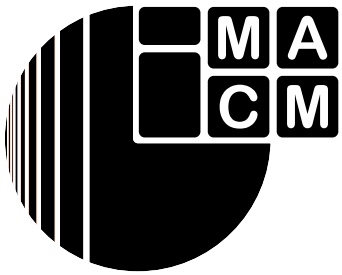


Bergische Universität Wuppertal

Fachbereich Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational Mathematics (IMACM)

Preprint BUW-IMACM 13/18

A. Frommer, K. Kahl, S. Krieg, B. Leder and M. Rottmann

Adaptive Aggregation Based Domain Decomposition Multigrid for the Lattice Wilson Dirac Operator

May 2013

http://www.math.uni-wuppertal.de

ADAPTIVE AGGREGATION BASED DOMAIN DECOMPOSITION MULTIGRID FOR THE LATTICE WILSON DIRAC OPERATOR *

A. FROMMER†, K. KAHL†, S. KRIEG‡, B. LEDER†, AND M. ROTTMANN†

Abstract. In lattice QCD computations a substantial amount of work is spent in solving discretized versions of the Dirac equation. Conventional Krylov solvers show critical slowing down for large system sizes and physically interesting parameter regions. We present a domain decomposition adaptive algebraic multigrid method used as a preconditioner to solve the "clover improved" Wilson discretization of the Dirac equation. This approach combines and improves two approaches, namely domain decomposition and adaptive algebraic multigrid, that have been used separately in lattice QCD before. We show in extensive numerical test conducted with a parallel production code implementation that considerable speed-up over conventional Krylov subspace methods, domain decomposition methods and other hierarchical approaches for realistic system sizes can be achieved.

Key words. multilevel, multigrid, lattice QCD, Wilson Dirac operator, domain decomposition, aggregation, adaptivity, parallel computing.

AMS subject classifications. 65F08, 65F10, 65Z05, 65Y05

1. Introduction.

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 [8, 26]. Our concern in this paper is three-fold: We want to make the mathematical modeling related with QCD and lattice QCD more popular in the scientific computing community and therefore spend some effort on explaining fundamentals. On top of that we develop a new and efficient adaptive algebraic multigrid method to solve systems with the discretized Dirac operator, and we show results for a large number of numerical experiments based on an advanced, production code quality implementation with up-to-date physical data.

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

$$Dz = b, \tag{1.1}$$

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 constant 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 [1, 4, 5, 20, 22]. Usually these linear systems are solved by standard Krylov subspace methods. Those suffer from critical slowing down when approaching the physically relevant parameter values (e.g., physical mass constant or lattice spacing $a \to 0$). 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 [18, 32], deflation techniques [35] and domain decomposition approaches [23, 34]. While these approaches yield significant speed-ups over the unpreconditioned versions, their scaling behavior is unchanged and critical slowing down still occurs. Comparing with

*This work was partially funded by Deutsche Forschungsgemeinschaft (DFG) Transregional Collaborative Research Centre 55 (SFB/TRR55)

†Department of Mathematics, Bergische Universität Wuppertal, 42097 Germany, {frommer,kkahl,leder,rottmann}@math.uni-wuppertal.de.

‡Department of Physics, Bergische Universität Wuppertal, 42097 Germany and Jülich Supercomputing Centre, Forschungszentrum Jülich, 52428 Jülich, Germany, s.krieg@fz-juelich.de.

established theoretical results for domain decomposition approaches for elliptic PDEs this behavior is to be expected since there the condition number of the preconditioned system scales with H/h, where h is the lattice spacing and H the block size [44].

Motivated by the potential (e.g., for elliptic PDEs) of convergence independence of the discretization mesh-size, multigrid methods have been considered in the lattice QCD community as well. 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 [7, 15, 29, 49]. With the advent of *adaptive algebraic* multigrid methods effective preconditioners for QCD calculations could be constructed in recent years. The pioneering work from [3, 12, 37] showed very promising results. There, an adaptive non-smoothed aggregation approach based on [13] has been proposed for the solution of the Wilson Dirac system of equations.

Within the physics community, another hierarchical technique, the recently proposed domain decomposition type solver named *inexact deflation* developed in [35] is widely used. A well-optimized code for this solver is publicly available [33]. 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 [13], 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 paper we present a multigrid method that combines aspects from [35], namely a domain decomposition smoother, and from non-smoothed aggregation as in [3, 37]. Our approach elaborates on the multigrid methods from [3, 37] 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 [3, 37] and [35] 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 [35] into a true multigrid method. The little Dirac system now needs to be solved only with low accuracy. This allows, in particular to apply the method recursively and thus allows for a true and more efficient multigrid instead of just a two-grid method, a feature which is impossible to do with the approach from [35].

The paper is organized as follows. In Section 2 we give an introduction into lattice QCD for the non-specialist and we introduce the domain decomposition Schwarz method in this context. In Section 3 we first outline algebraic multigrid methods in general and then focus on aggregation based approaches. Thereby we address the peculiarities of lattice QCD and explain different possible adaptive strategies for the construction of the multigrid hierarchy. The inexact deflation method from [35] is discussed in Section 4, where in particular we point out the differences to a multigrid method and describe the adaptive nature of its setup. In Section 5 we finally formulate our multigrid approach, for which we present thorough numerical tests in Section 6.

2. Lattice Quantum Chromodynamics. Quantum Chromodynamics (QCD) or the theory of strong interactions, is a quantum field theory in four-dimensional space-time and part of the standard model of elementary particle physics. It 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. [19]). 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. After a brief description of continuum QCD we introduce its discretization on a hyper-cubic lattice and discuss the need of iterative methods, e.g., Krylov subspace methods, for the solution of the arising linear systems of equations. Due to the ill-conditioned nature of these systems, preconditioning is advised and the use of a domain decomposition method is discussed as a prerequisite for our multigrid construction.

2.1. Continuum QCD. The degrees of freedom of QCD are matter fields, called quarks, and gauge fields, called gluons. The governing system of partial differential equations of QCD are the Dirac equations

$$(\mathcal{D} + m)\psi = \eta \tag{2.1}$$

which define the dynamics of the quarks and the interaction of quarks and gluons for a given gluon field background. Here, $\psi = \psi(x)$ and $\eta = \eta(x)$ represent matter fields. These depend on x, the points in space-time, $x = (x_0, x_1, x_2, x_3)$[1]. The gluons are represented in the Dirac operator $\mathcal{D}$ to be discussed below, and m represents a scalar mass parameter not depending on x. This mass parameter sets the mass of the quarks in the QCD theory.

In the continuum theory the Dirac operator $\mathcal{D}$ 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. 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 quark fields ψ and η in (2.1) carry two indices that are suppressed, i.e., $\psi = \psi_{c\sigma}$. These indices label internal degrees of freedom of the quarks; one is called color ($c = 1, 2, 3$) and the other spin ($\sigma = 0, 1, 2, 3$). At each point x in space-time, we can represent the spinor $\psi(x)$, i.e. the quark field ψ at a given point x, as a twelve component column vector

$$\psi(x) = (\psi_{10}(x), \psi_{20}(x), \psi_{30}(x), \psi_{11}(x), \ldots, \psi_{33}(x))^T \ . \tag{2.2}$$

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 is in the matrix representation of $\mathfrak{su}(3)$ and acts non-trivially on the color and trivially on the spin degrees of freedom, i.e, $(A_\mu\psi)(x) = (I_4 \otimes A_\mu(x))\psi(x)$.

The γ-matrices $\gamma_0, \gamma_1, \gamma_2, \gamma_3 \in \mathbb{C}^{4\times4}$ act non-trivially on the spin and trivially on the color degrees of freedom. They are hermitian and unitary matrices which generate

[1] Physical space-time is a four dimensional Minkowski space. We present the theory in Euclidean space-time since this version can be treated numerically. The two versions are equivalent, cf. [36].

the Clifford algebra $Cl_4(\mathbb{C})$, i.e., they satisfy

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = 2\delta_{\mu\nu}I_4 \quad \text{for} \quad \mu,\nu = 0,1,2,3. \tag{2.3}$$

Unlike the gauge fields A_μ, the γ-matrices do not depend on x. The multiplication of a γ-matrix with ψ is defined by $(\gamma_\mu\psi)(x) = (\gamma_\mu \otimes I_3)\psi(x)$.

