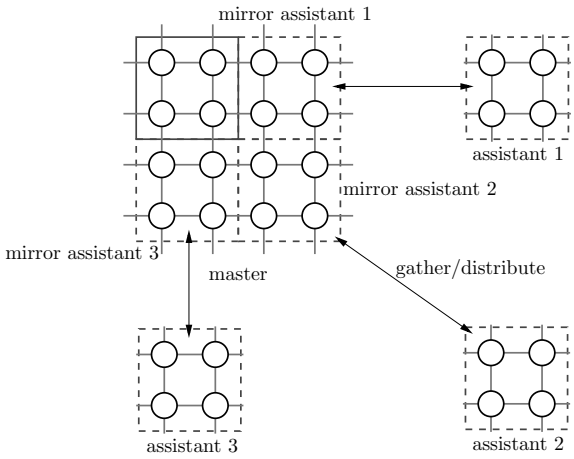

Figure 8.3: Illustration of a master and his assistants reduced to 2D.

After each application of the restriction the master processes gather the recently restricted information from their assistants and the assistants idle until the next application of the interpolation. Therein, the information computed on the current level master processes is distributed back to the respective assistants; afterwards the interpolation is performed. This principle is applied in a recursive fashion to the set of master processes on the consecutive coarser levels until the coarsest level is reached.

Figure 8.3 illustrates this master-assistant relation in a 2D setting, the master

	parameter		default
setup	number of iterations for $\ell = 1$	n_{inv}	6
	number of iterations for $\ell > 1$	k_ℓ	2
	number of test vectors for $\ell = 1$	N_1	20
	number of test vectors for $\ell > 1$	N_ℓ	30
	size of lattice-blocks for aggregates and SAP on level 1		4^4
	size of lattice-blocks for aggregates and SAP on level $\ell > 1$		2^4
	coarse system relative residual tolerance (stopping criterion for the coarse system)$^{(*)}$	ε	$5 \cdot 10^{-2}$
solver	restart length of FGMRES	n_{kv}	10
	relative residual tolerance (stopping criterion)	tol	10^{-10}
smoother	number of pre-smoothing steps$^{(*)}$	μ	0
	number of post-smoothing steps$^{(*)}$ on level 1	ν	1
	number of post-smoothing steps$^{(*)}$ on level $\ell > 1$		3
	number of Minimal Residual (MR) iterations to solve the local systems in SAP$^{(*)}$		4
K-cycle	maximal length$^{(*)}$		5
	maximal restarts$^{(*)}$		2
	relative residual tolerance (stopping criterion)$^{(*)}$		10^{-1}

Table 8.1: Parameters for multi-level DD-αAMG. $(*)$: same in solver and setup.

process owning his original lattice sub-block and mirrored lattice sub-blocks of all his assistants. Since a master process also needs to be able to apply the operator on all sites belonging to his gathered lattice block, the local part of the operator D_ℓ on the current level also has to be made available for these sites and therefore gathered from the set of assistants right after its construction.

8.4 Numerical Multilevel Results

We now report results of extensive numerical experiments for multilevel DD-αAMG with particular emphasis on the three- and four-level setting. As its two-level predecessor, the method is implemented as a parallel program in `C` using `MPI`, and we compute and apply the DD-αAMG preconditioner in single precision only. The outer FGMRES iteration remains in double precision. The system on the coarsest level is solved via odd-even preconditioned GMRES by requiring that the norm of the residual be reduced to ε times the norm of the initial residual.

Table 8.1 summarizes the default parameters used in our experiments. Note that these parameters are the same as those used for the two-level results in Section 7.2 except that we had to reduce the restart length of the outer FGMRES routine to n_{kv} from 25 to 10 due to memory limitations when using a small

ID	lattice size $N_t \times N_s^3$	pion mass m_π [MeV]	CGNR iterations	shift m_0	clover term c_{sw}	provided by
3	48×24^3	250	11,664	−0.09530	1.00000	BMW-c [39, 40]
5	48×48^3	136	53,932	−0.09933	1.00000	BMW-c [39, 40]
6	64×64^3	135	84,207	−0.05294	1.00000	BMW-c [39, 40]
7	128×64^3	270	45,804	−0.34262	1.75150	CLS [29, 44]
8	128×64^3	190	88,479	−0.33485	1.90952	CLS [29, 44]

Table 8.2: Configurations used together with their parameters. For details about their generation we refer to the references. Except for configuration ID 5, the pion masses are determined up to an accuracy of 5 MeV only. All BMW configurations were smeared with 3 steps of HEX [28] smearing, the CLS configurations were not smeared.

number of cores. This reduction of the cycle length has very little effect on the total number of iterations, and we never observed that passing from $n_{kv} = 25$ to $n_{kv} = 10$ augmented the total iteration count by more than 1. For consistency we therefore use $n_{kv} = 10$ in all numerical experiments. In our numerical experiments we saw that for SAP and the aggregates on the coarser levels $\ell > 1$ a lattice-block size of 2^4 is the best choice in terms of runtime performance, and larger block-sizes would hamper the use of higher degrees of parallelism. A lattice-block of size 4^4 on level $\ell = 2$ on a single process would either require a fine grid process lattice-block size of at least 16^4, or it would cause that processes are idle already on the second finest level.

In Table 8.2 we give an overview of the configurations used in our tests. After the production of the numerical results of the previous chapter, we obtained another non-smeared configuration from CLS termed configuration ID 8, for which it is particularly challenging to solve the Wilson-Dirac equation. The iteration count of CGNR, i.e., CG applied to the system $D^H D z = D^H b$, can be used as an indicator for the conditioning of the respective operator. The configurations yield truly challenging linear systems encountered in state-of-the-art lattice QCD calculations. Note that, as in Section 7.2, we always used the clover improvement with the Sheikoleslami-Wohlert parameter c_{sw} given in Table 8.2.

In what follows, we explore the potential benefits of additional levels in the DD-αAMG method. Special focus is put on the consequences of additional levels in terms of the degree of parallelization, i.e., the size of the local lattice kept on each node. We test the performance of the DD-αAMG method for different numbers of levels using a variety of cost measures and we analyze the scaling behavior as a function of the bare mass m_0. Finally, we compare the DD-αAMG approach to the recently improved version of the inexact deflation approach which now allows for inaccurate projection [76, 79].

If not stated otherwise all results were obtained on Juropa [66] at JSC.

8.4.1 Parallel Multilevel Methods

Our first tests analyze the influence of the degree of parallelization on the performance of the two- and three-level method. This is an important issue since using many processes causes idle times on the coarser levels.

We tracked the performance of the solve phase of two- and three-level DD-αAMG method for the configurations with IDs 3, 5 and 6 using different degrees of parallelization, reported as the size of the local lattice on each core. Our goal is to find the sweet spot with respect to consumed core minutes.

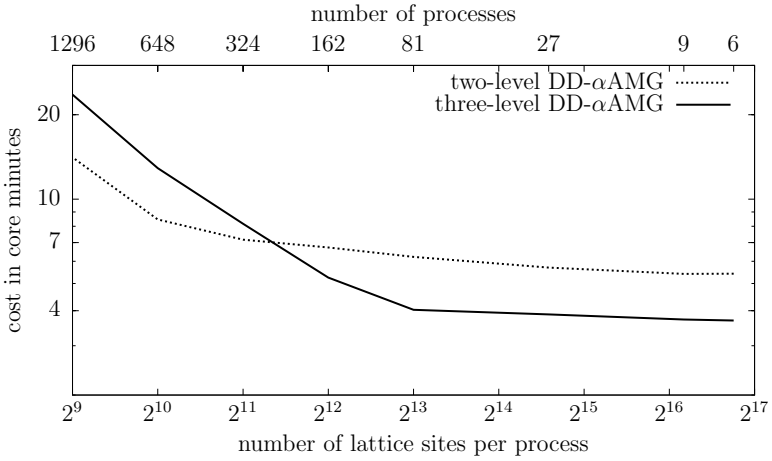

Figure 8.4: Estimation of the sweet spot (Table 8.2, configuration ID 3).

Figure 8.4 plots the dependence of the solver performance in terms of core minutes on the degree of parallelization for the rather small configuration with ID 3 and lattice size 48×24^3. Due to memory restrictions the minimal number of processors that can be used is 6 corresponding to $110{,}592 = 8 \times 24^3$ lattice sites per processor. On the other end of the horizontal axis we are limited to 1,296 processors which corresponds to $2^9 = 8 \times 4^3$ lattice sites per process on level one. This is the minimum number of lattice sites which is required for processes not to idle in red-black SAP with a block size of 4^4 on the finest level, since then each process holds exactly one red and one black block.

Figure 8.4 shows that as soon as we use $8^4 = 2^{12}$ or fewer lattice sites per process on level one, the three-level method performs worse than the two-level method. This is not surprising, since for a local lattice size of 8^4 on level one we obtain a local lattice size of 2^4 on level two. Although in the three-level method, smoothing on level two is performed with SAP with blocks of size 2^4 only, each process now holds just one block, and thus half of the processes will idle during the SAP iteration. This phenomenon carries over to the third level, where (we there take aggregates of size 2^4, see Table 8.1) we then have just one lattice site per process, resulting in idle times when performing odd-even preconditioned GMRES as the solver on this last level. Similarly, for a local lattice size of 2^{11}

on level one, 3 out of 4 processes are idling on levels two and three. For a local lattice size of 2^{10} and 2^9 this increases to 7 out of 8 and 15 out of 16 processes, respectively. We conclude that for the relatively small configuration ID 3 the additional third level is advisable only if a relatively small degree of parallelism is used.

On level three we end up with a global lattice size of 6×3^3 for which odd-even preconditioning is not possible. For a fair comparison we therefore switched off odd-even preconditioning everywhere for all tests involving configuration ID 3.

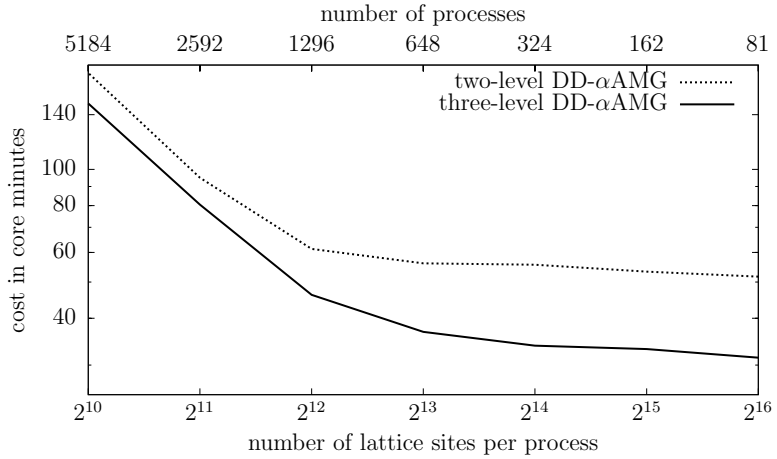

Figure 8.5: Performance of two- and three-level DD-αAMG (Table 8.2, configuration ID 5).

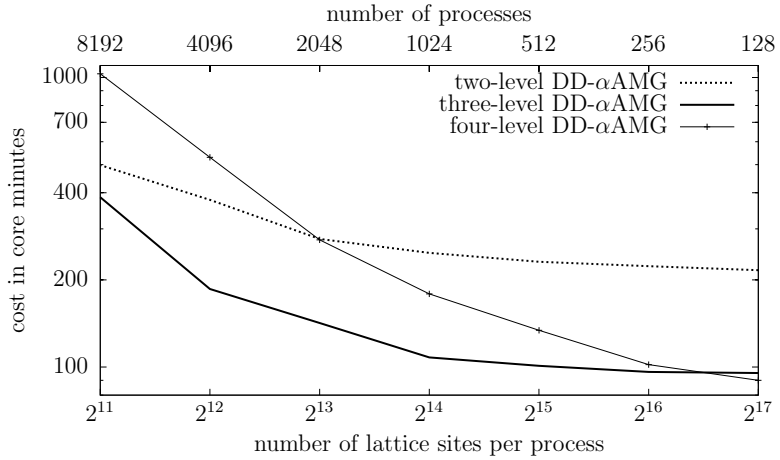

Figure 8.6: Performance of two-, three- and four-level DD-αAMG for estimation of the sweet spot (Table 8.2, configuration ID 6).

The picture changes when we investigate the same dependence for larger configurations. In Figures 8.5 and 8.6 we see that using three levels improves over two levels on the whole range for the number of processes. For both configurations

with IDs 5 and 6 we see that for two and three levels the cost in core minutes remains almost constant for roughly up to 1,000 processes, and that in this range the three-level method gains a factor of 1.6 and 2.3, respectively, over the two-level method. Using more than 1,000 processors degrades the performance in terms of core minutes for both methods, and the advantage of the three-level method becomes less apparent. For the four-level method we report results only for the larger configuration ID 6, and even there the method starts to degrade as soon as we use more than 128 processes, since then idling processes cannot be avoided on the last level. Only for the minimum number of processes the four-level method performs (slightly) better than the three-level method. On a different architecture, with more memory per core, which would allow to use fewer processes, the four-level method might achieve a more distinct speed-up over three levels.

8.4.2 Two, Three and Four Levels

Now we compare the time to solution (wall clock time) of the DD-αAMG method with two, three and four levels for the configurations with IDs 5 to 8 using only small numbers of processes, i.e., working with large local lattices.

	configuration ID	5	6	7	8
	lattice size	48×48^3	64×64^3	128×64^3	128×64^3
	pion mass m_π	136 MeV	135 MeV	270 MeV	190 MeV
two levels	setup time	272s	666s	535s	545s
	solve time	38.3s	101s	88.8s	108s
	solve iter	33	32	35	43
three levels	setup time	279s	562s	526s	548s
	solve time	23.3s	44.7s	46.4s	55.8s
	solve iter	30	27	33	36
four levels	setup time	–	594s	530s	548s
	solve time	–	42.2s	43.9s	52.5s
	solve iter	–	27	33	36
	processes	81	128	256	256
local lattice	level 1	16×16^3	32×16^3	32×16^3	32×16^3
	level 2	4×4^3	8×4^3	8×4^3	8×4^3
	level 3	2×2^3	4×2^3	4×2^3	4×2^3
	level 4	–	2×1^3	2×1^3	2×1^3

Table 8.3: Comparison of DD-αAMG with two, three and four levels for a small number of processes, parameters from Tables 8.1 and 8.2.