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 (Lorentz-transformations) in the same way as $\psi(x)$. Local gauge invariance and special relativity are fundamental principles of the standard model of elementary particle physics.

2.2. Lattice QCD. 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. For a more detailed introduction to QCD and lattice QCD we refer the interested reader to [17, 24, 36].

Consider a four-dimensional Torus $\mathcal{T}$. On $\mathcal{T}$ we have a periodic $N_t \times N_s^3$ lattice $\mathcal{L} \subset \mathcal{T}$ with lattice spacing a and $n_\mathcal{L} = N_t \cdot N_s^3$ lattice points. In here N_s denotes the number of lattice points for each of the three space dimensions and N_t the number of lattice points in the time dimension. Hence, for any $x, y \in \mathcal{L}$ there exists a $p \in \mathbb{Z}^4$ such that

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

For shift operations on the lattice, we define shift vectors $\hat{\mu} \in \mathbb{R}^4$ by

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

A quark field ψ is defined at each point of the lattice, i.e., it is a function

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

on the lattice $\mathcal{L}$ which maps a point $x \in \mathcal{L}$ to a spinor $\psi(x)$. As in continuum QCD, this spinor again has color and spin indices $\psi_{c\sigma},\ c = 1,2,3,\ \sigma = 0,1,2,3$. For future use we introduce the symbols $\mathcal{C}$ and $\mathcal{S}$ for the color and the spin space, i.e.

$$\mathcal{C} = \{1,2,3\},\ \mathcal{S} = \{0,1,2,3\}.$$

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.

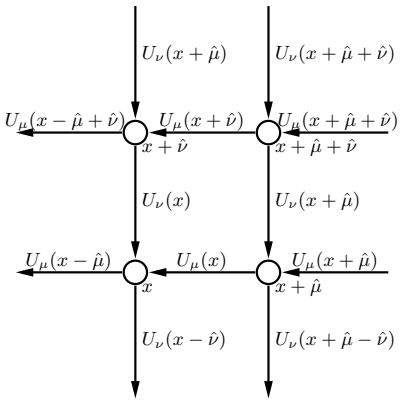


FIG. 2.1. *Naming conventions on the lattice.*

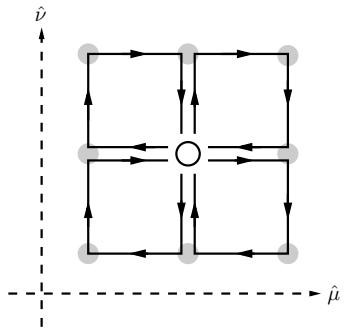


FIG. 2.2. *The clover term.*

To this purpose variables $U_\mu(x)$ are introduced. $U_\mu(x)$ connects x and $x+\hat\mu$, so that 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 $U_\mu(x)^{-1}$. The matrices $U_\mu(x)$ are an approximation to the path-ordered exponential of the integral of A_μ along the link. They 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 called a *gauge link* and the set of all gauge links $\{U_\mu(x) : x \in \mathcal{L},\ \mu = 0,1,2,3\}$ is called *configuration*.

The covariant derivative of the continuum theory can be discretized in many ways. Here we restrict ourselves to the widely used Wilson discretization (cf. [50]), noting that the multigrid solver developed in this paper is in principle applicable to any discretization resulting in local couplings.

We define forward covariant finite differences

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

and backward covariant finite differences

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

Since $(\Delta^\mu)^H = -\Delta_\mu$, the centralized covariant finite differences $(\Delta^\mu + \Delta_\mu)/2$ are anti-hermitian. The simplest discretization of the Dirac operator $\mathcal{D}$ is then given by

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

This so-called naive discretization suffers from unphysical eigenvectors to eigenvalues of small magnitude (also known as the "species doubling effect" or "red-black instability"). This is a standard phenomenon when discretizing first order derivatives using central finite differences, cf. [45]. Wilson introduced the stabilization term $a\Delta_\mu\Delta^\mu$, a centralized second order covariant finite difference, to avoid this problem. The Wilson

discretization of the Dirac operator is thus given by

$$D_W = \frac{m_0}{a} I + \frac{1}{2} \sum_{\mu=0}^{3} \Big(\gamma_\mu \otimes (\Delta_\mu + \Delta^\mu) - a I_4 \otimes \Delta_\mu \Delta^\mu \Big), \tag{2.4}$$

where the mass parameter m_0 is used to tune the mass of the quark to its physical value.

The commutativity relations (2.3) of the γ-matrices imply a non-trivial symmetry of D_W. Defining $\gamma_5 = \gamma_0\gamma_1\gamma_2\gamma_3$ we have $\gamma_5\gamma_\mu = -\gamma_\mu\gamma_5$ for $\mu = 0, 1, 2, 3$, and since γ_μ and γ_5 are hermitian we see that $\gamma_5\gamma_\mu$ is anti-hermitian. Thus the operator $(\gamma_5\gamma_\mu) \otimes (\Delta_\mu + \Delta^\mu)$ is hermitian, being the product of two anti-hermitian operators. To describe the resulting Γ_5-*symmetry* of the Wilson Dirac operator, we define $\Gamma_5 = I_{n_\mathcal{L}} \otimes \gamma_5 \otimes I_3$ and have $(\Gamma_5 D_W)^H = \Gamma_5 D_W$. To reduce the order of the discretization error as a function of a, a so-called Sheikholeslami-Wohlert or "clover" term (cf. [42] and Figure 2.2) is added to the lattice Wilson Dirac operator

$$D = D_W - \frac{c_{sw}}{32a} \sum_{\mu,\nu=0}^{3} (\gamma_\mu\gamma_\nu) \otimes (Q_{\mu\nu} - Q_{\nu\mu}), \tag{2.5}$$

where $(Q_{\mu\nu}\psi_\sigma)(x) = Q_{\mu\nu}(x)\psi_\sigma(x)$ with

$$\begin{aligned} Q_{\mu\nu}(x) \quad = \quad & U_\mu(x)\, U_\nu(x+\hat{\mu})\, U_\mu(x+\hat{\nu})^H\, U_\nu(x)^H + \\ & U_\nu(x)\, U_\mu(x-\hat{\mu}+\hat{\nu})^H\, U_\nu(x-\hat{\mu})^H\, U_\mu(x-\hat{\mu}) + \\ & U_\mu(x-\hat{\mu})^H\, U_\nu(x-\hat{\mu}-\hat{\nu})^H\, U_\mu(x-\hat{\mu}-\hat{\nu})\, U_\nu(x-\hat{\nu}) + \\ & U_\nu(x-\hat{\nu})^H\, U_\mu(x-\hat{\nu})\, U_\nu(x-\hat{\nu}+\hat{\mu})\, U_\mu(x)^H. \end{aligned}$$

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}). The resulting discretized Dirac operator D thus has discretization effects of order $\mathcal{O}(a^2)$. It is again Γ_5-symmetric, i.e., we have

$$(\Gamma_5 D)^H = \Gamma_5 D. \tag{2.6}$$

The Γ_5-symmetry induces a symmetry on the spectrum of D. For future use, we state this in the following lemma.

LEMMA 2.1. *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.*

Proof. Due to $D^H = \Gamma_5 D \Gamma_5$ we have

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

□

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 (2.6), depends on a mass parameter m_0, the Sheikholeslami-Wohlert constant c_{sw}, and a configuration $\{U_\mu(x) : x \in \mathcal{L},\, \mu = 0, 1, 2, 3\}$. For physically relevant mass parameters, the spectrum of D is contained in the right half plane, cf. Fig. 2.3 and Fig. 2.4.