Table 8.3 shows that the three-level method outperforms the two-level method in terms of wall clock time in all tests by factors between 1.6 and 2.3 as far as the solver time is concerned. The time required to set up the multigrid hierarchy does not depend significantly on the number of levels used because in the multilevel case we use the emerging multigrid hierarchy to solve on the coarser levels more efficiently. For configuration ID 6 we even observe a slight reduction of the setup

time when going from 2 to 3 levels. The performance gain in solve time of the multi-level methods compared to the two-level method is a factor of more than two for all the larger lattices (configurations with IDs 6 to 8), and the four-level method performs best in all these cases. We expect the gain of the four-level method to grow further for even larger lattices. We also note that, for all configurations, the quality of multilevel DD-αAMG as a preconditioner—measured in terms of the number of iterations to reach the stopping criterion—improves when going from two to three levels but remains unchanged when going to four levels. This indicates that a multigrid solver, either using two or three levels, employed on the second level with a relative residual tolerance of 10^{-1} provides a better approximation to the solution on the second level than odd-even preconditioned GMRES with a tolerance of $5 \cdot 10^{-2}$. This observation will be numerically confirmed in Section 8.4.4.

	configuration ID	5		6		7		8	
	lattice size	48×48^3		64×64^3		128×64^3		128×64^3	
	pion mass m_π	135 MeV		135 MeV		270 MeV		190 MeV	
	levels	2	3	2	3	2	3	2	3
	setup time	12.3s	32.6s	20.6s	38.1s	22.2s	37.7s	24.1s	39.6s
	solve time	2.09s	1.74s	3.65s	2.83s	4.50s	3.04s	5.40s	3.90s
	solve iter	33	31	31	27	36	33	43	36
level 1	consumed time	0.34s	0.33s	0.59s	0.51s	1.46s	1.42s	1.79s	1.47s
	wait time$^{(*)}$	0.031s	0.022s	0.042s	0.038s	0.092s	0.095s	0.10s	0.075s
	allreduce time	0.065s	0.077s	0.072s	0.054s	0.22s	0.18s	0.26s	0.11s
level 2	consumed time	1.75s	0.40s	3.06s	0.53s	3.04s	0.56s	3.61s	0.67s
	wait time$^{(*)}$	0.19s	0.027s	0.20s	0.063s	0.077s	0.052s	0.066s	0.039s
	allreduce time	1.26s	0.088s	1.86s	0.132s	1.49s	0.20s	1.83s	0.12s
level 3	consumed time	–	1.01s	–	1.79s	–	1.06s	–	1.76s
	wait time$^{(*)}$	–	0.056s	–	0.10s	–	0.035s	–	0.061s
	allreduce time	–	0.42s	–	1.10s	–	0.72s	–	1.13s
summarized	total wait time$^{(*)}$	0.22s	0.11s	0.24s	0.20s	0.17s	0.18s	0.17s	0.18s
	total allreduce time	1.33s	0.59s	1.93s	1.29s	1.71s	1.10s	2.09s	1.36s
	processes	5,184	5,184	8,192	8,192	8,192	8,192	8,192	8,192
local lattice	level 1	$4^2 \times 8^2$	$4^2 \times 8^2$	4×8^3	4×8^3	8×8^3	8×8^3	8×8^3	8×8^3
	level 2	$1^2 \times 2^2$	4×2^3	1×2^3	4×2^3	2×2^3	4×2^3	2×2^3	4×2^3
	level 3	–	2×1^3	–	2×1^3	–	2×1^3	–	2×1^3

Table 8.4: Comparison of DD-αAMG with two and three levels for large numbers of processes, parameters as in Tables 8.1 and 8.2. $(*)$: Wait time for point-to-point communication.

As a complement to the results from Table 8.3, Table 8.4 gives timings for large numbers of processes. Here, idle times on the coarser levels occur. For example, for the number of processes chosen for the tests with the configurations with IDs 7 and 8 every second process idles on the second and third level within the three-level method. For configuration ID 6, three out of four and for configuration ID 5 even 7 out of 8 processes idle on the second and third level. For all four configurations there are no idle times for the two-level method. Though, using a

third level in all cases still leads to decreased solve times (see also Figures 8.5 and 8.6). The larger degree of parallelization reduces the speed-up factors of the three-level method over the two-level method, which are now down to approximately 1.3.

In addition, the setup for the three-level method becomes more expensive and now takes 2 to 3 times longer than for two levels. However, in situations where the increased setup time can be compensated for by solving many systems with several right hand sides, the three-level method will still pay off.

Table 8.4 also shows, for each level, the time of a non-idling process spent in working on that level including wait times due to nearest neighbor communication and global reduction operations. The wait times for the three-level methods appear to be reasonably small, showing that point-to-point communication even for next to nearest neighbor cores (on coarser grids when every other process idles) is negligible. The overall performance loss is mostly dominated by all-to-all communications which makes up a significant part of the time spent on the coarsest level in all cases. Looking at the three-level method for configuration ID 5—which represents the worst case in terms of processes idling—the time spent in computations on the levels two and three (to be derived from the table as the sum of consumed times minus the time spent in communication on these levels) amounts to 0.82 seconds. Due to 7 out of 8 processes idling on these levels, we see a significant performance loss of 0.72 seconds, i.e., 40% of the solve time. For the three-level method on configuration ID 6 the analogous calculation results in a comparatively smaller but still significant performance loss of 25%.

8.4.3 Comparison with BiCGStab

We now compare the two- and three-level method with a mixed precision implementation of odd-even preconditioned BiCGStab (as in Section 7.2.1), the standard Krylov subspace method for solving the Wilson-Dirac system. In addition to wall clock times, we also give the flop count for each method together with the performance (in terms of Gflop/s per process) achieved by our implementations. Table 8.5 shows the result for the configuration IDs 6 and 8. We first note that compared to BiCGStab the solve time for the three-level method is almost two orders of magnitude smaller. The flop count per lattice site as well as the overall core minute count are accordingly reduced. As far as the setup cost is concerned, even in the worst case where the setup is followed by just one solve, the total times of the two-level and three-level methods are still 5 to 8 times smaller than the time for one BiCGStab solve. It is noteworthy that three-level DD-αAMG performs with up to 2.59 Gflop/s per core, corresponding to 9.3% peak performance, on Juropa with a pure `C`-code, i.e., without any machine specific optimization. Our implementation of BiCGStab, which is based on the same degree of optimization, is somewhat less efficient and achieves only 1.90 to 1.95 Gflop/s. But even if we were to increase this by a factor of two, BiCGStab would

still be significantly slower than any of the DD-αAMG methods.

	configuration ID 6			configuration ID 8		
	three-level DD-αAMG	two-level DD-αAMG	BiCGStab	three-level DD-αAMG	two-level DD-αAMG	BiCGStab
processes	128	128	128	256	256	256
setup time	562s	666s	–	548s	545s	–
solve time	44.7s	101s	3,859s	55.8s	108s	4,846s
consumed core minutes	95.4	216	8,230	235	460	20,700
consumed Mflop per site	0.89	2.17	57.4	1.00	2.21	70.3
Gflop/s per core	2.59	2.81	1.95	2.39	2.68	1.90

Table 8.5: Comparison of two- and three-level DD-αAMG with BiCGStab, parameters from Tables 8.1 and 8.2.

Combined with the results reported in Section 8.4.2, Table 8.5 suggests that in situations where run-time is less important than the consumed core minutes and where several Dirac systems with different right hand sides must be solved, e.g., in configuration analysis, it is best to run a multi-level method with a low degree of parallelization. This gives the largest number of system solves per core minute. On the other hand, in a situation, where run-time is of utmost importance, e.g., in the generation of configurations within the HMC process, it might be advisable to use only a two-level method and a high degree of parallelization.

8.4.4 Scaling Tests

An important motivation for the development of multilevel preconditioners for the Wilson-Dirac system is the removal of "critical slowing down", i.e., the observed dramatic increase in iterations for standard iterative methods when the mass parameter approaches its critical value. In the next set of tests we report solve times of the two-, three- and four-level DD-αAMG method with respect to the bare mass m_0 for configuration ID 6. These tests use 128 cores, i.e., a low degree of parallelization. They represent the continuation of the mass scaling tests in Section 7.2.3.

We see in Figure 8.7 that the time to solution of BiCGStab increases much more rapidly than the time to solution for the two-level method when approaching the critical bare mass m_{crit}. Using a third level additionally yields a significant improvement in the scaling behavior. Furthermore, we see that in the regime of m_{ud} the four-level method starts to outperform the three-level method. Note that for the computation of observables in lattice QCD it might actually be necessary to solve the Dirac equation for mass parameters beyond m_{ud}, e.g, to account for the up-down quark mass difference [18, 43]. For configuration ID 6 this requires a mass parameter $m_u = -0.05347$.

Approaching the critical mass parameter makes the Wilson-Dirac operator more ill-conditioned which explains why the number of iterations in BiCGStab

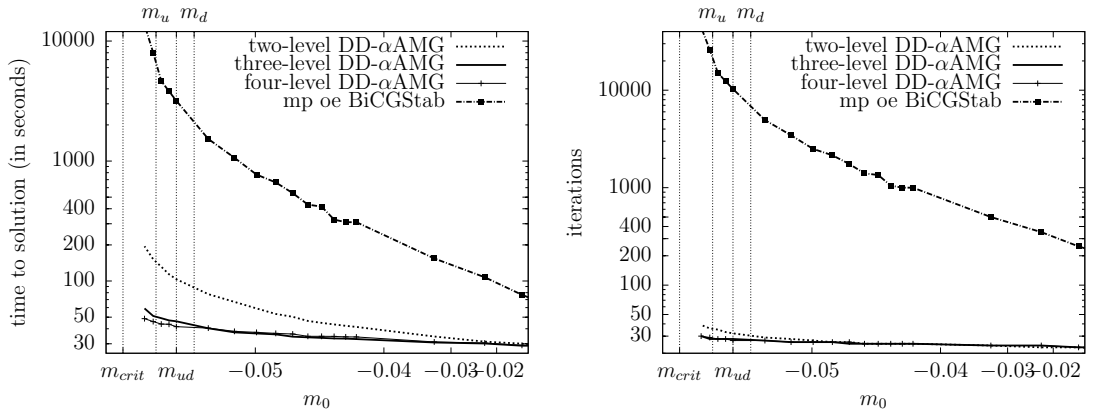

Figure 8.7: Scaling of BiCGStab and DD-αAMG with the bare mass m_0. In here, $m_{ud} = -0.05294$ denotes the physical mass parameter for which the configuration was thermalized, and $m_{crit} = -0.05419$ [39, 40] is the critical mass.

grows so tremendously. For multilevel DD-αAMG, the number of iterations grows relatively mildly as m_0 decreases. The observed increase in time to solution is mainly due to the fact that we have to invest more work on the coarsest level, where we solve the coarsest system D_L using odd-even preconditioned GMRES. Just as D, the matrix D_L becomes more ill-conditioned as m_0 decreases.

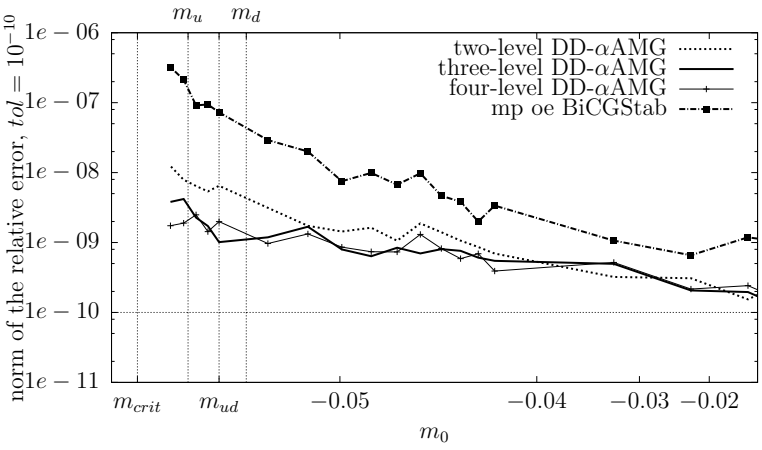

Figure 8.8: Scaling of the final error obtained from DD-αAMG and BiCGStab.

From the fundamental relation $De = r$ between the error e and the residual r of a solution of a linear system with matrix D, we have the relation $\|e\| \leq \|D^{-1}\| \, \|r\|$. For ill-conditioned matrices, $\|D^{-1}\|$ is large, so it may well be that the error is still large while the residual is already small. We therefore performed numerical experiments where we tracked the norm of the relative error $\|e\|/\|\psi^*\| = \|\psi^* - \psi\|/\|\psi^*\|$ with ψ the computed approximate solution as a function of m_0 for a pre-determined solution ψ^*. In the same fashion as in [92], ψ^* is

chosen as the inverse of a point source to a relative accuracy of 10^{-10} such that the right hand side $\eta = D\psi^*$ is approximately a point source again. This should result in linear systems with comparable degrees of difficulty for all methods. As always, the iterations were stopped once the initial residual was decreased by a factor of $tol = 10^{-10}$. Figure 8.8 shows that the error is usually significantly smaller than $\|D^{-1}\| \, \|r\|$ for all quark masses and for all methods. We calculated $\|D^{-1}\|_2 = \lambda_{\min}(\Gamma_5 D)^{-1} = 1{,}776$ for configuration ID 6 at $m_0 = m_{ud}$. This shows that the error e can be up to three orders of magnitude less accurate than the residual r and this is what we observe for BiCGStab at $m_0 = m_{ud}$. However, in the range of the physical masses there is a gain in accuracy of about one order of magnitude when going from BiCGStab to the two-level method and almost another order of magnitude when going to the three- or four-level method. This can be explained as follows:

The low mode components of the error fulfill $\|e\| \approx \|D^{-1}\| \, \|r\|$. By construction, the error of DD-αAMG has only small contributions of the low modes, as they are removed by the coarse grid correction, whereas the error of BiCGStab is dominated by low modes after the first few iterations. Thus, we can expect that $\|e\| \ll \|D^{-1}\| \, \|r\|$ for the error of DD-αAMG.

Combining the observations of Figures 8.7 and 8.8 results in an even more pronounced gap between DD-αAMG and BiCGStab as already observed when considering convergence to relative error instead of relative residual norm.

8.4.5 Reducing the Impact of Communication in Coarse Grid Solvers

Depending on the size of the configurations, the system the algorithm is run on and the number of levels used in the method, communication becomes a bottleneck. In here the main problem can be global communication (cf. Table 8.4 in Section 8.4.2) for inner products as well as local communication in matrix-vector multiplications (to be shown in this section). In this section we present measures that we implemented in order to dampen the impact of communication on the performance of the algorithm.

Global Communication

As already indicated by Table 8.4 in Section 8.4.2, situations might arise where the time spent in communication on the coarsest grid starts to dominate the solve time. For instance, when using a two-level method on configuration ID 5 the all-to-all communication time already amounts to 64% of the solve time. Thus, we expect the situation to become worse when doing computations with larger configurations on larger machines. The source of allreduce operations on the coarsest grid is found at the heart of the Krylov subspace method used to solve this system. The Arnoldi process (cf. Algorithm 22) calculates an orthonormal basis

V_m of the Krylov subspace. In every iteration i, the current iterate $v_{i+1} = D_c v_i$ is orthogonalized against the previous ones $v_1, \ldots, v_i$ and then normalized, here this is done by classical Gram-Schmidt (GS; see, e.g., [14,109]). For larger Krylov subspace dimensions m, classical GS tends to be numerically unstable in the sense that a loss of orthogonality can occur. In order to avoid this, modifications like modified GS (see, e.g., [14, 109]), re-orthogonalization (see, e.g., [14]) and many other variants were introduced. In modified GS the non-orthogonal component of each vector v_j is subtracted from v_{i+1} before one orthogonalizes against the next vector v_{j-1}, i.e., one computes

$$v_{i+1} \leftarrow v_{i+1} - (v_j^\dagger v_{i+1}) v_j \quad \text{for} \quad j = 1, \ldots, i\,.$$

This procedure introduces smaller round off errors than classical GS, and, as opposed to classical GS, the method is backward stable (see, e.g., [64]). In parallel computations, however, all i inner products would have to be calculated one after the other which leads to a cost of $\mathcal{O}(m^2)$ global communications for computing V_m. In classical GS, which we employ in Algorithm 22, the contributions of v_j, $j = 1, \ldots, i$ are subtracted from v_{i+1} at once (cf. line 4). Thus, the communication of i inner products can be bundled to one global communication leading to m global communications for inner products and another m for vector norms, i.e., V_m is computed at the cost of $\mathcal{O}(m)$. Note that the computational costs are equal for both approaches.

When numerical instabilities in parallel computations occur, it can be more beneficial to employ one step of re-orthogonalization instead of modified GS. This means that each computed vector v_i is again orthogonalized against all vectors $v_1, \ldots, v_{i-1}$ via classical GS. In other words, the orthogonalization process is applied twice, and for moderately conditioned problems it has been shown that "twice is enough", see [14, 94]. The orthonormalization costs are doubled (for communication and for computation), but it gives us a better complexity of global communications which remains at $\mathcal{O}(m)$.