While the continuum Dirac operator is normal, the Wilson Dirac operator is not, but it approaches normality when discretization effects become smaller. For small

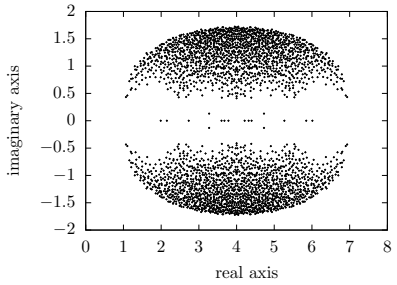


FIG. 2.3. *Spectrum of a 4^4 Wilson Dirac operator with $m_0 = 0$ and $c_{sw} = 0$.*

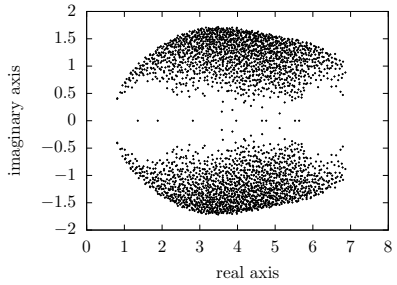


FIG. 2.4. *Spectrum of a 4^4 "Clover improved" Wilson Dirac operator with $m_0 = 0$ and $c_{sw} = 1$.*

lattice spacing, large lattice sizes and physically relevant mass parameters we can thus expect that the whole field of values $\mathcal{F}(D) = \{\psi^H D\psi : \psi^H\psi = 1\}$ of D is in the right half plane.

To explicitly formulate D in matrix terms we fix a default 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

$$\gamma_5 = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{pmatrix}.$$

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

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

2.3. Domain Decomposition in Lattice QCD. For ease of notation we from now on drop the lattice spacing a, so that the lattice $\mathcal{L}$ is given as

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

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

DEFINITION 2.2. *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,

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

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

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.,

$$\mathcal{V}_i = \mathcal{L}_i \times \mathcal{C} \times \mathcal{S}. \tag{2.7}$$

Since the systems arising in lattice QCD calculations tend to have hundreds of millions of unknowns they require the use of parallel computers. For this reason and due to the fact that, as a rule, naive domain decomposition is already used to parallelize the matrix vector product Dz which is needed for Krylov subspace methods, it is natural to also use a domain decomposition approach as a preconditioner.

The method of choice here is a colored version of the multiplicative Schwarz method [41, 44]. Since the discretized Dirac operator has only nearest-neighbor couplings, only two colors are needed. This red-black Schwarz method for the solution of $Dz = b$ is given in Algorithm 1 for a block decomposition of the lattice and variable blocks $\mathcal{V}_i$ according to (2.7). Figure 2.5 illustrates a 2D example. The corresponding trivial embeddings and block solvers are denoted by

$$\mathcal{I}_{\mathcal{V}_i} : \mathcal{V}_i \to \mathcal{V} \text{ and } B_i = \mathcal{I}_{\mathcal{V}_i}[\mathcal{I}_{\mathcal{V}_i}^T D \mathcal{I}_{\mathcal{V}_i}]^{-1}\mathcal{I}_{\mathcal{V}_i}^T.$$

In each iteration, for the updates in lines 6 and 10 we have to solve the local systems

$$D_i e_i = \mathcal{I}_{\mathcal{V}_i}^T r \quad \text{with } D_i = \mathcal{I}_{\mathcal{V}_i}^T D \mathcal{I}_{\mathcal{V}_i}, \tag{2.8}$$

D_i representing the restriction of D on the lattice-block $\mathcal{V}_i$.

Algorithm 1 Red-Black Schwarz

```
1: input: z, b, ν
2: output: z
3: for k = 1 to ν do
4:    r ← b − Dz
5:    for all red i do
6:       z ← z + B_i r
7:    end for
8:    r ← b − Dz
9:    for all black i do
10:      z ← z + B_i r
11:   end for
12: end for
```

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 1 as (cf. [44])

$$z \leftarrow (I - KD)z + Kb. \tag{2.9}$$

Since the solution $z^* = D^{-1}b$ satisfies $z^* = (I - KD)z^* + Kb$, one iteration of Algorithm 1 updates the error $e = z - z^*$ via

$$e \leftarrow (I - KD)e,$$

with $I - KD$ the error propagation operator,

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

The red-black Schwarz method has been introduced to lattice QCD in [34] and been used ever since in several lattice QCD implementations as a preconditioner (cf. [2, 23, 35]). In this context red-black Schwarz is also known as Schwarz Alternating Procedure (SAP). In what follows the application of ν iterations of SAP to a vector b with initial guess $z = 0$ is denoted by the linear operator

$$M_{SAP}^{(\nu)} b = K \sum_{k=0}^{\nu-1} (I - DK)^k\, b\,.$$

This representation follows by repeated application of (2.9). Note that $E_{SAP} = I - M_{SAP}D$ with $M_{SAP} = M_{SAP}^{(1)}$.

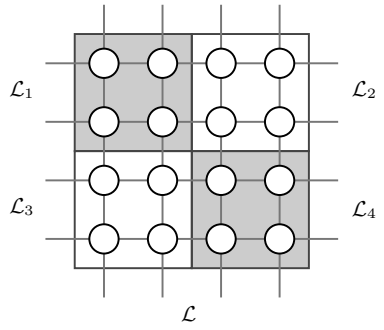


FIG. 2.5. *Block decomposed lattice (reduced to 2D) with 2 colors*

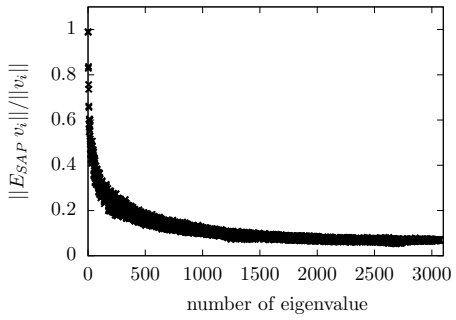


FIG. 2.6. *Error component reduction on a 4^4 lattice with block size 2^4*

Typically the solution of the local systems (2.8), 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 like FGMRES or GCR in case that SAP is used as a preconditioner (cf. [23, 34, 40]).

It turns out that SAP as a preconditioner is not able to remedy the unfavorable scaling behavior with respect to system size, quark mass and physical volume of Krylov subspace methods. 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 not affected by SAP. We illustrate this in Figure 2.6 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 2.6. 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. Algebraic Multigrid Methods. Any multigrid method consists of two components, namely a smoother and a coarse grid correction [13, 27, 39, 47]. Typically, the smoother can be chosen as a simple iterative method. This can be a relaxation scheme like Jacobi or Gauss-Seidel or their block variants as well as Krylov subspace methods. Given the properties of SAP presented in the previous section we choose SAP as our smoothing scheme in the QCD context.

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 2.6), 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. 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. We also need a linear map $R : \mathbb{C}^n \to \mathbb{C}^{n_c}$ to restrict information from the original space to the subspace and a linear map $P : \mathbb{C}^{n_c} \to \mathbb{C}^n$ which transports information back to the original space. The coarse grid correction for a current iterate z on the original space is then obtained by restricting the residual $r = b - Dz$ to the subspace, there solving

$$D_c e_c = Rr \tag{3.1}$$

and transporting the coarse error e_c back to the original space as a correction for z, resulting in the subspace correction

$$z \leftarrow z + PD_c^{-1}Rr, \quad r = b - Dz \tag{3.2}$$

with the corresponding error propagator

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

Typically, the coarse grid system is obtained as the Petrov-Galerkin projection with respect to P and R, i.e.,

$$D_c = RDP.$$

The coarse grid correction $I - P(RDP)^{-1}RD$ then 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 $\text{range}(R)$ 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 range(R) = range(RD) is spanned by left eigenvectors of D, then 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. This procedure can be recursively extended to true multigrid by formulating a two-level algorithm of this kind for the solution of (3.1) until we obtain an operator which is small enough to solve (3.1) directly.

To be computationally beneficial, solving (3.1) has to be much cheaper than solving the original equation $Dz = b$. For this purpose D_c has to be very small or sparse. As the number of eigenvectors that are not sufficiently reduced by the SAP smoother grows with n, cf. [35], one should not aim at fixing n_c (like in deflation methods), but at finding sparse matrices R and P whose ranges approximate the small left and right near kernel of D well, respectively.

3.1. Aggregation-based Intergrid Transfer Operators. Consider a block decomposition of the lattice $\mathcal{L}$ with lattice-blocks $\mathcal{L}_i$. It has been observed in [35] 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" in [35]. As a consequence, we can represent many eigenvectors with small eigenvalues from just a few by decomposing them into the parts belonging to each of the lattice-blocks. This is the philosophy behind the so-called aggregation-based intergrid transfer operators introduced in a general setting in [9, 13] and applied to QCD problems in [3, 12, 37].

DEFINITION 3.1. *An* aggregation $\{\mathcal{A}_1, \dots \mathcal{A}_s\}$ *is a partitioning of the set* $\mathcal{V} = \mathcal{L} \times \mathcal{C} \times \mathcal{S}$ *of all variables. It is termed a* lattice-block based *aggregation if each* $\mathcal{A}_i$ *is of the form*

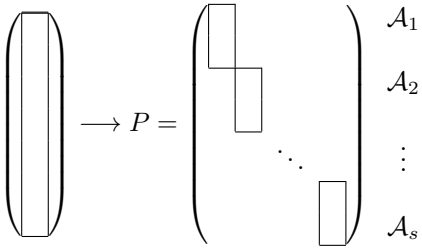
$$\mathcal{A}_i = \mathcal{L}_{j(i)} \times \mathcal{W}_i,$$

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

Aggregates for the lattice Wilson Dirac operator (2.5) will typically be realized as lattice-block based aggregates. 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 $\mathcal{L}$.

Starting from a set of *test vectors* $\{v_1, \dots, v_N\}$ which represent the near kernel and a set of aggregates $\{\mathcal{A}_1, \dots, \mathcal{A}_s\}$, the interpolation P is obtained by decomposing the test vectors over the aggregates

$$(v_1 \mid \dots \mid v_N) = \quad \longrightarrow P = \qquad \begin{matrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \\ \vdots \\ \mathcal{A}_s \end{matrix} . \tag{3.3}$$



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

$$Pe_{(i-1)N+j} = \mathcal{I}^T_{\mathcal{A}_i} v_j, \quad , i = 1, \dots, s,\ j = 1, \dots, N. \tag{3.4}$$

Herein, $\mathcal{I}_{\mathcal{A}_i}$ represents the trivial restriction operator for the aggregate $\mathcal{A}_i$, i.e., $\mathcal{I}_{\mathcal{A}_i}^T v_j$ 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 $\mathcal{I}_{\mathcal{A}_i}^T v_j$ in (3.4) by the j-th basis vector of an orthonormal basis of $\text{span}(\mathcal{I}_{\mathcal{A}_i}^T v_1, \ldots, \mathcal{I}_{\mathcal{A}_i}^T v_N)$. This does not alter the range of P nor does it change 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 3.1 illustrates a lattice-block based aggregation from a lattice point of view where in each aggregate $\mathcal{A}$ we take $\mathcal{W}_i$ as the whole set $\mathcal{C} \times \mathcal{S}$, again reduced to two dimensions. 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.

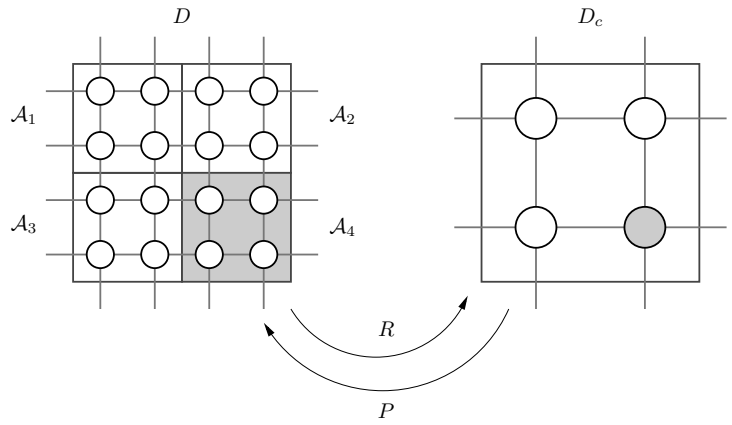


FIG. 3.1. *Aggregation-based interpolation (geometrical point of view reduced to 2D)*

3.2. 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 [3, 12, 35, 37] 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 2.1 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 to 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 to small eigenvalues.

As was pointed out in [3], it is furthermore possible to even obtain $R = P^H$ 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 3.2. *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 (3.3). 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 non-zero blocks in P belonging to a specific aggregate is either multiplied by $+1$ or by -1. This gives

$$\Gamma_5 P = P\Gamma_5^c. \tag{3.5}$$

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 3.3. *Let the aggregation be Γ_5-compatible and P the corresponding aggregation based prolongation as in* (3.3) *and $R = (\Gamma_5 P)^H$. Consider the two coarse grid operators*

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

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 we have $\mathcal{F}(D_c) \subseteq \mathcal{F}(D)$.

Proof. We first observe that just as Γ_5 the matrix 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 3.3 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 (matrix D_c^{PG} with $R = \Gamma_5 P$) or a Galerkin approach (matrix D_c, restriction is the adjoint of the prolongation). The Petrov-Galerkin matrix D_c^{PG} inherits the hermiticity of the matrix $\Gamma_5 D$, whereas the Galerkin matrix D_c inherits the Γ_5-symmetry (and thus the symmetry of the spectrum, see Lemma 2.1) 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 [25], i.e., the number of negative eigenvalues is about the same as the positive ones. This property is also inherited by 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. Indeed, if

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

denotes the eigendecomposition of the hermitian matrix $\Gamma_5 D$, then

$$D = (\Gamma_5 V \text{sign}(\Lambda)) \; |\Lambda| \; V^H = U \Sigma V^H \tag{3.6}$$

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

The theory of algebraic multigrid methods for non-hermitian problems recently developed in [14] 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 (3.6). 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 3.4. *Let $\mathcal{L}_j, j = 1, \ldots, s_L$ be a block decomposition of the lattice $\mathcal{L}$. Then the* standard aggregation *$\{\mathcal{A}_{j,\sigma}, j = 1, \ldots, s_L, \sigma = 0, 1\}$ is given by*

$$\mathcal{A}_{j,0} = \mathcal{L}_j \times \{0, 1\} \times \mathcal{C}, \;\; \mathcal{A}_{j,1} = \mathcal{L}_j \times \{2, 3\} \times \mathcal{C}.$$

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 two only 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 $n_c = 2N n_{\mathcal{L}_c}$.

With standard aggregation, in addition to the properties listed in Lemma 3.3 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.

We also note that applying R and P to a vector does not require any communication in a parallel implementation if whole aggregates are assigned to one process.

3.3. Adaptivity in Aggregation-based AMG. 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*. We now briefly review the setup concept of adaptive (smoothed) aggregation as described in [13]. 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 (3.3). This construction guarantees that v is in range(P) and thus is treated on the coarse grid. Once a first two- 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 [13], 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 [3, 12, 37].

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.

3.4. 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 done in the *bootstrap* approach pursued in [10, 11] which we explain now. The following observation is crucial: Given an eigenpair (v_c, λ_c) of the generalized eigenvalue problem on the coarse grid

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

we observe that (Pv_c, λ_c) solves the constrained eigenvalue problem

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

on the fine grid. This observation allows to use the coarse grid system as a source of information about the eigenvectors to small eigenvalues of the fine grid system. Computing eigenvectors to 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 [35], explained in the next section, 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. Multigrid and Inexact Deflation. A hierarchical approach for solving the Wilson Dirac equation (1.1), which lately received attention in the Lattice QCD community, was proposed in [35]. It is a combination of what is called "inexact deflation" with an SAP preconditioned generalized conjugate residuals (GCR) method. The paper [35] does not relate its approach to the existing multigrid literature. The purpose of this section is to recast the formulations from [35] into established terminology from algebraic multigrid theory and to explain the limitations of the overall method from [35] which composes its multigrid ingredients in a non-optimal manner. We also

explain how the setup employed in [35] to construct the "inexact deflation subspace" (i.e., the test vectors) can be viewed and used as an approximate bootstrap setup in the sense of Section 3.4.