For Krylov subspace dimensions like $m = 30$ in our coarse grid solves, we

Algorithm 22: $(V_m, H) = \text{Arnoldi}(v_1, m)$

1 $v_1 \leftarrow v_1 / ||v_1||$
2 **for** $i = 1$ to $m - 1$
3 $\quad z \leftarrow D_c v_i$
4 $\quad h_{j,i} \leftarrow v_j^\dagger z \quad \text{for} \quad j = 1, \ldots, i$
5 $\quad v_{i+1} \leftarrow z - \sum_{j=1}^{i} h_{j,i} v_j$
6 $\quad h_{i+1,i} \leftarrow ||v_{i+1}||$
7 $\quad v_{i+1} \leftarrow v_{i+1} / h_{i+1,i}$
8 return $V_m = (v_1, \ldots, v_m)$ and $H = (h_{j,i})$

did not observe any significant numerical instabilities. Thus, we employ classical GS in the Arnoldi procedure as displayed in Algorithm 22 with two global communications per iteration, one in line 4, and one in 6.

A simple modification found in [52] cuts the number of global communications down to one by computing the norm in line 6 as

$$h_{i+1,i} = \sqrt{z^\dagger z - \sum_{j=0}^{i} v_j^\dagger z}\,, \tag{8.1}$$

thus making it possible to calculate it together with the block inner products in one allreduce operation. Even with this simple trick, the problem remains that every iteration of GMRES requires at least one global communication. There are two general approaches to attack this remaining problem.

First, one can reduce the amount of global communication by equipping the Krylov subspace solver on the coarsest grid with a polynomial preconditioner at the cost of additional matrix-vector multiplications, cf. [100]. The simplest idea to construct such a polynomial preconditioner is to run GMRES the first time the coarsest grid is visited and then, using the Arnoldi relation

$$AV_m = V_{m+1}H\,,$$

to extract the information required to build the polynomial which interpolates $f(t) = 1/t$ in the harmonic Ritz-values λ_k, $k = 1, \ldots, m$ obtained from H (see, e.g., [36]) This polynomial is then used every time the coarsest grid is visited again; cf. [100]. Note that the degree m of the polynomial can be any value smaller than the maximum dimension of the Krylov subspace used in the first solve.

After the computation of the harmonic Ritz-values λ_k, $k = 1, \ldots, m$, the corresponding residual polynomial is given by

$$p(t) = \prod_{j=0}^{m-1}(1 - \frac{t}{\lambda_k})\,. \tag{8.2}$$

We choose the initial guess $x^{(0)} = 0$ and the residual $r^{(0)}$ equal to current iterate v_i from the Arnoldi process. After the application of the polynomial preconditioner the residual is given by

$$r^{(m)} = \prod_{k=0}^{m-1}(I - \frac{1}{\lambda_k}D_c)r^{(0)} = p(D_c)r^{(0)}\,.$$

Thus, the residual update in each "intermediate" iteration k of the polynomial preconditioner satisfies

$$r^{(k+1)} = r^{(k)} - \frac{1}{\lambda_k}D_c r^{(k)}\,,$$

and the corresponding approximate solution is computed via

$$x^{(k+1)} = x^{(k)} + \frac{1}{\lambda_k} r^{(k)} = x^{(k)} + \frac{1}{\lambda_k}(r^{(0)} - D_c x^{(k)}) \,.$$

As state-of-the-art approach to protract the occurrence of numerical instabilities in (8.2), we apply a Leja point ordering to the Ritz values λ_k, cf. [96].

The second approach to deal with global communication is the use of so-called pipelined Krylov subspace methods as proposed in [52]. These algorithms are mathematically equivalent to their original non-pipelined counterparts, the differences rather being of algorithmic and thus numerical kind. It is only possible to use them if the system software allows for non-blocking global communication, which is not always the case. In this approach, the current allreduce and subsequent matrix-vector multiplications are computed at the same time in a pipelined fashion.

polynomial degree	#coarse operator applications	#coarse allreduces
–	3698 (100%)	3698 (100%)
9	5179 (140%)	611 (17%)
19	5180 (140%)	320 (9%)
29	5179 (140%)	286 (8%)
39	4710 (127%)	188 (5%)
49	4636 (125%)	172 (5%)
59	6048 (164%)	249 (7%)

Table 8.6: Comparison of coarse operator applications and the number of allreduces on the coarse grid for a range of polynomial degrees in a two-level method on configuration ID 6. The numbers in brackets denote the gain/reduction in percent compared to the numbers for the unpreconditioned coarse grid solver, parameters from Tables 8.1 and 8.2.

Table 8.6 gives results from a two-level method equipped with polynomial preconditioners with different polynomial degrees. We study the effect of the respective polynomial preconditioners on the number of coarse grid operator applications and the number of allreduces on the coarse grid where the norm of the relative residual was to be decreased by a factor of $5 \cdot 10^{-2}$. We observe that polynomials of degree 39 and 49 as preconditioners lower the number of allreduces on the coarse grid by a factor of 20, conceding a factor of less than 1.3 in coarse operator applications. Higher polynomial degrees do not have to yield additional benefit since the stopping criterion for the coarse grid solver can be already reached while the preconditioner is still evaluated to its full polynomial degree, thus causing unnecessary coarse operator applications. In addition to that, the polynomial still can become ill-conditioned for higher degrees. This seems to be the case for the degree 59 as the number of allreduces increases significantly. We measured the orthogonality of the Krylov subspace basis V_m when setting up the polynomial and did not observe any instabilities.

In order to give meaning to these numbers, let us consider an example how the reductions reported in Table 8.6 would translate to savings in the overall solve time. Unfortunately, we can at this point of time only give a theoretical calculation instead of actual measurements due to the fact that Juropa was shut down in the meantime, and using its successor Jureca we do not observe large communication times. This is also due to the computational part of the code not being optimized for this machine, yet.

In Table 8.4, the two-level method for configuration ID 5 required 2.09 seconds to converge, of which 1.75 seconds were spent on the coarse grid, split into 1.26 seconds for allreduces and 0.49 seconds for all other operations. The results shown in Table 8.4 were produced without making use of (8.1), thus the number of coarse allreduces is twice the number of coarse operator applications. Theoretically, making use of (8.1) we would obtain 0.63 seconds for coarse allreduces, 1.12 seconds coarse grid time and 1.46 seconds solve time. Assuming that the remaining 0.49 seconds on the coarse grid are mainly spent on coarse operator applications, a polynomial of degree 49 would result in 0.03 seconds for coarse allreduces and 0.61 seconds for other operations on the coarse grid, together 0.64 seconds which is a reduction of 43% on the coarse grid and 33% for the total solve.

In a hypothetical extreme case where the time spent in a solve is composed of 80% for coarse allreduces, 10% for the coarse operator and 10% for other operations, i.e.,

$$\tau = \tau_{cip} + \tau_{cop} + \tau_{other} = 80\% + 10\% + 10\%\,,$$

applying a polynomial preconditioner of the same quality as we achieved for the degree 49 would reduce the solve time to

$$\tau_{prec} = \frac{\tau_{cip}}{20} + \frac{5\tau_{cop}}{4} + \tau_{other} = 26.5\%\,.$$

Local Communication

Our collaborators from BMW-c [72] implemented a two-level method with GMRES as a smoother and optimized it for Juqueen at JSC, an IBM BlueGene/Q system with a 5D torus network and 28,672 nodes, each node contains an IBM PowerPC A2 processor with 16 1.6 GHz cores, i.e., a total of 458,752 cores. Up to now, the BMW-c implementation is only used for the calculation of observables where the same system is solved for many right hand sides, and setup time is no bottleneck. Consequently, the interpolation is built using eigenvector approximations. The communication bottleneck on this machine is the latency of the nearest neighbor communication on the coarse grid when using a large number of cores rather than the global communication on this grid. This is due to the fact that the number of coarse lattice sites approaches the minimal number supported.

To overcome this in observable calculation, the solver has been implemented as a block solver for many right hand sides. A chosen amount of right hand sides, typically 12, is solved at the same time and instead of communicating ghost cells for each vector one after the other in a matrix-vector multiplication, ghost cells for the same directions are bundled and communicated together. Naturally the all-to-all communications are blocked in a similar fashion.

	1 right hand side			12 right hand sides		
cores	solve time	coarse operator time	coarse nearest neighbor comm	solve time	coarse operator time	coarse nearest neighbor
32,768	0.317s	0.0458s	0.141s	1.35s	0.446s	0.136s
16,384	0.401s	0.0515s	0.145s	2.33s	0.629s	0.212s
8,192	0.647s	0.103s	0.150s	4.35s	1.08s	0.247s
4,196	1.42s	0.335s	0.180s	9.04s	2.53s	0.367s

Table 8.7: Comparison of the BMW-c implementation for 1 and 12 right hand sides on configuration ID 6, 5 iterations of GMRES as a smoother, 24 iterations of GMRES as a coarse grid solver, 10^{-7} relative residual decrease for the outer solver, other parameters from Tables 8.1 and 8.2. Note that these numbers were obtained from single runs and thus timings may be inaccurate, especially for larger numbers of cores. We acknowledge [72] for the computation of these results.

Table 8.7 displays the strong scaling behavior of solve time, time spent in applications of the coarse operator and time spent in nearest neighbor communication for 1 right hand side and for treating 12 right hand sides at once.

On the one hand, when doubling the number of cores in the 1 right hand side case, the speed-up in solve time and coarse operator time almost stagnates. The nearest neighbor communication remains almost constant, indicating that it is dominated by the latency on the coarse grid. It dominates the time spent on the coarsest grid when using 8,192 cores or more, and in fact it represents a significant part of the solve time as well.

On the other hand, the 12 right hand sides case shows almost perfect strong scaling for solve time and coarse operator time, and the time spent in nearest neighbor communication is not significant anymore. We also note that solving for 12 right hand sides in the shown cases always required less than 7 times the solve time for 1 right hand side. This is due to the bundled communication but also due to the fact that the code is vectorized over the right hand sides, i.e., low level optimized for the case when using several right hand sides.

8.4.6 Low Level Optimization

In the lattice QCD community a lot of effort is put into porting code to different platforms and its optimization. Based on our DD-αAMG implementation, an optimized version has been developed for the Intel Xeon-Phi processor, it is pub-

licly available [63]. In order to demonstrate the effects of low-level optimization we re-integrated the vectorized modules from [63] into our code and established a low effort SSE-optimization that already shows a decent improvement.

		DD-αAMG	SSE-optimized DD-αAMG
two levels	setup time	666s	453s
	solve time	101s	81.1s
	solve iter	32	31
three levels	setup time	562s	337s
	solve time	44.7s	30.7s
	solve iter	27	28
four levels	setup time	594s	341s
	solve time	42.2s	29.5s
	solve iter	27	29

Table 8.8: Comparison of DD-αAMG and SSE-optimized DD-αAMG with two, three and four levels on configuration ID 6 using 128 processes, parameters from Tables 8.1 and 8.2.

In Table 8.8 we give numbers for the SSE-optimized version of our DD-αAMG implementation. We did not spend much effort on optimizing the SAP smoother on the fine grid, its optimization is rather difficult since the small 3-by-3 gauge link matrices $U_\mu(x)$ on the fine grid are not suitable for the SSE-register length (4 single precision values). However, the number of test vectors N_ℓ on each level can be chosen such that the coupling matrices of size $2N_\ell \times 2N_\ell$ as well as the interpolation and restriction operators fit well to the register length. Thus, optimization on the coarser levels is comparatively simple. The different degrees of optimization are displayed in the results. For two levels we only observe a reduction of 20% in solve time and 32% in setup time, whereas for three and four levels we see a more substantial reduction of 30% in solve time and 40% in setup time. The solver iterations of the optimized DD-αAMG code slightly deviate from the non-optimized code due to two changes. First, we employed a mixed precision FGMRES as the outer solver so that we could restrain ourselves to SSE-optimize the double precision part. Second, the SSE-optimization requires different data layouts. The different operation execution orderings ultimately result in different round-off effects. Besides that we make use of the cheaper matrix-vector multiplication described in Section 3.5 which results in different execution orderings as well. The latter in combination with a mixed precision Krylov solver is used in [75] as well.

8.4.7 Comparison with Inexact Deflation with Inaccurate Projection

Recently the implementation of inexact deflation was upgraded within the openQCD code [76]. The new version of inexact deflation, termed "inexact deflation with inaccurate projection" was inspired by our work published in [49] and is now similar in spirit to algebraic multigrid. It differs from (two-level) DD-αAMG in its construction of the interpolation and the coarse-level operator. In the inexact deflation approach Γ_5-symmetry is not preserved on the coarse level. Moreover, in inexact deflation the multigrid principle is not applied recursively. Rather, the coarse system is solved using an explicitly deflated GCR variant, where the projected test vectors $P^H v_1, \ldots, P^H v_N$ are used to deflate the coarse system, cf. Section 5.1. We also implemented this for DD-αAMG but in terms of runtime performance it did not pay off.

In order to account for this recent upgrade of the openQCD code, we compare it with multilevel DD-αAMG. As the openQCD code[3] uses open boundary conditions in time direction we cannot directly use our set of configurations with this code. Thus, we integrated the new modules containing the inexact deflation with inaccurate projection method into the DD-HMC-code [75] and then compared both methods.

	three-level DD-αAMG	two-level DD-αAMG	inexact deflation (DD-HMC-1.2.2) [75]	inexact deflation with inaccurate projection [76]
test vectors N	20	20	20	32
setup iter	6	6	10	6
post-smoothing iter ν	1	1	5	5
setup time	562s	666s	681s	897s
solve iter	27	32	48	18
solve time	44.7s	101s	223s	96.6s

Table 8.9: Comparison using 128 processes (Table 8.2, configuration ID 6), parameters from Tables 8.1 and 8.2.

Table 8.9 shows that the solve times for inexact deflation with inaccurate projection for configuration ID 6 is 2.3 times less than without inaccurate projection, and also slightly faster than two-level DD-αAMG. Due to the different approaches in constructing prolongation operators, inexact deflation ends up with only half as many variables and consequently a coarse-grid operator with four times fewer nonzero entries (in a matrix-representation) if the same number of test vectors is used. We found that the inexact deflation with inaccurate projection approach

[3]Version 1.2 of the openQCD code [76] supports only open boundary conditions but the later version 1.4, that was published after the publication of these results, supports several types of boundary conditions.

works best if we use more test vectors than in the DD-αAMG setup, rather 30 instead of 20.

Even with the larger number of test vectors, the cost of one iteration on the second level is smaller in inexact deflation with inaccurate projection. However, in DD-αAMG the third level substantially lowers the overall amount of work needed to solve the second level system which explains the superiority of three-level DD-αAMG over all two-level methods.

Inexact deflation with inaccurate projection needs the largest setup time of all approaches, while it is minimal for three-level DD-αAMG. The results for the Γ_5-preserving interpolation show that the recursive extension of DD-αAMG works and, in particular, that the SAP smoother still works on coarse levels. It is unclear whether this is also the case when Γ_5-symmetry is not preserved on the coarse level, so that a recursive extension of the inexact deflation with inaccurate projection approach might not benefit as much from additional levels.