4.1. Inexact Deflation. The inexact deflation subspace constructed in [35] is the range of a linear operator P which resembles the definition of aggregation based interpolation from (3.3). As in the aggregation-based construction it uses a set of test vectors $v_1, \ldots, v_N$ which are "chopped" up over aggregates (called subdomains in [35]) to obtain the locally supported columns of P. These aggregates are not Γ_5-compatible, so 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 a (purely algebraic) deflating technique is applied when solving the coarse system.

Two projections π_L, π_R are defined in [35] as follows

$$\pi_L = I - D P D_c^{-1} P^H \quad \text{and} \quad \pi_R = I - P D_c^{-1} P^H D. \tag{4.1}$$

Clearly π_R is the the coarse grid correction introduced in section 3; cf. Lemma 3.3(i). In the context of inexact deflation [35] uses these projections and the relation $D\pi_R = \pi_L D$ to decompose the linear system of equations $Dz = b$ as

$$D\pi_R z = \pi_L b, \quad D(I - \pi_R)z = (I - \pi_L)b.$$

The second equation can be simplified to $(I - \pi_R)z = P D_c^{-1} P^H b$. Thus the solution z can be computed as $z = \pi_R z + (I - \pi_R)z = \chi + \chi'$, where

$$\chi' = P D_c^{-1} P^H b$$

only requires the solution of the coarse grid system D_c and

$$D\chi = D\pi_R \chi = \pi_L b$$

is the "inexactly deflated" system which in [35] 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 $\pi_L b$, and the Krylov subspace method is GCR (general conjugate residuals, cf. [40]), a minimum residual approach which automatically adapts itself to the fact that the preconditioner $M_{SAP}^{(\nu)}$ is not stationary, see the discussion in section 2.3.

4.2. Comparison of Multigrid and Inexact Deflation. Although the ingredients of an aggregation based algebraic multigrid method as described in section 3 and of "inexact deflation" as described in the previous paragraph 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

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

Hence we obtain for one iteration of the V-cycle

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

where z denotes the current iterate and r the current residual $b - Dz$, and

$$C^{(\nu)} = M_{SAP}^{(\nu)} + P D_c^{-1} P^H - M_{SAP}^{(\nu)} D P D_c^{-1} P^H \,. \tag{4.2}$$

In terms of the projectors (4.1) this can be written as

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

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 (4.2), i.e., the solution of systems with the matrix D_c^{-1} 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. On the other hand, in the "inexact deflation" context the exact splitting of the solution $z = \chi' + \chi$ with

$$\chi' = P D_c^{-1} P^H b, \;\; D \pi_R \chi = \pi_L b$$

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 [43, 48]).

The difference of the two approaches is now apparent. In the multigrid context we are able to relax on the accuracy of the coarse system, 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 are allowed to 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 particularly efficiently be obtained by a recursive application of the two-grid approach, resulting in a true multigrid method.

For a more detailed analysis of the connection between deflation methods (including inexact deflation) and multigrid approaches we refer to [28, 38, 46], e.g.

4.3. 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 inexact deflation operators. Once these vectors are found the method is completely defined

(see section 4.1). In complete analogy to the discussion of adaptive algebraic multigrid methods in sections 3.3 and 3.4, 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 [35] is similar in nature to the one described in section 3.3, 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 ψ_j and approximately computes $D^{-1}\psi_j$ using SAP. The (approximate) multiplication with D^{-1} will amplify the components of ψ belonging to the near kernel. These new vectors are now used to define P (and D_c), yielding an inexact deflation method which can anew be used to approximately compute $D^{-1}\psi_j$ giving new vectors for P. The whole process is repeated several times; see Algorithm 2 for a detailed description where a total of n_{inv} of these cycles is performed.

Algorithm 2 Inexact deflation setup – IDsetup(n_{inv},ν) as used in [35]

1: Let $v_1, \ldots, v_N \in \mathbb{C}^n$ be random test vectors
2: **for** $\eta = 1$ to 3 **do**
3: **for** $j = 1$ to N **do**
4: $v_j \leftarrow M_{SAP}^{(\eta)} v_j$
5: **end for**
6: **end for**
7: **for** $\eta = 1$ to n_{inv} **do**
8: (re-)construct P and D_c from current $v_1, \ldots, v_N$
9: **for** $j = 1$ to N **do**
10: $v_j \leftarrow (M_{SAP}^{(\nu)} \pi_L + P D_c^{-1} P^H) v_j$
11: $v_j \leftarrow \frac{v_j}{||v_j||}$
12: **end for**
13: **end for**

The update $v_j \leftarrow (M_{SAP}^{(\nu)} \pi_L + P D_c^{-1} P^H) v_j$ in line 10 of the algorithm is equivalent to the application of the V-cycle iteration matrix $C^{(\nu)}$ (cf. (4.2)). It can be interpreted as one step of an iteration to solve $Dv = v_j$ with initial guess 0 and iteration matrix $C^{(\nu)}$.

This update of the test vectors can also be viewed in terms of the bootstrap AMG setup outlined in section 3.4. 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 3.3), the second part of the update, $P D_c^{-1} P^H v_j$ is in range(P). In contrast to the bootstrap methodology (cf. section 3.4) where an update in range(P) is obtained by interpolating eigenvectors to small eigenvalues of D_c, in the inexact deflation variant these "optimal" vectors are only approximated.

5. DD-αAMG. We now have all the ingredients available to describe our domain decomposition/aggregation based adaptive algebraic multigrid (DD-αAMG) method for the Wilson Dirac operator (1.1).

As its smoother we take $M_{SAP}^{(\nu)}$, i.e., we perform ν iterations of red-black Schwarz as formulated in Algorithm 1. 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 the adaptive setup phase. The aggregates are

from a standard aggregation according to Definition 3.4, implying that it is in particular lattice-block based and Γ_5-compatible. Parameters of the aggregation are the underlying block decomposition of $\mathcal{L}$ (which does not necessarily match the one underlying the SAP smoother) and the test vectors $v_1, \ldots, v_N$ upon which P is built. The coarse grid matrix D_c inherits all of the important properties of D, cf. Lemma 3.3.

We combine the smoothing iteration and the coarse grid correction into a standard V-cycle with no pre- and ν steps of post smoothing so that the iteration matrix of one V-cycle is given by $C^{(\nu)}$ from (4.2). Instead of using iterations with the V-cycle as a stand-alone solver, we run FGMRES, the flexible GMRES method (cf. [40]) with one V-cycle used as a (right) preconditioner.

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 2) is the most efficient. The modification is a change in the update of the vectors v_j in the second half. Instead of doing one iteration 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 3.

Algorithm 3 DD-αAMG-setup(n_{inv}, ν)

perform Algorithm 2 with line 10 replaced by
$v_j \leftarrow v_j + C^{(\nu)}(v_j - Dv_j) \quad \{= C^{(\nu)}v_j + (I - C^{(\nu)}D)v_j\}$

6. Numerical Results. We implemented the DD-αAMG method in the programming language `C` using the parallelization interface of `MPI`. Up to now, we only have an implementation of a two-grid method at hand for which the numerical experiments already show truly satisfactory results. As we will see, we spend a large amount of time solving the coarse grid system, though. This indicates the potential for additional speed up when the method is applied recursively in a true multigrid approach. We will come back to this point in the conclusions in Section 7.

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 we perform the V-cycle of the preconditioner in single precision. Low level optimization (e.g., making use of the SSE-registers on Intel architectures) has not been considered, yet. All Krylov subspace methods (FGMRES, 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 CGNR, the CG method applied to the normal equation $D^H D\psi = D^H b$. Let us note already at this stage that in CGNR we used the residual with respect to the original equation $D\psi = b$ for the stopping criterion. This residual can be obtained at the cost of one AXPY operation in the CGNR code. We include a comparison with the inexact deflation approach, since an efficient implementation is publicly available. A further comparison with the multilevel approaches from [3, 37] would have required a very substantial additional implementation effort which is out of scope for this paper.