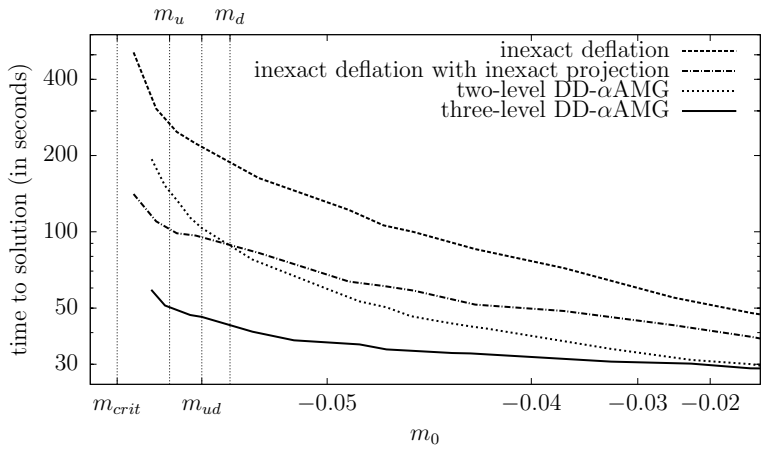

Figure 8.9: Scaling with the bare mass m_0 of inexact deflation and DD-αAMG (Table 8.2, configuration ID 6) using 128 processes.

As the behavior of solvers approaching the critical mass is an important benchmark, we also compare the mass scaling of the inexact deflation with inaccurate projection method with multilevel DD-αAMG. Figure 8.9 shows the scaling behavior for all the methods considered in Table 8.1 as a function of the bare mass m_0. Inexact deflation with inaccurate projection and two-level DD-αAMG show a similar scaling behavior until m_{ud}. Beyond this mass, two-level DD-αAMG tends to scale similarly to the ordinary inexact deflation approach. The upgrade of the inexact deflation method to use inaccurate projection leads to a better scaling behavior, which is similar to that of three-level DD-αAMG. Thus, the ability to solve the coarse system inexactly and the ability to use more test vectors and still having a cheap coarse-level operator can lead to a significantly different scaling behavior. The overall best performance and scaling behavior is achieved by three-level DD-αAMG, where the third level already pays off for heavier masses than

m_d and is expected to pay off even more in the future when even larger lattices will be used.

	4 level AMG	4 level DD-αAMG	speed-up factor
smoother iter	6	1	
setup iter	6	6	
setup time	815s	541s	1.51
solve iter	23	27	
solve time	60.6s	41.4s	1.46
total time	876s	582s	1.51

Table 8.10: Comparison of four-level DD-αAMG with four-level AMG (Table 8.2, configuration ID 6), parameters from Table 7.2, 128 cores.

8.4.8 GMRES/GCR Smoothing and "AMG"

In Section 7.2.5 we compared AMG (cf. Chapter 6) and DD-αAMG for the two-level case. The "emulated" tests in a common framework showed the SAP smoother being 1.5 times faster than GMRES for an identical outer iteration count for both methods. Due to a costly coarse grid correction, this resulted in an overall speed-up of just 1.2 for a DD-αAMG solve over an AMG solve.

However, when we move to more than two levels, the algorithm spends a larger percentage of its time smoothing on the various levels so that the advantage of SAP over GMRES should become more visible. Table 8.10 confirms this. In this setting, employing a low degree of parallelism, the four-level method performed best with respect to solve time for both smoothers, SAP and GMRES. We achieve the target speed-up factor of 1.5 (cf. smoother time in Table 7.11 in Section 7.2.5) in setup and solve.

Chapter 9

Preconditioning the Overlap Dirac Operator

When discretizing the continuum Dirac equation

$$(\mathcal{D} + m)\psi = \eta \tag{9.1}$$

it is desirable to preserve four properties of $\mathcal{D}$: local coupling, invariance under global gauge transformations, chiral symmetry [10, 82], and no species doubling [107, 115]. It was proven in [89] that a discretization of $\mathcal{D}$ cannot fulfill all four properties at the same time. This result is known as the "Nielsen-Ninomiya no-go theorem". Consequently, there exists a variety of different discretizations preserving different properties.

The overlap operator is a discretization of $\mathcal{D}$ that respects chiral symmetry, an important property of the continuum operator which is violated by the Wilson discretization. Chiral symmetry is of vital importance for some physical observables like hadron spectra in the presence of magnetic fields, for example. From a practical point of view, the overlap operator has the disadvantage that its computational cost can be two orders of magnitude larger than when using standard discretizations, the reason being that the operator contains a matrix function which has to be evaluated iteratively when applying the operator to a vector.

In this chapter we use DD-αAMG to solve systems with the overlap operator. Surely, DD-αAMG cannot be transferred and applied directly. The basic idea is to use a standard Wilson discretization of the Dirac equation to form a preconditioner for the overlap operator. This may be regarded as a variant of the fictitious (or auxiliary) space preconditioning technique [85] that has been used for developing and analyzing multilevel preconditioners for various nonconforming finite element approximations of PDEs; cf. [93, 116]. In this context, one works with a mapping from the original space to a fictitious space, yielding an equivalent problem that is easier to solve. Preconditioning is then done by (approximately) solving this equivalent problem. The convergence properties of

auxiliary space preconditioning depend on the choice of the fictitious space, and its computational efficiency depends, in addition, on the efficiency of the solver used in that space; cf. [85].

For the overlap operator in lattice QCD, choosing its kernel—the Wilson-Dirac operator—as the auxiliary space preconditioner is facilitated by the fact that both operators are defined on the same Hilbert space. In this way, the preconditioner for the former can be constructed using DD-αAMG for the latter on the same finite dimensional lattice. We note that similar approaches are possible for other QCD discretizations. For example, the direct and strong coupling of the Wilson blocks used in the 5d domain wall operator [69] suggest that a similar Wilson auxiliary-space preconditioner (with a more general mapping) may also be effective.

We demonstrate that the technique we develop in this chapter is able to reduce the computational cost for solving systems with the overlap operator substantially, reaching speed-ups of a factor of 10 or more in realistic settings. The preconditioning technique thus contributes to making the overlap operator more tractable in lattice QCD calculations. The results of this chapter were published in [24].

9.1 The Overlap Discretization

Definition 9.1
For two quark fields φ and ψ the Dirac fermion action of mass m *in the continuum is given as*

$$\begin{aligned} S &= \int_{\mathbb{R}^4} \varphi^H(x)\,(\mathcal{D}+m)\,\psi(x)\,dx \\ &= \int_{\mathbb{R}^4} \varphi^H(x)\left(\sum_{\mu=0}^{3}\gamma_\mu\otimes(\partial_\mu+A_\mu(x))+m\right)\psi(x)\,dx \end{aligned}$$

where $\mathcal{D}$ denotes the continuum Dirac operator from (2.2)*. For the massless case $m=0$,* chiral symmetry *is the invariance of S under transformations*

$$\varphi_c(x)^H \to \varphi_c(x)^H e^{-i\alpha\gamma_5} \quad \text{and} \quad \psi_c(x) \to e^{-i\alpha\gamma_5}\psi_c(x) \tag{9.2}$$

where $\alpha\in\mathbb{R}$ and $\psi_c(x)$ denotes the spin degrees of freedom of $\psi(x)$ for a fixed color index c (cf. Section 2.1*).*

Note that for $m=0$ the invariance of S holds due to $\gamma_5\gamma_\mu=-\gamma_\mu\gamma_5$ (cf. Lemma 2.10) and

$$\begin{aligned} & \varphi_c(x)^H e^{-i\alpha\gamma_5}\gamma_\mu e^{-i\alpha\gamma_5}\psi_c(x) \\ = \;& \varphi_c(x)^H e^{-i\alpha\gamma_5}\gamma_\mu\left(\sum_{k=0}^{\infty}\tfrac{(-i\alpha\gamma_5)^k}{k!}\right)\psi_c(x) \\ = \;& \varphi_c(x)^H e^{-i\alpha\gamma_5}\left(\sum_{k=0}^{\infty}\tfrac{(-i\alpha\gamma_5)^k(-1)^k}{k!}\right)\gamma_\mu\psi_c(x) \\ = \;& \varphi_c(x)^H e^{-i\alpha\gamma_5}e^{i\alpha\gamma_5}\gamma_\mu\psi_c(x) \\ = \;& \varphi_c(x)^H\gamma_\mu\psi_c(x)\,. \end{aligned}$$

This also implies that (9.2) is equivalent to that $\mathcal{D}$ anti-commutes with $\gamma_5 \otimes I_3$, i.e.,

$$(\gamma_5 \otimes I_3)\mathcal{D} + \mathcal{D}(\gamma_5 \otimes I_3) = 0$$

in a sense that

$$(\gamma_5 \otimes I_3)\mathcal{D}\psi(x) + \mathcal{D}(\gamma_5 \otimes I_3)\psi(x) = 0$$

for all matter fields ψ and space-time points x. For further details about chiral symmetry, we refer to [10, 82]. As was pointed out in [77], a lattice discretization D of $\mathcal{D}$ which obeys the Ginsparg-Wilson relation [53]

$$\Gamma_5 D + D\Gamma_5 = aD\Gamma_5 D \tag{9.3}$$

satisfies an appropriate lattice variant of chiral symmetry. It has long been unknown whether such a discretization exists until Neuberger constructed it in [86]. For convenience, the essentials of the arguments in [86] are summarized in the following proposition and its proof. As a shorthand we define

$$D_W(m) := D_W + mI\,.$$

Proposition 9.2
Neuberger's overlap operator

$$D_N = \frac{1}{a}\left(\rho I + D_W(m_0^{ker})\Big(D_W(m_0^{ker})^H(D_W(m_0^{ker})\Big)^{-\frac{1}{2}}\right)$$

fulfills (9.3) *for* $\rho = 1$*, has local discretization error* $\mathcal{O}(a)$*, and is a stable discretization provided* $-2 < m_0^{ker} < 0$*.*

Proof. We write $\mathcal{D}_\mathcal{L}$ for the restriction of the continuum Dirac operator $\mathcal{D}$ to the lattice $\mathcal{L}$, i.e., $\mathcal{D}_\mathcal{L}$ is the finite dimensional operator which takes the same values as $\mathcal{D}$ at the points from $\mathcal{L}$. The fact that the Wilson-Dirac operator has first order discretization error can then be expressed as[4]

$$\mathcal{D}_\mathcal{L} = D_W(0) + \mathcal{O}(a),$$

implying

$$\mathcal{D}_\mathcal{L} + \frac{m_0}{a}I = D_W(m_0) + \mathcal{O}(a) \tag{9.4}$$

for any mass parameter m_0.

To construct D_N we first note that any operator $\widehat{D}$ that is Γ_5-symmetric and fulfills (9.3) can be parametrized by

$$a\widehat{D} = I + \Gamma_5 S, \tag{9.5}$$

[4]For simplicity, we consider here the "naive" limit $a \to 0$. In the full quantum theory one has $\mathcal{D}_\mathcal{L} = D_W(m_0(a)) + \mathcal{O}(a)$ with the mass $m_0(a)$ of order $1/\log(a)$; see [82].

with $S^H = S$ and $S^2 = I$. Both conditions are fulfilled for

$$S = \Gamma_5 D_W(m_0^{ker})\Big(D_W(m_0^{ker})^H(D_W(m_0^{ker})\Big)^{-\frac{1}{2}}, \quad -m_0^{ker} \in \mathbb{R} \setminus \mathrm{spec}(D_W(0)).$$

Using (9.4) we obtain

$$S \;=\; \Gamma_5\Big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\Big)\Big(\big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)^H\big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)\Big)^{-\frac{1}{2}}.$$

Since $\mathcal{D}$ is anti-self-adjoint, we have $\mathcal{D}_\mathcal{L}^H = -\mathcal{D}_\mathcal{L}$ and thus

$$\begin{aligned}
&\Big(\big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)^H\big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)\Big)^{-\frac{1}{2}} \\
&= \tfrac{a}{|m_0^{ker}|}\Big(\big(\tfrac{a}{m_0^{ker}}\mathcal{D}_\mathcal{L} + I + \mathcal{O}(a^2)\big)^H\big(\tfrac{a}{m_0^{ker}}\mathcal{D}_\mathcal{L} + I + \mathcal{O}(a^2)\big)\Big)^{-\frac{1}{2}} \\
&= \tfrac{a}{|m_0^{ker}|} I + \mathcal{O}(a^3),
\end{aligned}$$

which in turn yields

$$S = \Gamma_5\Big(\frac{a}{|m_0^{ker}|}\mathcal{D}_\mathcal{L} + \mathrm{sign}(m_0^{ker}) I + \mathcal{O}(a^2)\Big). \tag{9.6}$$

Combining (9.6) with (9.5) we find

$$a\widehat{D} = I + \frac{a}{|m_0^{ker}|}\mathcal{D}_\mathcal{L} + \mathrm{sign}(m_0^{ker}) I + \mathcal{O}(a^2)\,,$$

so that for $m_0^{ker} < 0$ we have

$$\widehat{D} = \frac{1}{|m_0^{ker}|}\mathcal{D}_\mathcal{L} + \mathcal{O}(a).$$

This shows that $\widehat{D}$ is a first order discretization of $\mathcal{D}$. For it to be stable one has to choose $-2 < m_0^{ker} < 0$, a result for which we do not reproduce a proof here, referring to [86] instead. □

Note that $D_N = \widehat{D} + \frac{\rho-1}{a} I$, so $\rho - 1$ sets the quark mass (see (9.1)) up to a re-normalization factor.

Using the Wilson-Dirac operator as the kernel in the overlap operator is the most popular choice, even though other kernel operators have been investigated as well [31]. Neuberger's overlap operator has emerged as a popular scheme in lattice QCD over the years. In the literature one often writes

$$D_N = \rho I + \Gamma_5 \mathrm{sign}\big(\Gamma_5 D_W(m_0^{ker})\big) \tag{9.7}$$

with sign denoting the matrix extension of the sign function

$$\mathrm{sign}(z) = \begin{cases} +1 & \text{if } \mathrm{Re}(z) > 0 \\ -1 & \text{if } \mathrm{Re}(z) < 0 \end{cases}.$$

We note that $\mathrm{sign}(z)$ is undefined if $\mathrm{Re}(z) = 0$. Since $\Gamma_5 D_W(m_0)$ is hermitian, see Section 2.2, the matrix $\mathrm{sign}(\Gamma_5 D_W(m_0^{ker}))$ is hermitian as well. Since $\Gamma_5^2 = I$, we also see that the overlap operator satisfies the same Γ_5-symmetry as its kernel D_W,

$$\big(\Gamma_5 D_N\big)^H = \Gamma_5 D_N. \tag{9.8}$$

We end this section with a characterization of the spectrum of the overlap operator.

Lemma 9.3
The overlap operator D_N is normal. Its spectrum is symmetric to the real axis and part of the circle with midpoint ρ and radius 1, i.e.,

$$\lambda \in \mathrm{spec}\big(D_N\big) \Rightarrow \overline{\lambda} \in \mathrm{spec}\big(D_N\big) \text{ and } |\lambda - \rho| = 1.$$

Proof. We first remark that the sign function is its own inverse and that the operator $\Gamma_5 D_W(m_0)$ is hermitian (cf. Lemma 2.10). This implies that $\mathrm{sign}(\Gamma_5 D_W(m_0))$ is its own inverse and hermitian, thus unitary. Its product with the unitary matrix Γ_5 is unitary as well, implying that all its eigenvalues have modulus one. As a unitary matrix, this product is also normal. The term ρI in (9.7) preserves normality and shifts the eigenvalues by ρ.

It remains to show that $\mathrm{spec}(D_N)$ is symmetric with respect to the real axis, which follows from the Γ_5-symmetry (9.8) of the overlap operator in the same manner as in Lemma 2.14. □

For the purposes of illustration, Figure 9.1 gives the spectra of the Wilson-Dirac operator and the overlap operator for a 4^4 lattice. There, as everywhere else from now on, we set $a = 1$ which is no restriction since a^{-1} enters D_W simply as a linear scaling. For this small configuration all 3,072 eigenvalues and the sign function can be computed with standard methods for full matrices.

In the previous chapters m_0 was chosen as a negative number such that the spectrum of D_W lies in the right half plane with some eigenvalues being close to the imaginary axis. The choice for m_0 when $D_W(m_0)$ appears in the kernel of the sign function is different (namely smaller, see Proposition 9.2).

9.2 A Preconditioner Based on the Wilson-Dirac Operator

The spectral gaps to be observed as four disks with relatively few eigenvalues in the left part of Figure 9.1 are typical for the spectrum of the Wilson-Dirac operator and become even more pronounced as lattice sizes are increased or smearing (cf. Section 2.3) is applied. In practice, the mass parameter m_0 that appears

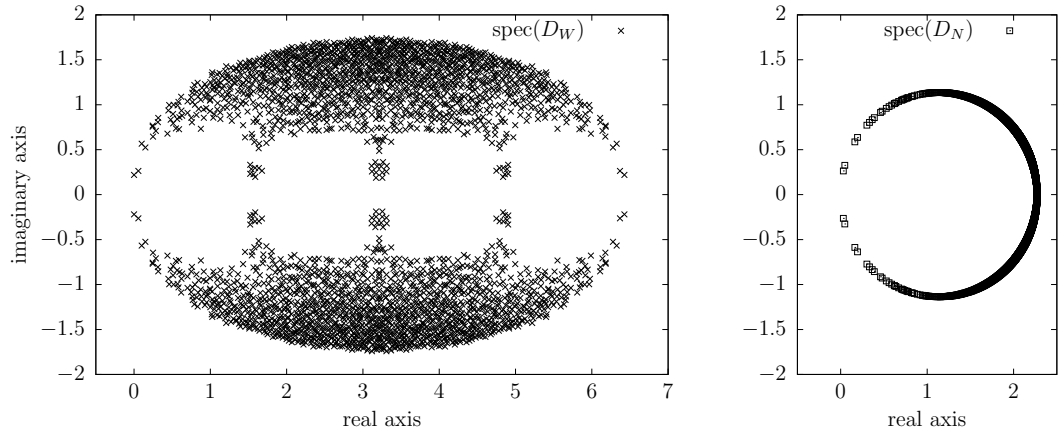

Figure 9.1: Typical spectra of the Wilson-Dirac and the overlap operator for a 4^4 lattice.

in the definition of the kernel $D_W(m_0^{ker})$ of the overlap operator is chosen such that the origin lies in the middle of the leftmost of these disks. For this choice of m_0^{ker} we now motivate why the Wilson-Dirac operator $D_W(m_0^{prec})$ with adequately chosen mass m_0^{prec} provides a good preconditioner for the overlap operator.

To do so we investigate the connection of the spectrum of the overlap operator and the Wilson-Dirac operator in the special case that $D_W(0)$ is normal. This means that $D_W(0)$ is unitarily diagonalizable with possibly complex eigenvalues, i.e.,

$$D_W(0) = X\Lambda X^H, \text{ with } \Lambda \text{ diagonal and } X \text{ unitary.} \tag{9.9}$$

Trivially, then, $D_W(m_0)$ is normal for all mass parameters m_0 and

$$D_W(m_0) = X(\Lambda + m_0 I)X^H. \tag{9.10}$$

To formulate the resulting non-trivial relation between the eigenvalues of D_N and its kernel $D_W(m_0^{ker})$ in the theorem below we use the notation csign(z) for a complex number z to denote its "complex" sign, i.e.,

$$\text{csign}(z) = z/|z| \text{ for } z \neq 0.$$

The theorem works with the singular value decomposition $A = U\Sigma V^H$ of a matrix A in which U and V are orthonormal, containing the left and right singular vectors as their columns, respectively, and Σ is diagonal with non-negative diagonal elements, the singular values. The singular value decomposition is unique up to choices for the orthonormal basis of singular vectors belonging to the same singular value, i.e., up to transformations $U \to UQ, V \to VQ$ with Q a unitary matrix commuting with Σ; cf. [56].

Theorem 9.4
Assume that $D_W(0)$ is normal, so that $D_W(m)$ is normal as well for all $m \in \mathbb{C}$,

and let X and Λ be from (9.9)*. Then we have*

$$D_N = X\big(\rho I + \mathrm{csign}(\Lambda + m_0 I)\big)X^H. \tag{9.11}$$

Proof. Let

$$\Gamma_5 D_W(m) = W_m \Delta_m W_m^H \text{ with } \Delta_m \text{ diagonal}, W_m \text{ unitary}, \tag{9.12}$$

be the eigendecomposition of the hermitian matrix $\Gamma_5 D_W(m)$. We have two different representations for the singular value decomposition of $\Gamma_5 D_W(m)$,

$$\begin{array}{rcll} \Gamma_5 D_W(m) & = & \big(\Gamma_5 X \mathrm{csign}(\Lambda + mI)\big) \cdot |\Lambda + mI| \cdot X^H & \text{(from (9.10))}, \\ \Gamma_5 D_W(m) & = & \big(W_m \mathrm{sign}(\Delta_m)\big) \cdot |\Delta_m| \cdot W_m^H & \text{(from (9.12))}. \end{array}$$

Thus, there exists a unitary matrix Q such that

$$W_m = XQ \text{ and } W_m \mathrm{sign}(\Delta_m) = \Gamma_5 X \mathrm{csign}(\Lambda + mI)Q. \tag{9.13}$$

Using the definition of D_N in (9.7), the relations (9.13) give

$$\begin{array}{rcl} D_N & = & \rho I + \Gamma_5 \mathrm{sign}(\Gamma_5 D_m) \\ & = & \rho I + \Gamma_5 W_m \mathrm{sign}(\Delta_m) W_m^H \\ & = & \rho I + \Gamma_5 \Gamma_5 X \mathrm{csign}(\Lambda + mI) Q (VQ)^X \\ & = & X(\rho I + \mathrm{csign}(\Lambda + mI) X^H. \end{array}$$

□

We remark that as an implicit consequence of the proof above the eigenvectors of $\Gamma_5 D_W(m) = \Gamma_5 D_W(0) + m\Gamma_5$ do not depend on m. Thus, if D_W is normal, Γ_5 and $\Gamma_5 D_W$ admit a basis of common eigenvectors.

The result in (9.11) implies that $D_N = \rho I + \Gamma_5 \mathrm{sign}(\Gamma_5 D_W(m_0^{ker}))$ and $D_W(0)$ share the same eigenvectors and that

$$\mathrm{spec}(D_N) = \{\rho + \mathrm{csign}(\lambda + m_0^{ker}), \lambda \in \mathrm{spec}(D_W(0))\}.$$

Taking $D_W(m_0^{prec})$ as a preconditioner for D_N, we would like eigenvalues of D_N which are small in modulus to be mapped to eigenvalues close to 1 in the preconditioned matrix $D_N D_W(m_0^{prec})^{-1}$. Since $D_W(m_0^{prec})$ and D_N share the same eigenvectors, the spectrum of the preconditioned matrix is

$$\mathrm{spec}\big(D_N D_W(m_0^{prec})^{-1}\big) = \Big\{\frac{\rho + \mathrm{csign}(\lambda + m_0^{ker})}{\lambda + m_0^{prec}}, \lambda \in \mathrm{spec}(D_W(0)\Big\}.$$

For $\omega > 0$ and $m_0^{prec} = \omega\rho + m_0^{ker}$, the mapping

$$g : \mathbb{C} \to \mathbb{C}, z \mapsto \frac{\rho + \mathrm{csign}(z + m_0^{ker})}{z + m_0^{prec}}$$

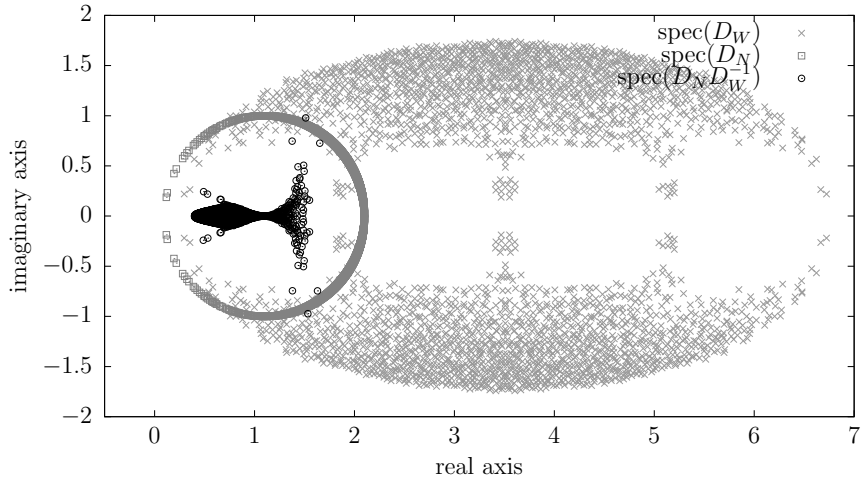

Figure 9.2: Spectra for a configuration of size 4^4.

sends $C(-m_0^{ker}, \omega)$, the circle with center $-m_0^{ker}$ and radius ω, to one single value $\frac{1}{\omega}$. We thus expect $D_W(m_0^{prec})$ to be a good preconditioner if we choose m_0^{prec} in such a manner that the small eigenvalues of $D_W(m_0^{prec})$ lie close to $C(-m_0^{ker}, \omega)$. Let $\sigma_{\min} > 0$ denote the smallest real part of all eigenvalues of $D_W(0)$. Assuming for the moment that $\sigma_{\min}$ is actually an eigenvalue, this eigenvalue will lie exactly on $C(-m_0^{ker}, \omega)$ if we have

$$\omega = \omega^{def} := -m_0^{ker} - \sigma_{\min} \text{ and thus } m_0^{prec} = m_0^{def} := \omega^{def}\rho + m_0^{ker}. \qquad (9.14)$$

For physically relevant parameters, ω^{def} is close to 1. We will take m_0^{def} from (9.14) as our default choice for the mass parameter when preconditioning with the Wilson-Dirac operator, although a slightly larger value for ω might appear adequate in situations where the eigenvalues with smallest real part come as a complex conjugate pair with nonzero imaginary part.

Although $D_W(0)$ is non-normal in physically relevant situations, we expect the above reasoning to also lead to an effective Wilson-Dirac preconditioner in these settings, and particularly so when the deviation of $D_W(0)$ from normality becomes small. Figure 9.2 shows the spectrum for the preconditioned matrix with the choice (9.14) for m_0^{prec} for the same 4^4 configuration as in Figure 9.1. The matrices in these tests are not normal, nonetheless the spectrum of the preconditioned matrix tends to concentrate around 0.7.

In the normal case, the singular values are the absolute values of the eigenvalues, and the singular vectors are intimately related to the eigenvectors. This relation was crucial to the proof of Theorem 9.4. In the non-normal case, the relation (9.11), which uses the eigenvectors of $D_W(0)$, does not hold. For the sake of completeness we give, for the general, non-normal case, the following result which links the overlap operator to the singular value decomposition of its kernel $D_W(m)$.

Lemma 9.5
Let $\Gamma_5 D_W(m) = W_m \Delta_m W_m^H$ denote an eigendecomposition of the hermitian matrix $\Gamma_5 D_W(m)$, where Δ_m is real and diagonal and W_m is unitary. Then

(i) A singular value decomposition of $D_W(m)$ is given as

$$D_W(m) = U_m \Sigma_m V_m^H \text{ with } V_m = W_m, \Sigma_m = |\Delta_m|, U_m = \Gamma_5 W_m \text{sign}(\Delta_m).$$

(ii) The overlap operator with kernel $D_W(m)$ is given as

$$D_N = \rho I + \Gamma_5 \text{sign}\big(\Gamma_5 D_W(m)\big) = \rho I + U_m V_m^H.$$

Proof. Since $\Gamma_5^{-1} = \Gamma_5$, we have the factorization $D_W(m) = \Gamma_5 W_m \Delta_m W_m^H = \Gamma_5 W_m \text{sign}(\Delta_m) |\Delta_m| W_m^H$, in which $\Gamma_5 W_m \text{sign}(\Delta_m)$ and W_m are unitary and $|\Delta_m|$ is diagonal and non-negative. This proves (i). To show (ii), just observe that for the hermitian matrix $\Gamma_5 D_W(m)$ we have $\text{sign}(\Gamma_5 D_W(m)) = W_m \text{sign}(\Delta_m) W_m^H$ and use (i). □

9.3 Numerical Results

In this section we report numerical results obtained on relatively large configurations used in current simulations involving the overlap operator, detailed in Table 9.1. For studying the influence of the deviation of normality, we use configuration ID 9 which is available for different numbers $s = 0, \ldots, 6$ of stout smearing steps to the gauge field (see Section 2.3 and [83]). Note that s influences $\sigma_{\min}$, the smallest real part of all eigenvalues of $D_W(0)$. The given choice for m_0^{ker} as a function of $\sigma_{\min}$, used in $D_N = \rho I + \Gamma_5 \text{sign}(\Gamma_5 D_W(m_0^{ker}))$ places the middle of the first 'hole' in the spectrum of $D_W(m_0^{ker})$ to be at the origin. The configuration with ID 10 was obtained using 3 steps of HEX smearing, its results were published in [19]. The value $m_0^{ker} = -1.3$ is the one used in the simulation. The middle of the first 'hole' in $D_W(m_0^{ker})$ is thus close to but not exactly at the origin. To be

ID	lattice size $N_t \times N_s^3$	kernel mass m_0^{ker}	default overlap mass μ	smearing s	provided by
9	32×32^3	$-1 - \frac{3}{4}\sigma_{\min}$	0.0150000	$\{0, \ldots, 6\}$-stout [83]	generated from [34, 35]
10	32×32^3	-1.3	0.0135778	3HEX [28]	BMW-c [19]

Table 9.1: Configurations used together with their parameters. See the references for details about their generation.

in line with the conventions from [17], e.g., we express the parameter $\rho \geq 1$ used in the overlap operator D_N as

$$\rho = \frac{-\mu/2 + m_0^{ker}}{\mu/2 + m_0^{ker}},$$

where $\mu > 0$ is yet another, "overlap" mass parameter. In our experiments, we will frequently consider a whole range for μ rather than just the default value from Table 9.1. The default value for μ is chosen such that it fits to other physically interpretable properties of the respective configurations like, e.g., the pion mass m_π. For both sets of configurations used, m_π is approximately twice as large as the value observed in nature, and the ultimate goal is to drive m_π to its physical value, which very substantially increases the cost for generating the respective configurations. We would then use smaller values for μ, and the results of our experiments for such smaller μ hint at how the preconditioning will perform in future simulations at physical parameter values. Note that smaller values for μ make ρ become closer to 1, so D_N becomes more ill-conditioned.

All results were obtained on Juropa [66] at JSC. In the following experiments we always use 1,024 cores. In all tests, our code ran with roughly 2 Gflop/s per core which accounts to $8 - 9\%$ peak performance. The multigrid solver used to precondition with $D_W(m_0^{prec})$ (see below) performs at roughly 10% peak, i.e., 2.4 Gflop/s per core.

9.3.1 Accuracy of the Preconditioner and Influence of m_0^{prec}

In a first series of experiments we solve the system

$$D_N \psi = \eta \tag{9.15}$$

on the one hand without any preconditioning, using GMRES(100), i.e., restarted GMRES with a cycle length of 100. On the other hand, we solve the same system with GMRES(100) using D_W^{-1} as a (right) preconditioner. To solve the respective linear systems with D_W we use DD-αAMG. Any other efficient solver for Wilson-Dirac equations as, e.g., the "AMG" solver developed in [5, 23, 92] and discussed in Chapter 6, could be used as well. In our approach, preconditioning is done by iterating with DD-αAMG until the relative residual is below a prescribed bound ε^{prec}. The DD-αAMG setup has to be done only once for a given Wilson-Dirac operator D_W, so its cost becomes negligible when using DD-αAMG as a preconditioner in a significant number of FGMRES iterations. In all our experiments, the setup never exceeded 2% of the total execution time, so we do not report timings for the setup. The restart length for FGMRES is again 100.