A commonly used technique in lattice QCD computations is odd-even preconditioning. A lattice site x is called even if $x_1 + x_2 + x_3 + x_4$ is even, else it is called odd. Due to the nearest neighbor coupling, 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. Herein, D_{ee} and D_{oo} are block diagonal with 12×12 diagonal blocks. Instead of solving a system with D we can solve the corresponding system for the odd lattice sites given by the Schur complement $D_S = D_{oo} - D_{oe} D_{ee}^{-1} D_{eo}$ and then retrieve the solution at the even lattice sites, cf. [37]. The inverse D_{ee}^{-1} is precomputed once for all, and the operator D_S is applied in factorized form. A matrix-vector multiplication with D_S thus requires the same work than 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. Our results for CGNR do not use the odd-even reduced system, since an efficient implementation would require a storage scheme for the links $U_\mu(x)$ different from the one needed for DD-αAMG. We do, however, use odd-even preconditioning when we solve the coarse system involving D_c in DD-αAMG, which by default is done using odd-even preconditioned restarted GMRES with a restart length of 30. We implemented this odd-even preconditioning similarly in spirit to what was proposed for the Wilson Dirac operator in [31].

	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	5
	size of lattice-blocks in SAP$^{(*)}$		2^4
	number of Minimal Residual (MR) iterations to solve the local systems (2.8) in SAP$^{(*)}$		3

TABLE 6.1
Parameters for the DD-αAMG two-level method. $(*)$ *: same in solver and setup*

Table 6.1 summarizes the default parameters used for DD-αAMG in our experiments. Besides those discussed in Section 5, the table also gives 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 solve the local systems (2.8). Instead of requiring a certain decrease in the residual we here fix the number of iterative steps (to 3). The iterative method we use here is the minimal residual method MR, i.e., restarted GMRES with a restart length of 1, where each iterative step is particularly cheap.

For the various configurations and respective matrices 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 and 2^4) 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 tuned. 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. 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 6.2. The default $n_{inv} = 6$ given in Table 6.1 should be regarded as a good compromise.

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	48×16^3	250	7,055	−0.095300	1.00000	BMW-c [20, 21]
2	48×24^3	250	11,664	−0.095300	1.00000	BMW-c [20, 21]
3	48×32^3	250	15,872	−0.095300	1.00000	BMW-c [20, 21]
4	48×48^3	135	53,932	−0.099330	1.00000	BMW-c [20, 21]
5	64×64^3	135	84,207	−0.052940	1.00000	BMW-c [20, 21]
6	128×64^3	270	45,804	−0.342623	1.75150	CLS [16, 22]

TABLE 6.2

Configurations used together with their parameters. For details about their generation we refer to the references. Pion masses rounded to steps of 5 *MeV.*

The configurations we used are listed in Table 6.2. In principle the pion mass m_π and the lattice spacing (not listed) determine the condition of the respective matrix, e.g., the smaller m_π, the more ill conditioned is the respective matrix. The physical pion mass is $m_{\pi_{phys}} = 135$ MeV which is taken on by the configurations 4 and 5. The conditioning of the matrices is indicated by the iteration count of CGNR to decrease the residual by a factor of 10^{10}.

We ran DD-αAMG on the various configurations, analyzed the behavior of the setup routine and performed different scaling test. All results have been computed on the Juropa machine at Jülich Supercomputing Centre, a cluster with 2,208 compute nodes, each with two Intel Xeon X5570 (Nehalem-EP) quad-core processors. Unless stated otherwise the `icc`-compiler with the optimization flag `-O3` was used.

6.1. Comparison with CGNR. First we compare CGNR 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 $n = 201{,}326{,}592$.

	CGNR	DD-αAMG	speed up factor	coarse grid
setup time		15.38s		6.96s
solve iter	84,207	20		$3{,}997^{(*)}$
solve time	816.26s	5.17s	157.9	4.11s
total time	816.26s	20.55s	39.7	11.07s

TABLE 6.3

CGNR vs. DD-αAMG with default parameters (Table 6.1) on an ill-conditioned 64^4 *lattice (Table 6.2: id 5),* 8,192 *cores,* $(*)$ *: total iteration count summed up over all solver iterations.*

The results reported in Table 6.3 show that we obtain a speed-up factor of 40 over CGNR with respect to the total timing. Excluding the setup time, we gain a factor of 158. The right most column of Table 6.3 shows that in this ill-conditioned case about 80% of the solve time go into computations on the coarse grid.

6.2. Setup Evaluation. Lattice QCD computations are dominated by two major tasks: generating configurations within the Hybrid Monte-Carlo (HMC) algorithm [30] 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 solved only once 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 typically requires several 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	2.02	349.6	343	356	19.67	21.69
2	3.19	122.2	118	127	8.26	11.45
3	4.63	50.8	50	52	4.38	9.01
4	6.87	31.3	31	32	3.14	10.01
5	10.24	25.2	25	26	2.68	12.92
6	14.43	23.0	23	23	2.52	16.95
7	18.30	22.0	22	22	2.55	20.85
8	21.97	21.7	21	22	2.67	24.64
9	24.97	21.0	21	21	2.73	26.70
10	28.14	21.3	21	22	2.86	31.00
11	31.05	21.9	21	22	3.05	34.10
12	33.75	22.1	22	23	3.11	36.86

TABLE 6.4

Evaluation of DD-αAMG-setup(n_{inv}, 5) cf. Algorithm 3, 48^4 *lattice, ill-conditioned configuration (Table 6.2: id 4),* 2,592 *cores, averaged over 10 runs.*

Table 6.4 illustrates how the ratio between setup and solve can be balanced depending on the amount of right hand sides. In this particular case 3 steps in the setup might be the best choice if only a single solution of the system is needed. For several right hand sides 6 steps might be the best choice. Doing up to 9 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 numbers shown have been averaged over 10 runs, because the measurements vary due to the choice of random initial test vectors. The fourth and the fifth column of Table 6.4 show that the fluctuations in the iteration count of the solver are modest. For $n_{inv} \geq 3$ the fluctuations almost vanish completely.

	DD-αAMG iteration counts					
n_{inv}	conf 1	conf 2	conf 3	conf 4	conf 5	conf 6
3	54	58	56	55	54	65
6	23	24	24	24	23	23

TABLE 6.5

Configuration dependence study of DD-αAMG with DD-αAMG-setup(n_{inv}, 5), ill-conditioned 48^4 *lattice for 6 different, ill-conditioned configurations on* 48^4 *lattices, (Table 6.2: id 4),* 2,592 *cores.*

Table 6.5 gives the iteration count of DD-αAMG for a set of 6 stochastically independent configurations from a single HMC simulation. If we invest just 3 iterations

in the setup we observe a (tolerable) variation of the iteration count, whereas for 6 iterations in the setup the iteration count becomes very stable.

6.3. Scaling Tests. We now study the scaling behavior of the solver as a function of the mass parameter and the system size. While the first determines the condition number of the Wilson Dirac operator, the second has an effect on the density of the eigenvalues. In particular, increasing the volume leads to a higher density of small eigenvalues [6]. In a weak scaling test we also analyze the performance as a function of the number of processors used.

6.3.1. 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 Wilson Dirac operator (2.4). This represents the most ill-conditioned system where the pion mass with 135 MeV is physical. We then used this interpolation operator for a variety of other mass parameters, where we then ran the DD-αAMG solver without any further setup.

	CGNR		DD-αAMG		coarse system	
m_0	iteration count	solver timing	iteration count	solver timing	⌀iteration count	timing (% solve time)
−0.03933	1,597	14.1	15	0.83	10	0.13 (15.5)
−0.04933	1,937	17.2	16	0.89	11	0.15 (16.6)
−0.05933	2,454	21.8	17	0.95	13	0.18 (18.9)
−0.06933	3,320	29.4	18	1.04	16	0.22 (21.1)
−0.07933	5,102	45.3	18	1.13	20	0.29 (25.1)
−0.08933	10,294	91.5	20	1.44	31	0.50 (34.8)
−0.09033	11,305	100.3	20	1.47	33	0.53 (35.8)
−0.09133	12,527	111.2	20	1.43	36	0.53 (37.1)
−0.09233	14,009	124.4	20	1.48	38	0.57 (38.4)
−0.09333	15,869	141.3	21	1.68	41	0.67 (39.7)
−0.09433	18,608	165.5	21	1.68	45	0.71 (42.2)
−0.09533	22,580	201.2	21	1.70	49	0.75 (43.9)
−0.09633	27,434	244.4	21	1.79	54	0.82 (45.7)
−0.09733	33,276	296.5	22	2.15	63	1.08 (50.4)
−0.09833	42,067	373.7	22	2.30	74	1.24 (53.9)
−0.09933	53,932	480.4	23	2.60	86	1.49 (57.4)

TABLE 6.6
Mass scaling behavior of DD-αAMG, 48^4 lattice (Table 6.2: id 4), 2,592 cores.

In Table 6.6 we compare CGNR and DD-αAMG with respect to the timing for one right hand side and the scaling with the mass parameter m_0. For the smallest m_0, DD-αAMG is 185 times faster than CGNR and even for the largest value of m_0 there remains a factor of 17. We also see that the two methods scale in a completely different manner. The CGNR solve for the smallest m_0 is 34 times more expensive than the solve for the largest one. On the other hand the DD-αAMG timings just increases by a factor of 3.1, the iteration count even only by a factor of 1.5. The coarse grid iteration count, however, increases by a factor of 8.6.

6.3.2. Scaling of the Error. In all results shown so far the stopping criterion was based on the norm of the residual r, since this is a quantity which is available computationally. Ultimately we are interested in the norm of the error e. Since we have $r = De$ the error components belonging to small eigenvalues of D have

strong influence on the solution and weak influence on the residual. Obviously, $\|e\| \leq \|D^{-1}\| \cdot \|r\|$, and in the worst case we have equality. As the norm of D^{-1} tends to be quite large for our ill-conditioned systems, it is worthwhile to compare CGNR with our two-grid method.

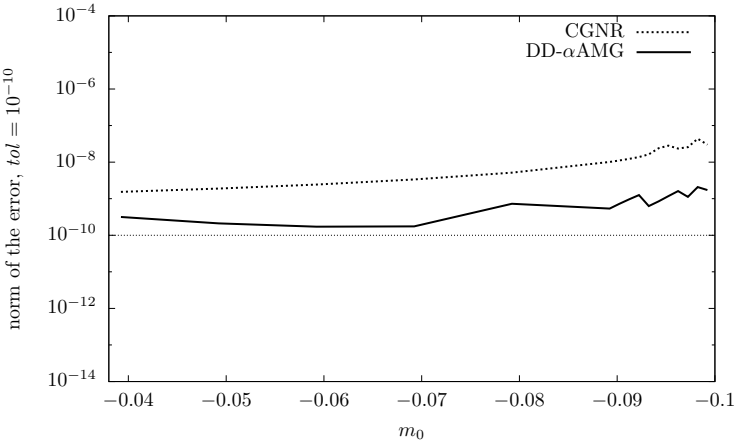


FIG. 6.1. *Behavior of the error as a function of m_0.*

We studied systems with different mass parameters m_0 with predetermined solutions and compared the error norms obtained by both methods with the stopping criterion to reduce the norm of the initial residual by 10^{10}. Figure 6.1 shows that the error of DD-αAMG tends to be one order of magnitude better than the error of CGNR. Both errors show almost the same scaling behavior as a function of m_0,with the error becoming larger as the system gets more ill-conditioned.

6.3.3. System Size Scaling. In Table 6.7 we report tests on the scaling with the system size for constant mass parameter and (physical) lattice spacing. We again compare DD-αAMG with CGNR. The iteration count of CGNR for $N_t \times N_s^3$ lattices appears to scale with N_s and thus approximately doubles from $N_s = 16$ to $N_s = 32$. For DD-αAMG we observe a scaling of the iteration count with $\log(N_s)$. The solver timing scales somewhat less well due to the increased time spent on the coarse grid. Overall, the time to solution for CGNR increases by a factor of 2.36, whereas for DD-αAMG it is only a factor of 1.51.

	CGNR		DD-αAMG		
lattice size $N_t \times N_s^3$	iteration count	solver timing	setup timing	iteration count	solver timing
48×16^3	7,055	55.9s	4.14	22	1.32
48×24^3	11,664	96.2s	4.22	26	1.65
48×32^3	15,872	131.9s	4.33	30	1.99

TABLE 6.7

Lattice size scaling of DD-αAMG, $n_{inv} = 3$ setup iterations, lattices generated with the same mass parameter and lattice spacing (Table 6.2: id 1, 2 and 3), local lattice size 4×8^3.

6.3.4. Weak Scaling. For a weak scaling test we ran 100 iterations of DD-αAMG with $n_{inv} = 3$ in the setup on lattices ranging from size $16^2 \cdot 8^2$ on a single

node (8 cores/node) to 64^4 on 1,024 nodes with $4 \cdot 8^3$ local lattice size on each core, cf. Figure 6.2.

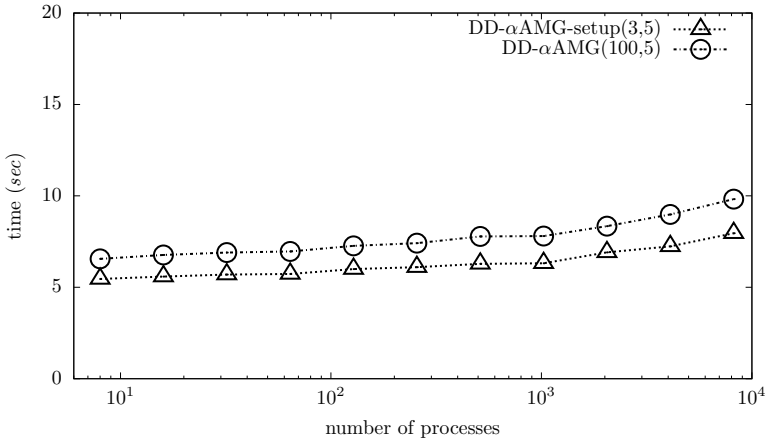


FIG. 6.2. *Weak scaling test of DD-αAMG. The lattice size is increased with the number of processes, keeping the local lattice size per process fixed to* $4 \cdot 8^3$.

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. We therefore see the usual $\log(p)$ dependence, p the number of processes, caused by global communication. Apart from this, our method scales well up to 8,192 processes.

6.4. Comparison with the Inexact Deflation Method. Since the inexact deflation code of [35] is publicly available [33], we can compare its performance with DD-αAMG. The standard size for the lattice-blocks in the SAP iteration in the inexact deflation code is 4^4, which is what we used in our experiments. The local systems (2.8) on the lattice-blocks are solved in [33] via the odd-even preconditioned minimal residual (MR) method with the iteration number fixed to 4. This ensures that the local systems are solved accurately enough with relatively low computational cost and turns out to be optimal or nearly optimal for the overall execution time. These parameters for inexact deflation are thus different from the ones we use in DD-αAMG, where the lattice-block size is 2^4 and 3 steps of the non-preconditioned minimal residual method are performed.

6.4.1. Comparison Without Low-level Optimization. The following results were produced on the same 48^4 lattice as in Sections 6.2 and 6.3.1 and on a 128×64^3 lattice (Table 6.2, id 6). Except for the coarse grid tolerance and the SAP block size we have chosen a common set of parameters (cf. Table 6.1) and the Intel `icc` compiler with optimization flag `-O3`.