Figure 9.3 presents results for configuration ID 9 with $s = 3$ stout smearing steps and the default overlap mass μ from Table 9.1. We scanned the values for m_0^{prec} in steps of 0.01 and report the number of iterations necessary to reduce the

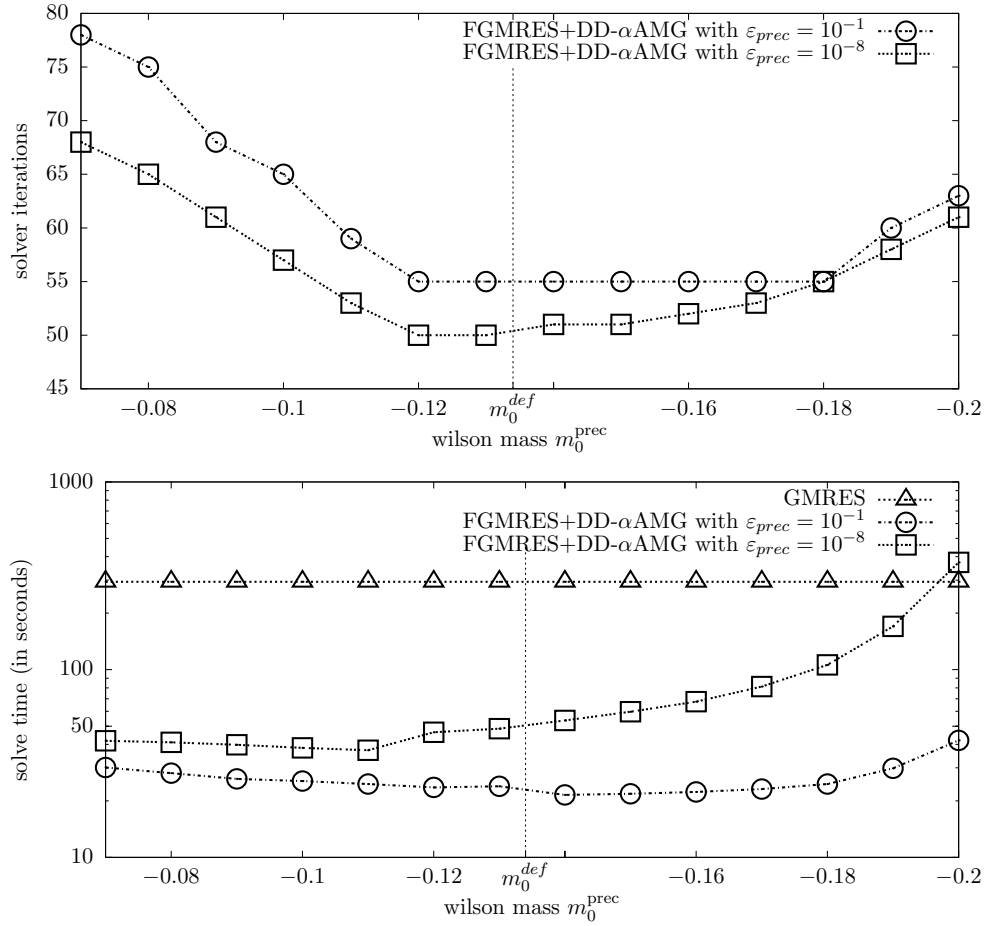

Figure 9.3: Preconditioner efficiency as a function of m_0^{prec} for two accuracies for the DD-αAMG solver (configuration ID 9, $s = 3$). Top: number of iterations, bottom: execution times.

initial residual by a factor of 10^{-8} for each of these values. We chose two different values ε^{prec} for the residual reduction required in the DD-αAMG iteration in the preconditioning. The choice $\varepsilon^{prec} = 10^{-8}$ asks for a relatively accurate solution of the systems with $D_W(m_0^{prec})$, whereas the choice $\varepsilon^{prec} = 10^{-1}$ requires an only quite low accuracy and thus only a few iterations of DD-αAMG. The upper part of Figure 9.3 shows that there is a dependence of the number of FGMRES iterations on m_0^{prec}, while at the same time there is a fairly large interval around the optimal value for m_0^{prec} in which the number of iterations required is not more than 20% larger than the minimum. These observations hold for both accuracy requirements for the DD-αAMG solver, $\varepsilon^{prec} = 10^{-8}$ and $\varepsilon^{prec} = 10^{-1}$. The number of iterations needed without preconditioning was 973.

The lower part of Figure 9.3 shows that similar observations hold for the execution times. However, the smaller iteration numbers obtained with $\varepsilon^{prec} =$

10^{-8} do not translate into smaller execution times, since the time for each DD-αAMG solve in the preconditioning is substantially higher as for $\varepsilon^{prec} = 10^{-1}$. This turned out to hold in all our experiments, so from now on we invariably report results for $\varepsilon^{prec} = 10^{-1}$. We also observe that the value of m_0^{def} from (9.14) lies within an interval in which iteration numbers and execution times (for both values for ε^{prec}) are quite close to the optimum. The execution time without preconditioning was 294 seconds.

Figure 9.4 reports results which show that the default value m_0^{def} is a fairly good choice in general. For two different configurations (no smearing and 3 steps of stout smearing) and a whole range of overlap masses μ, the plots at the top give the relative difference $\delta m_0 = (m_0^{opt} - m_0^{def})/m_0^{def}$ of the optimal value m_0^{opt} for m_0^{prec} and its default value from (9.14) as well as the similarly defined relative difference δiter of the corresponding iteration numbers. These results show that the iteration count for the default value m_0^{def} is never more than 15% off the best possible iteration count. The plot at the bottom backs these findings. We further scanned a whole range of smearing steps s at the default value for μ from Table 9.1, and the number of iterations with m_0^{def} is never more than 5% off the optimal value. The large values for δm_0 in the top right plot for $\mu = 2^{-3}$ are to be attributed to the fact that the denominator in the definition of δm_0, i.e., m_0^{def} is almost zero in this case.

These results suggest that (9.14) is indeed a good choice for m_0^{prec}. However, $\sigma_{\min}$ needed to compute m_0^{def} from (9.14) is not necessarily known a priori, and it may be more efficient to approximate the optimal value for m_0 "on the fly" by changing its value from one preconditioned FGMRES iteration to the next.

In order to minimize the influence of the choice of m_0^{prec} on the aspects discussed in the following sections we will always use the optimal m_0^{prec}, computed to a precision of .01 by scanning the range $[-\widetilde{\sigma}_{\min}, 0]$, where $\widetilde{\sigma}_{\min}$ is a rough guess at $\sigma_{\min}$ which fulfills $\widetilde{\sigma}_{\min} > \sigma_{\min}$. This guess can be easily obtained by a fixed number of power iterations to get an approximation for the largest real part $\widetilde{\sigma}_{\max}$ of an eigenvalue of D and then using the symmetry of the spectrum to obtain $\widetilde{\sigma}_{\min}$ by rounding $8 - \widetilde{\sigma}_{\max}$ to the first digit.

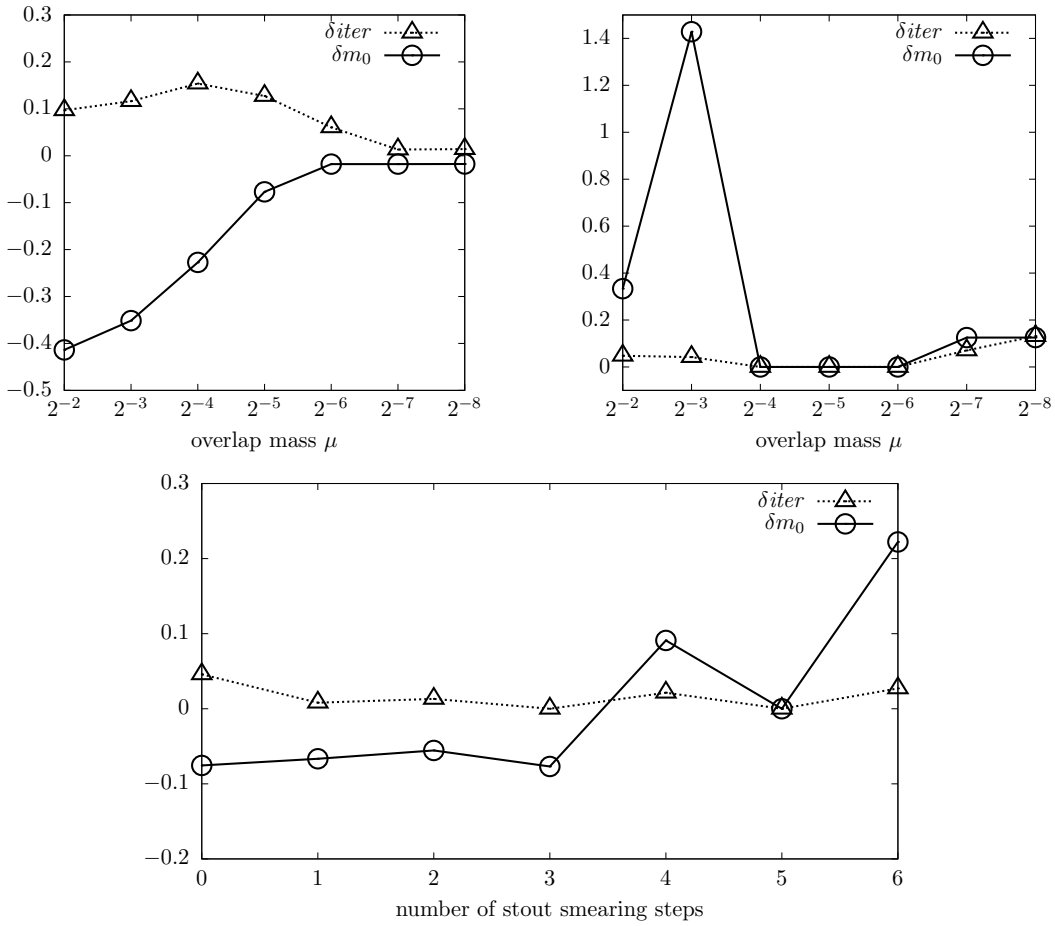

Figure 9.4: Quality of m_0^{def} without smearing (top left), with $s = 3$ steps of stout smearing (top right), and for $s = 0, \ldots, 6$ steps of stout smearing at fixed μ (bottom), configuration ID 9.

9.3.2 Quality and Cost of the Preconditioner

We proceed to compare in more detail preconditioned FGMRES(100) with unpreconditioned GMRES(100) in terms of the iteration count. As before, the iterations were stopped when the initial residual was reduced by a factor of at least 10^{-8}.

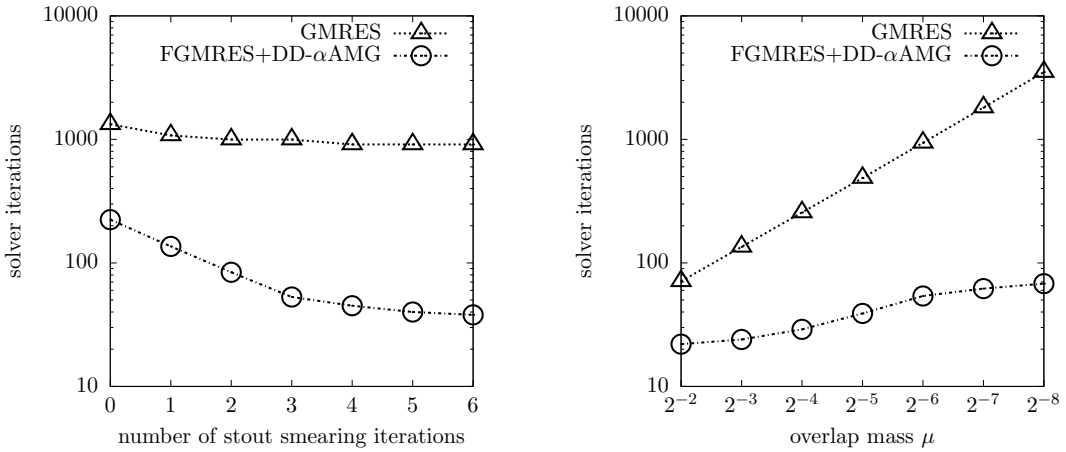

Figure 9.5: Comparison of preconditioned FGMRES(100) with unpreconditioned GMRES(100) (configuration ID 9). Left: dependence on the number of stout smearing steps s for default value for μ, cf. Table 9.1. Right: dependence on the overlap mass μ for $s = 3$.

Figure 9.5 gives this comparison, once as a function of the non-normality of the configuration, i.e., the number s of stout smearing steps applied, and once as a function of the overlap mass μ. We see that for the default value of μ from Table 9.1, the quality of the preconditioner increases with the number s of stout smearing steps, ranging from a factor of approximately 5 for $s = 0$ over 12 for $s = 3$ up to 25 for $s = 6$. We also see that the quality of the preconditioner increases as μ becomes smaller, i.e., when D_N becomes more ill-conditioned.

From the practical side, a comparison of the execution times is more important than comparing iteration numbers. Before giving timings, we have to discuss relevant aspects of the implementation in some detail.

Each iteration in GMRES or preconditioned FGMRES for (9.15) requires one matrix-vector multiplication with $D_N = \rho I + \Gamma_5 \mathrm{sign}(\Gamma_5 D_W)$. The matrix D_N is not given explicitly as it would be a full, very large matrix despite $\Gamma_5 D_W$ being sparse. Therefore, a matrix-vector multiplication $D_N \chi$ is obtained via an additional "sign function iteration" which approximates $\mathrm{sign}(\Gamma_5 D_W)\chi$ as part of the computation of $D_N \chi$. For this sign function iteration we use the restarted Krylov subspace method proposed recently in [45, 46] which allows for thick restarts of the Arnoldi process and has proven to be among the most efficient methods to approximate $\mathrm{sign}(\Gamma_5 D_W)\chi$. The sign function iteration then still represents the

by far most expensive part of the overall computation.

A first approach to reduce this cost, see [30], is to use relaxation in the sense that one lowers the (relative) accuracy $\varepsilon_{\text{sign}}$ of the approximation as the outer (F)GMRES iteration proceeds. The theoretical analysis of inexact Krylov subspace methods in [103,112] shows that the relative accuracy of the approximation to the matrix-vector product at iteration k should be in the order of $\varepsilon_{outer}/\|r_k\|$ (with r_k the (F)GMRES residual at iteration k) to achieve that at the end of the (F)GMRES iteration the initial residual be decreased by a factor of ε_{outer}. We used this relaxation strategy in our experiments with an additional factor of 10^{-2}, i.e., we assert $\varepsilon_{\text{sign}} < \varepsilon_{outer} \cdot 10^{-2}/\|r_k\|$ in order to prevent the iteration from stagnating when r_k approaches ε_{outer}.

A second commonly used approach, see e.g. [42, 54, 111], to reduce the cost of the sign function iteration is deflation. In this approach the k smallest in modulus eigenvalues $\lambda_1, \ldots, \lambda_k$ and their normalized eigenvectors $\xi_1, \ldots, \xi_k$ are precomputed once. With $\Xi = [\xi_1 | \ldots | \xi_k]$ and $\Pi = I - \Xi\Xi^H$ the orthogonal projector on the complement of these eigenvectors, $\text{sign}(\Gamma_5 D_W)\chi$ is given as

$$\text{sign}(\Gamma_5 D_W)\chi = \sum_{i=1}^{k} \text{sign}(\lambda_i)(\xi_i^H \chi)\xi_i + \text{sign}(\Gamma_5 D_W)\Pi\chi.$$

The first term on the right side can be computed explicitly and the second term is now easier to approximate with the sign function iteration, since the k eigenvalues closest to the singularity of $\text{sign}(\cdot)$ are effectively eliminated via Π.

	parameter	notation	default
(F)GMRES^{dp}	required reduction of initial residual	ε_{outer}	10^{-8}
	relaxation strategy	$\varepsilon_{\text{sign}}$	$\frac{\varepsilon_{outer}}{\|r_k\|} \cdot 10^{-2}$
	restart length for (F)GMRES	$m_{restart}$	100
DD-αAMG^{sp}	required reduction of initial residual	ε_{prec}	10^{-1}
	number of levels		2

Table 9.2: Parameters for the overlap solver. Here, *dp* denotes double precision and *sp* single precision.

Table 9.2 summarizes the default settings used for the results reported in Figure 9.6. The superscripts *dp* and *sp* indicate that we perform the preconditioning in single precision arithmetic, while the multiplication with D_N within the (F)GMRES iteration is done in double precision arithmetic, similarly to what has been done in the Chapters 7 and 8.

For the results reported in Figure 9.6 we tried to keep the cost for a matrix-vector multiplication with D_N independent of the number of smoothing steps which were applied to the configuration. To do so, we used the 100th smallest

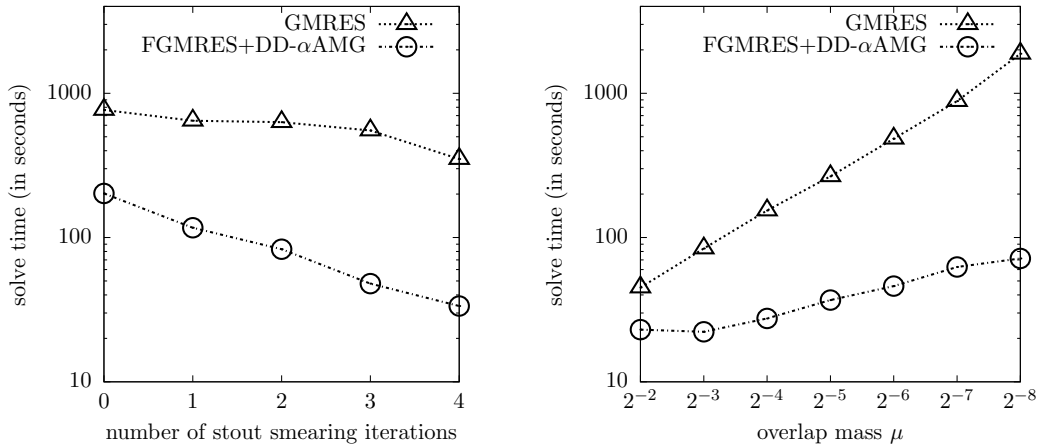

Figure 9.6: Comparison of execution times for preconditioned FGMRES and GMRES. Left: for 0 to 4 steps of stout smearing (configuration ID 9, default value for μ from Table 9.1), right: different overlap masses μ for configuration ID 9 and 3-step stout smearing.

eigenvalue of $\Gamma_5 D_W$ for $s = 0$ as a threshold, and deflated all eigenpairs with eigenvalues below this threshold for the configurations with $s > 0$. To be specific, for $s = 1, 2$ and 3 we deflated the smallest 17, 5 and 1 eigenpairs, respectively. For $s = 4$ none of the eigenvalues were below the threshold. In [41] it was already observed that smearing lowers the condition number of the kernel operator and thus cheapens the matrix-vector multiplication with D_N.

The left plot in Figure 9.6 shows that, at fixed default overlap mass μ, we gain a factor of 4 to 10 in execution time using the preconditioner. The quality of the preconditioning improves with the number of smearing steps. The right part of Figure 9.6 shows that for smaller values of μ we can expect an even larger reduction of the execution time. For the smallest value considered, $\mu = 2^{-8}$, which is realistic for future lattice simulations, the improvement due to preconditioning is a factor of about 25.

9.3.3 Comparison of Optimized Solvers

Physics production codes for simulations with the overlap operator use recursive preconditioning as an additional technique to further reduce the cost for the matrix-vector multiplication (MVM) with D_N; cf. [30]. This means that the FGMRES iteration is preconditioned by using an additional "inner" iteration to approximately invert D_N, this inner iteration being itself again FGMRES. The point is that we may require only low accuracy for this inner iteration, implying that all MVMs with $\mathrm{sign}(\Gamma_5 D_W)$ in the inner iteration may be approximated to low accuracy and computed in single precision, only.

In this framework, we can apply the DD-αAMG preconditioner, too, but this

	parameter	notation	default
inner FGMRES^{sp}	required reduction of initial residual (with preconditioning)	$\varepsilon_{inner}^{prec}$	10^{-2}
	required reduction of initial residual (without preconditioning)	ε_{inner}	10^{-1}
	relaxation strategy		$\frac{\varepsilon_{inner}, \varepsilon_{inner}^{prec}}{\|r_k\|} \cdot 10^{-2}$
	restart length	$m_{restart}^{inner}$	100

Table 9.3: Parameters for the inner iteration.

time as a preconditioner for the inner FGMRES iteration. In this manner we keep the advantage of needing only a low accuracy approximation to the MVM with $\text{sign}(\Gamma_5 D_W)$, while at the same time reducing the number of inner iterations and thus the (low accuracy) evaluations of MVMs with $\text{sign}(\Gamma_5 D_W)$.

We denote ε_{inner} the residual reduction we ask for in the unpreconditioned inner iteration and $\varepsilon_{inner}^{prec}$ the corresponding accuracy required when using the DD-αAMG iteration as a preconditioner. The inner iteration converges much faster when we use preconditioning. More accurate solutions in the inner iteration reduce the number of outer iterations and thus the number of costly high precision MVMs with $\text{sign}(\Gamma_5 D_W)$. When preconditioning is used for the inner iteration, requiring a higher accuracy in the inner iteration comes at relatively low additional cost. It is therefore advantageous to choose $\varepsilon_{inner}^{prec}$ smaller than ε_{inner}. As an addition to Table 9.2, Table 9.3 lists the default values we used for the inner iteration and which were found to be fairly optimal via numerical testing.

Figure 9.7 shows results for the solvers optimized in this way. We consider different sizes for the deflation subspace, i.e., the number of smallest eigen-

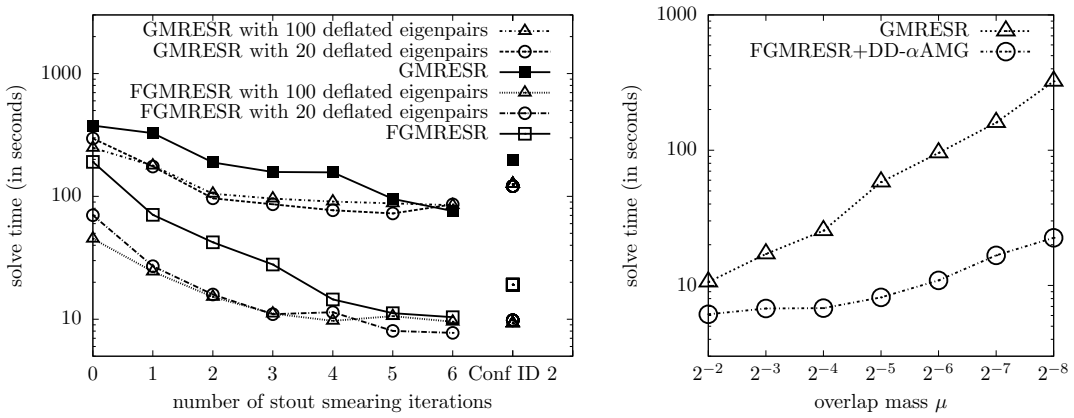

Figure 9.7: Comparison of GMRESR with FGMRESR with different deflation spaces (configuration IDs 9 and 10 with 1,024 processes).

values which we deflate explicitly. The computation of these eigenvalues (via PARPACK [106]) is costly, so that deflating a larger number of eigenvalues is efficient only if several system solves with the same overlap operator are to be performed. The figure shows that, irrespectively from the number of deflated eigenvalues, the preconditioned recursive method outperforms the unpreconditioned method in a similar way it did in the non-recursive case considered before. When more smearing steps are applied, the improvement grows; improvement factors reach 10 or more. The figure also shows that in the case that we have to solve only one or two linear systems with the same matrix, it is not advisable to use deflation at all, the cost for the computation of the eigenvalues being too large. We attribute this finding at least partly to the fact that the thick restart method used to approximate the sign function from [46] is particularly efficient, here. While all other data in Figure 9.7 was obtained for configuration ID 9, the rightmost data on the left plot refers to configuration ID 10. We see a similar high efficiency of our preconditioner as we did for configuration ID 9 with 3 smearing steps, an observation consistent with the fact that configuration ID 10 was also obtained using 3 steps of (HEX) smearing, see Table 9.1.

Conclusion & Outlook

The developed DD-αAMG method, combining domain decomposition techniques and algebraic multigrid, shows great potential to speed-up calculations in lattice QCD. For small quark masses and large lattice sizes our method outperforms conventional Krylov subspace methods like odd-even preconditioned mixed precision BiCGStab. It also outperforms the inexact deflation method for many right hand sides as well as for a single right hand side. This result is mainly due to the introduction of the highly parallel domain decomposition smoother, which improves upon the point smoother (GCR) used in the AMG method (Chapter 6) and the efficient setup procedure in our algebraic multigrid method. Based on the results for two-level DD-αAMG published in a preprint of [49], the inexact deflation method has been upgraded in the spirit of DD-αAMG.

Additional improvements can be achieved by a recursive extension of the two-grid method to a true multigrid method. We were able to reduce the time spent in the coarse grid solves. While it made up approximately 75% of the total time in the two-grid method, it is reduced to about 31% by using additional levels, yielding a speed-up factor of up to 2.5, particularly when using a moderate number of processors. Another factor of 1.5 can be obtained by machine specific `SSE` optimization; for longer registers and other architectures, the gain might be even bigger.

Furthermore, the fast DD-αAMG solvers for the Wilson-Dirac operator D_W now allow to efficiently use this operator as a preconditioner for the overlap operator. We presented a thorough analysis of this auxiliary space preconditioner in the case that D_W is normal. This is not the case in practice, but the trend in current simulations in lattice QCD is to reduce the non-normality of D_W as one approaches the continuum limit and smearing techniques are applied. For a state-of-the-art parallel implementation and for physically relevant configurations and parameters we showed that the improvements in time to solution gained through the preconditioning are at least a factor of 4 and, typically, more than 10.

Besides from working on the integration of our algorithm into the production

codes of our collaborators within the SFB/TRR55 "Hadron Physics from Lattice QCD", we plan to incorporate AVX optimization for recent machines. Our collaborators from Regensburg already released an optimized DD-αAMG solver for Intel KNC technology in [63], that is supplied with an optimized version of the implementation of our coarse grid correction. We plan to publish the code and to analyze the behavior of γ_5-preserving adaptive algebraic multigrid in the HMC. Additionally, we are working on a multigrid based eigensolver for the hermitian indefinite Wilson-Dirac operator $\Gamma_5 D_W$ and its applications. Due to the Γ_5 preservation, our coarse grid correction can be used for $\Gamma_5 D_W$ as well. Preliminary results from a Rayleigh quotient iteration with multigrid have already been published in [7].

List of Algorithms

List of Figures

List of Tables

Bibliography

[1] M. Albanese, F. Costantini, G. Fiorentini, F. Flore, M. P. Lombardo, P. Bacilieri R. Tripiccione, L. Fonti, E. Remiddi, M. Bernaschi, N. Cabibbo, L. A. Fernandez, E. Marinari, G. Parisi, G. Salina, S. Cabasino, F. Marzano, P. Paolucci, S. Petrarca, F. Rapuano, P. Marchesini, P. Giacomelli, and R. Rusack. Glueball masses and string tension in lattice QCD. *Phys. Lett.*, B192:163–169, 1987.

[2] C. Alexandrou, M. Brinet, J. Carbonell, M. Constantinou, P. A. Harraud, P. Guichon, K. Jansen, T. Korzec, and M. Papinutto. Nucleon electromagnetic form factors in twisted mass lattice QCD. *Phys. Rev.*, D83:094502, 2011.

[3] S. Aoki, K.-I. Ishikawa, N. Ishizuka, T. Izubuchi, D. Kadoh, K. Kanaya, Y. Kuramashi, Y. Namekawa, M. Okawa, Y. Taniguchi, A. Ukawa, N. Ukita, and T. Yoshie. 2+1 flavor lattice QCD toward the physical point. *Phys. Rev.*, D79:034503, 2009.

[4] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, S. D. Cohen, J. C. Osborn, and C. Rebbi. The role of multigrid algorithms for LQCD. *PoS*, LATTICE2009:031, 2009.

[5] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, T. A. Manteuffel, S. F. McCormick, J. C. Osborn, and C. Rebbi. Adaptive multigrid algorithm for the lattice Wilson-Dirac operator. *Phys. Rev. Lett.*, 105:201602, 2010.

[6] T. Bae, Y.-C. Jang, C. Jung, H.-J. Kim, J. Kim, K. Kim, W. Lee, S. R. Sharpe, and B. Yoon. Kaon B-parameter from improved staggered fermions in $n_f = 2 + 1$ QCD. *Phys. Rev. Lett.*, 109:041601, 2012.

[7] G. Bali, S. Collins, A. Frommer, K. Kahl, I. Kanamori, B. Müller, M. Rottmann, and J. Simeth. (Approximate) low-mode averaging with a new multigrid eigensolver. *PoS*, LATTICE2015:350, 2015.

[8] G. S. Bali, P. C. Bruns, S. Collins, M. Deka, B. Gläßle, M. Göckeler, L. Greil, T. R. Hemmert, R. Horsley, J. Najjar, Y. Nakamura, A. Nobile, D. Pleiter, P. E. L. Rakow, A. Schäfer, R. Schiel, G. Schierholz, A. Sternbeck, and J. Zanotti. Nucleon mass and sigma term from lattice QCD with two light fermion flavors. *Nucl. Phys.*, B866:1–25, 2013.

[9] T. Banks and A. Casher. Chiral symmetry breaking in confining theories. *Nucl. Phys.*, B169:103, 1980.

[10] G. Baym and D. K. Campbell. Chiral symmentry and pion condensates. In *Mesons in Nuclei*, 1978.

[11] R. Ben-Av, M. Harmatz, S. Solomon, and P. G. Lauwers. Fermion simulations using parallel transported multigrid. *Phys. Lett.*, B253:185–192, 1991.

[12] T. Bergrath, M. Ramalho, R. Kenway, et al. PRACE scientific annual report 2012. Technical report, PRACE, 2012. `http://www.prace-ri.eu/IMG/pdf/PRACE_Scientific_Annual_Report_2012.pdf`, p. 32.

[13] F. Berruto, R. Narayanan, and H. Neuberger. Exact local fermionic zero modes. *Phys. Lett.*, B489:243–250, 2000.

[14] A. Bjorck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, USA, 1996.

[15] U. Block, A. Frommer, and G. Mayer. Block colouring schemes for the SOR method on local memory parallel computers. *Parallel Computing*, 14(1):61–75, 1990.

[16] C. Bonati and M. D'Elia. A comparison of the gradient flow with cooling in SU(3) pure gauge theory. *Phys. Rev.*, D89:105005, 2014.

[17] S. Borsanyi, Y. Delgado, S. Durr, Z. Fodor, S. D. Katz, S. Krieg, T. Lippert, D. Nogradi, and K. K. Szabo. QCD thermodynamics with dynamical overlap fermions. *Phys. Lett.*, B713:342–346, 2012.

[18] S. Borsanyi, S. Dürr, Z. Fodor, J. Frison, C. Hoelbling, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, A. Portelli, A. Ramos, A. Sastre, and K. K. Szabo. Isospin splittings in the light baryon octet from lattice QCD and QED. *Phys. Rev. Lett.*, 111:252001, 2013.

[19] S. Borsanyi, Z. Fodor, S. D. Katz, S. Krieg, T. Lippert, D. Nogradi, F. Pittler, K. K. Szabo, and B. C. Toth. QCD thermodynamics with continuum extrapolated dynamical overlap fermions. *arXiv:1510.03376 [hep-lat]*, 2015.

[20] D. Braess. Towards algebraic multigrid for elliptic problems of second order. *Computing*, 55(4):379–393, 1995.

[21] A. Brandt. Multiscale scientific computation: Review 2001. In *Multiscale and Multiresolution Methods*, volume 20 of *Lecture Notes in Computational Science and Engineering*, pages 3–95. Springer Berlin Heidelberg, 2002.

[22] A. Brandt, J. Brannick, K. Kahl, and I. Livshits. Bootstrap AMG. *SIAM J. Sci. Comput.*, 33(2):612–632, 2011.

[23] J. Brannick, R. C. Brower, M. A. Clark, J. C. Osborn, and C. Rebbi. Adaptive multigrid algorithm for lattice QCD. *Phys. Rev. Lett.*, 100:041601, 2007.

[24] J. Brannick, A. Frommer, K. Kahl, B. Leder, M. Rottmann, and A. Strebel. Multigrid preconditioning for the overlap operator in lattice QCD. *Numer. Math.*, 2015.

[25] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Review*, 47:317–346, 2005.

[26] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders. Towards adaptive smoothed aggregation (αSA) for nonsymmetric systems. *SIAM J. Sci. Comput.*, 32:14–39, 2010.

[27] R. Brower, E. Myers, C. Rebbi, and K. Moriarty. The multigrid method for fermion calculations in quantum chromodynamics. Technical Report Print-87-0335, IAS, Princeton, 1987.

[28] S. Capitani, S. Durr, and C. Hoelbling. Rationale for UV-filtered clover fermions. *JHEP*, 0611(2006)028, 2006.

[29] CLS. Coordinated lattice simulation. `https://twiki.cern.ch/twiki/bin/view/CLS/`.

[30] N. Cundy, J. van den Eshof, A. Frommer, S. Krieg, and K. Schäfer. Numerical methods for the QCD overlap operator. III: Nested iterations. *Comput. Phys. Commun.*, 165:221–242, 2005.

[31] P. de Forcrand, A. Kurkela, and M. Panero. Numerical properties of staggered overlap fermions. *PoS*, LATTICE2010:080, 2010.

[32] T. DeGrand and C. E. Detar. *Lattice Methods for Quantum Chromodynamics*. World Scientific, 2006.

[33] T. A. DeGrand and P. Rossi. Conditioning techniques for dynamical fermions. *Comput. Phys. Commun.*, 60(2):211–214, 1990.

[34] L. Del Debbio, L. Giusti, M. Lüscher, R. Petronzio, and N. Tantalo. QCD with light Wilson quarks on fine lattices (i): First experiences and physics results. *JHEP*, 02(2007)056, 2007.

[35] L. Del Debbio, L. Giusti, M. Lüscher, R. Petronzio, and N. Tantalo. QCD with light Wilson quarks on fine lattices (ii): DD-HMC simulations and data analysis. *JHEP*, 0702(2007)082, 2007.

[36] J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA, USA, 2000.

[37] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers*. SIAM, Philadelphia, PA, USA, 1990.

[38] S. Dürr, Z. Fodor, J. Frison, C. Hoelbling, R. Hoffmann, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, K. K. Szabo, and G. Vulvert. Ab initio determination of light hadron masses. *Science*, 322(5905):1224–1227, 2008.

[39] S. Dürr, Z. Fodor, C. Hoelbling, S. D. Katz, S. Krieg, et al. Lattice QCD at the physical point: Simulation and analysis details. *JHEP*, 08(2011)148, 2011.

[40] S. Dürr, Z. Fodor, C. Hoelbling, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, K. K. Szabo, and G. Vulvert. Lattice QCD at the physical point: Light quark masses. *Phys. Lett. B701*, pages 265–268, 2011.

[41] S. Durr, C. Hoelbling, and U. Wenger. UV-filtered overlap fermions. *PoS*, LATTICE2005:144, 2006.

[42] R. G. Edwards, U. M. Heller, and R. Narayanan. A study of practical implementations of the overlap Dirac operator in four-dimensions. *Nucl. Phys.*, B540:457–471, 1999.

[43] J. Finkenrath, F. Knechtli, and B. Leder. One flavor mass reweighting in lattice QCD. *Nucl. Phys.*, B877:441–456, 2013.

[44] P. Fritzsch, F. Knechtli, B. Leder, M. Marinkovic, S. Schaefer, R. Sommer, and F. Virotta. The strange quark mass and Lambda parameter of two flavor QCD. *Nucl. Phys.*, B865:397–429, 2012.

[45] A. Frommer, S. Güttel, and M. Schweitzer. Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices. *SIAM J. Matrix Anal. Appl.*, 35:1602–1624, 2014.

[46] A. Frommer, S. Güttel, and M. Schweitzer. Efficient and stable Arnoldi restarts for matrix functions based on quadrature. *SIAM J. Matrix Anal. Appl.*, 35:661–683, 2014.

[47] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann. Aggregation-based multilevel methods for lattice QCD. In *Proceedings of Science,* `http://pos.sissa.it`, volume LATTICE2011:046, 2011.

[48] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann. An adaptive aggregation based domain decomposition multilevel method for the lattice Wilson Dirac operator: Multilevel results. *arXiv:1307.6101*, 2013.

[49] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann. Adaptive aggregation based domain decomposition multigrid for the lattice Wilson Dirac operator. *SIAM J. Sci. Comp.*, 36(4):A1581–A1608, 2014.

[50] A. Frommer, A. Nobile, and P. Zingler. Deflation and flexible SAP-preconditioning of GMRES in lattice QCD simulation. *arXiv:1204.5463 [hep-lat]*, 2012.

[51] C. Gattringer and C. B. Lang. *Quantum Chromodynamics on the Lattice*, volume 788 of *Lect. Notes Phys.* Springer, 2009.

[52] P. Ghysels, T. J. Ashby, K. Meerbergen, and W. Vanroose. Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM J. Sci. Comput.*, 35(1):C48–C71, 2013.

[53] P. H. Ginsparg and K. G. Wilson. A remnant of chiral symmetry on the lattice. *Phys. Rev. D*, 25:2649–2657, May 1982.

[54] L. Giusti, C. Hoelbling, M. Lüscher, and H. Wittig. Numerical techniques for lattice QCD in the epsilon regime. *Comput. Phys. Commun.*, 153:31–51, 2003.

[55] I. Gohberg, P. Lancaster, and L. Rodman. *Indefinite Linear Algebra and Applications.* Birkhäuser, Basel, 2005.

[56] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 2nd edition, 1989.

[57] F. Gray. Pulse code communication, 1953. US Patent 2,632,058.

[58] M. Guest, G. Aloisio, R. Kenway, et al. The scientific case for HPC in Europe 2012 - 2020. Technical report, PRACE, October 2012. http://www.prace-ri.eu/PRACE-The-Scientific-Case-for-HPC, p. 75.

[59] W. Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer, 1985.

[60] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2010.

[61] A. Hasenfratz, R. Hoffmann, and S. Schaefer. Hypercubic smeared links for dynamical fermions. *JHEP*, 0705(2007)029, 2007.

[62] A. Hasenfratz, R. Hoffmann, and S. Schaefer. Localized eigenmodes of the overlap operator and their impact on the eigenvalue distribution. *JHEP*, 0711(2007)071, 2007.

[63] S. Heybrock, D. Richtmann, and T. Wettig. QCD solver library for Intel MIC architecture. https://rqcd.ur.de:8443/hes10653/mic-qcd-solver.

[64] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, USA, 2nd edition, 2002.

[65] Intel. Intel xeon x5570 (nehalem-ep) quad-core processors. http://ark.intel.com/products/37111/Intel-Xeon-Processor-X5570-8M-Cache-2_93-GHz-6_40-GTs-Intel-QPI.

[66] Jülich Supercomputing Centre. Juropa - Jülich research on petaflop architectures. http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUROPA/JUROPA_node.html.

[67] K. Kahl and H. Rittich. Analysis of the deflated conjugate gradient method based on symmetric multigrid theory. Preprint BUW-IMACM 12/19, 2012.

[68] T. Kalkreuter. Multigrid methods for propagators in lattice gauge theories. *J. Comput. Appl. Math.*, 63:57–68, 1995.

[69] D. B. Kaplan. A method for simulating chiral fermions on the lattice. *Phys. Lett. B*, 288(3–4):342–347, 1992.

[70] A. D. Kennedy. Algorithms for dynamical fermions. *arXiv:hep-lat/0607038*, 2006.

[71] S. Krieg and T. Lippert. Tuning lattice QCD to petascale on Blue Gene/P. *NIC Symposium 2010*, pages 155–164, 2010.

[72] S. Krieg and K. Szabo. Personal Communication, 2015.

[73] H. B. Lawson and M. L. Michelsohn. *Spin Geometry*. Princeton Mathematical Series. Princeton University Press, 1989.

[74] T. Lippert. Parallel SSOR preconditioning for lattice QCD. *Parallel Computing*, 25(10–11):1357–1370, 1999.

[75] M. Lüscher. DD-HMC algorithm for two-flavour lattice QCD. `http://luscher.web.cern.ch/luscher/DD-HMC`, used version: DD-HMC-1.2.2, September 2008.

[76] M. Lüscher. openQCD simulation program for lattice QCD with open boundary conditions. `http://luscher.web.cern.ch/luscher/openQCD/`, used version: openQCD-1.2, May 2013.

[77] M. Lüscher. Exact chiral symmetry on the lattice and the Ginsparg-Wilson relation. *Phys. Lett.*, B428:342–345, 1998.

[78] M. Lüscher. Solution of the Dirac equation in lattice QCD using a domain decomposition method. *Comput. Phys. Commun. 156*, pages 209–220, 2004.

[79] M. Lüscher. Local coherence and deflation of the low quark modes in lattice QCD. *JHEP*, 07(2007)081, 2007.

[80] M. Lüscher. Properties and uses of the Wilson flow in lattice QCD. *JHEP*, 1008(2010)071, 2010.

[81] M. Lüscher. Trivializing maps, the Wilson flow and the HMC algorithm. *Commun. Math. Phys.*, 293:899–919, 2010.

[82] I. Montvay and G. Münster. *Quantum Fields on a Lattice*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1994.

[83] C. Morningstar and M. J. Peardon. Analytic smearing of SU(3) link variables in lattice QCD. *Phys. Rev.*, D69:054501, 2004.

[84] J. W. Negele. Instantons, the QCD vacuum, and hadronic physics. *Nucl. Phys. Proc. Suppl.*, 73:92–104, 1999.

[85] S. V. Nepomnyaschikh. Mesh theorems on traces, normalizations of function traces and their inversion. *Soviet J. Numer. Anal. Math. Modelling*, 6:223–242, 1991.

[86] H. Neuberger. Exactly massless quarks on the lattice. *Phys. Lett. B*, 417(1–2):141–144, 1998.

[87] H. Neuberger. Bounds on the Wilson Dirac operator. *Phys. Rev.*, D61:085015, 2000.

[88] F. Niedermayer. Exact chiral symmetry, topological charge and related topics. *Nucl. Phys. Proc. Suppl.*, 73:105–119, 1999.

[89] H. B. Nielsen and M. Ninomiya. No go theorem for regularizing chiral fermions. *Phys. Lett.*, B105:219–223, 1981.

[90] Y. Notay and P. S. Vassilevski. Recursive Krylov-based multigrid cycles. *Numerical Linear Algebra with Applications*, 15:473–487, 2008.

[91] J. C. Osborn. Multigrid solver for clover fermions, implementation within QOPQDP. `http://usqcd.jlab.org/usqcd-docs/qopqdp/`, used version: QOPQDP 0.19.0, April 2013.

[92] J. C. Osborn, R. Babich, J. Brannick, R. C. Brower, M. A. Clark, S. D. Cohen, and C. Rebbi. Multigrid solver for clover fermions. *PoS*, LATTICE2010:037, 2010.

[93] P. Oswald. Preconditioners for nonconforming discretizations. *Math. Comp.*, 65:923–941, 1996.

[94] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice Hall, Englewood Cliffs, NJ, 1980.

[95] M. E. Peskin and D. V. Schroeder. *An Introduction To Quantum Field Theory (Frontiers in Physics)*. Advanced Book Program. Westview Press, Boulder Colorado, 1995.

[96] L. Reichel. The application of Leja points to Richardson iteration and polynomial preconditioning. *Linear Algebra Appl.*, 154:389–414, 1991.

[97] H. Rittich. Deflation in multigrid methods. Master's thesis, Bergische Universtität Wuppertal, 2011.

[98] J. Ruge and K. Stüben. Algebraic multigrid. In Stephen F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, USA, 1987.

[99] Y. Saad. Analysis of augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.*, 18(2):435–449, 1997.

[100] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, USA, 2nd edition, 2003.

[101] H. Schwarz. Gesammelte mathematische Abhandlungen. *Vierteljahrschrift Naturforsch. Ges. Zürich*, pages 272–286, 1870.

[102] B. Sheikholeslami and R. Wohlert. Improved continuum limit lattice action for QCD with Wilson fermions. *Nucl. Phys.*, B259:572–597, 1985.

[103] V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25:454–477, 2003.

[104] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, New York, 1996.

[105] G. D. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford Applied Mathematics and Computing Science Series. Clarendon Press, 1985.

[106] D. C. Sorensen, R. B. Lehoucq, C. Yang, and K. Maschhoff. PARPACK. `http://http://www.caam.rice.edu/software/ARPACK`, used version: 2.1, September 1996.

[107] L. Susskind. Lattice fermions. *Phys. Rev.*, D16:3031–3039, 1977.

[108] J. M. Tang, S. P. MacLachlan, R. Nabben, and C. Vuik. A comparison of two-level preconditioners based on multigrid and deflation. *SIAM J. Matrix Anal. Appl.*, 31(4):1715–1739, March 2010.

[109] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.

[110] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid. With Guest Contributions by A. Brandt, P. Oswald, K. Stüben*. Academic Press, Orlando, FL, 2001.

[111] J. van den Eshof, A. Frommer, T. Lippert, K. Schilling, and H. A. van der Vorst. Numerical methods for the QCD overlap operator. I: Sign-function and error bounds. *Comput. Phys. Commun.*, 146:203–224, 2002.

[112] J. van den Eshof and G. L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 26:125–153, 2004.

[113] J. C. Vink. Multigrid inversion of staggered and Wilson fermion operators with $SU(2)$ gauge fields in two-dimensions. *Phys. Lett.*, B272:81–85, 1991.

[114] H. S. Wilf and A. Nijenhuis. *Combinatorial Algorithms: An Update*. CBMS-NSF regional conference series in applied mathematics. SIAM, Philadelphia, PA, USA, 1989.

[115] K. G. Wilson. Quarks and strings on a lattice. In A. Zichichi, editor, *New Phenomena in Subnuclear Physics. Part A. Proceedings of the First Half of the 1975 International School of Subnuclear Physics, Erice, Sicily, July 11 - August 1, 1975*, volume 321 of *CLNS Reports*, pages 69–138, New York, 1977. Plenum Press.

[116] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56:215–235, 1996.