Table 6.8 compares inexact deflation and DD-αAMG for a whole range for n_{inv}, the number of iterations in the setup. We see that the DD-αAMG provides the fastest solver for $n_{inv} = 6$ requiring 2.50s. The inexact deflation setup provides the fastest solver for $n_{inv} = 10$, but this solver needs a factor of 2.2 more time than the DD-αAMG solver and the setup by itself is also (slightly) more expensive. If we look at the time for setup and solve for one right hand side, $n_{inv} = 3$ is best for the DD-αAMG method, where we obtain a total timing of 9.01s. Here, the best choice for

	Inexact deflation			DD-αAMG		
setup steps n_{inv}	setup timing	iteration count (coarse)	solver timing	setup timing	iteration count (coarse)	solver timing
1	2.03s	233 (82)	18.3s	1.99s	350 (12)	19.7s
2	3.19s	155 (145)	14.7s	3.17s	120 (29)	8.05s
3	4.45s	108 (224)	12.1s	4.58s	52 (54)	4.43s
4	5.88s	84 (301)	10.5s	6.95s	32 (74)	3.28s
5	7.71s	70 (320)	8.86s	10.3s	25 (81)	2.60s
6	9.22s	63 (282)	7.30s	14.2s	23 (86)	2.50s
7	10.6s	58 (277)	6.53s	18.3s	22 (100)	2.62s
8	12.3s	54 (267)	6.07s	22.7s	22 (102)	2.61s
9	14.1s	51 (263)	5.53s	24.9s	21 (116)	2.70s
10	15.9s	49 (265)	5.44s	27.6s	21 (127)	2.83s
11	17.5s	50 (266)	5.48s	30.6s	22 (129)	3.06s
12	19.5s	53 (254)	5.72s	33.8s	23 (131)	3.27s

TABLE 6.8

Comparison with fair compiler settings of DD-αAMG with inexact deflation, both with 5 *SAP iterations in each setup step, both methods tuned equally, except SAP block size* 4^4 *with MR(4) and coarse system solver tolerance* 10^{-12} *in ID, ill-conditioned system on a* 48^4 *lattice (Table 6.2: id 4),* 2,592 *cores.*

inexact deflation is $n_{inv} = 6$ with a total timing of 16.2s.

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. This might change, however, in a true multigrid approach. 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 has to be solved very accurately, the number of iterations needed to solve the coarse grid system is higher than in DD-αAMG. It is only moderately (a factor of 2 to 4) higher, though, because the code from [33] uses an additional adaptively computed preconditioner for the GCR iterations on the coarse system, whereas we just use the less efficient odd-even preconditioning in DD-αAMG.

For the same number of test vectors, DD-αAMG produces a coarse system twice as large 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.

We also compared both methods in Table 6.9 for another configuration typical for many recent lattice QCD computations. The distribution function for the links $U_\mu(x)$ in configuration 6 is quite different from that in configuration 4 from Table 6.8. Again we took the default parameter set, but now with a relatively cheap setup ($n_{inv} = 4$). This results in a speed up factor of more than 1.5 for setup and solve in DD-αAMG against inexact deflation. Still 56% of the execution time is spent in solves of the coarse system in DD-αAMG.

	Inexact deflation	DD-αAMG	speed up factor
setup iter	6	4	
setup time	22.3s	14.8s	1.51
solve iter	37	40	
solve time	16.6s	10.1s	1.64
total time	38.9s	24.9s	1.56

TABLE 6.9

Comparison of DD-αAMG with inexact deflation on an ill-conditioned system on a 128×64^3 lattice (Table 6.2, id 6), same parameters as in Table 6.8, 8,192 *cores.*

	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	1.78s	350 (12)	19.1s
2	1.87s	155 (145)	10.2s	2.76s	122 (29)	7.66s
3	2.69s	108 (224)	9.96s	4.33s	51 (55)	4.45s
4	3.43s	84 (301)	9.25s	6.47s	31 (73)	2.69s
5	6.14s	70 (320)	7.50s	9.02s	25 (80)	2.54s
6	5.68s	63 (282)	5.21s	13.5s	23 (86)	2.49s
7	6.93s	58 (277)	4.67s	16.6s	22 (96)	2.23s
8	7.71s	54 (267)	4.12s	20.5s	22 (108)	2.35s
9	8.74s	51 (263)	3.89s	21.7s	21 (118)	2.62s
10	10.1s	49 (265)	3.62s	25.2s	21 (126)	2.77s
11	11.3s	50 (266)	3.77s	28.7s	22 (129)	3.08s
12	12.6s	53 (254)	4.13s	32.5s	22 (132)	2.69s

TABLE 6.10

Comparison with best compiler settings for each method using the same set of parameters as in Table 6.8

6.4.2. Comparison With Low-level Optimization. The inexact deflation code provides assembly coded parts which allow to optimally use the SSE registers of Intel architectures. For the sake of completeness we have repeated the runs from Table 6.8 for both methods with the respectively best compiler options available. This means that we used the `gcc` compiler with the `-O3` flag and customized `SSE` optimization for the inexact deflation method and the `icc` compiler with the optimization flags `-O3 -ipo -axSSE4.2 -m64` for for the DD-αAMG method. Since our focus is on algorithmic improvements we did not work on customized SSE optimization for DD-αAMG, although this should, in principle, give additional speed-up. Table 6.10 shows that a DD-αAMG setup with $n_{inv} = 7$ provides the fastest DD-αAMG solver which is still 1.6 times faster than the fastest inexact deflation solver for $n_{inv} = 10$. The sum of DD-αAMG setup and solver timing is minimized for $n_{inv} = 3$ and this is still 1.2 times less than the minimum of the inexact deflation setup and solver timing for $n_{inv} = 6$.

7. Conclusions and Outlook. The developed 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 and also the fastest publicly available solver 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 and the efficient setup procedure into the algebraic multigrid method. Additional improvements can be expected by a recursive application in a true multigrid method. For example, if in the situation of Table 6.3 true multigrid reduces the part of the work spent on the coarse grid from 80% to 20%—so that it is well balanced with the 20% spent on the fine grid—we obtain another speed-up factor of 2.5. In addition, true multigrid should also lead to significant improvements when moving towards ever larger lattice sizes and we expect it to improve the scaling with respect to the mass parameter and the system size. Finally, in accordance with observations for multigrid methods for elliptic PDEs made in [47], we can speculate on further improvements over the observations in Figure 6.1, i.e., that in a true multigrid method the norm of the error is even closer to that of the residual than in the two-grid method.

Another factor of 1.5 - 2 could probably be obtained by machine specific optimization (SSE/AVX/QPX). We are currently working on the implementation of additional levels and incorporating our algorithm into the production codes of our collaborators within SFB/TRR55. Furthermore we are planning to investigate how to incorporate our DD-αAMG method into the Hybrid Monte Carlo Method, i.e., updating the multigrid hierarchy in a cost efficient way.

Acknowledgments. We thank the Budapest-Marseille-Wuppertal collaboration for providing configurations and compute time on Juropa at Jülich Supercomputing Centre (JSC) where most of the results were computed. We would also like to acknowledge James Brannick (Pennsylvania State University) for his advice regarding the development of our multigrid method, especially the extension of the bootstrap setup to the QCD context. Further thanks go to Kalman Szabó (Bergische Universität Wuppertal) for his support and counsel regarding the implementation of the multigrid solver and Wolfgang Söldner (University of Regensburg) for his time spent discussing our code and helping out with the I/O-interfaces.

REFERENCES

[1] C. Alexandrou, M. Brinet, J. Carbonell, M. Constantinou, P. Harraud, et al., *Nucleon electromagnetic form factors in twisted mass lattice QCD*, Phys. Rev., D83:094502 (2011).

[2] S. Aoki et al., *2+1 flavor lattice QCD toward the physical point*, Phys. Rev., D79:034503 (2009).

[3] 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).

[4] T. Bae, Y.-C. Jang, C. Jung, H.-J. Kim, J. Kim, et al., *Kaon B-parameter from improved staggered fermions in $n_f = 2 + 1$ QCD*, Phys. Rev. Lett., 109:041601 (2012).

[5] G. Bali, P. Bruns, S. Collins, M. Deka, B. Glasle, et al., *Nucleon mass and sigma term from lattice QCD with two light fermion flavors*, Nucl. Phys., B866 (2013), pp. 1–25.

[6] T. Banks and A. Casher, *Chiral Symmetry Breaking in Confining Theories*, Nucl.Phys., B169 (1980), p. 103.

[7] R. Ben-Av, M. Harmatz, S. Solomon, and P. G. Lauwers, *Fermion simulations using parallel transported multigrid*, Phys. Lett., B253 (1991), pp. 185–192.

[8] T. Bergrath, M. Ramalho, R. Kenway, et al., *PRACE scientific annual report 2012*, tech. report, PRACE, 2012. `http://www.prace-ri.eu/IMG/pdf/PRACE_Scientific_Annual_Report_2012.pdf`, p. 32.

[9] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393.

[10] A. Brandt, *Multiscale scientific computation: Review 2001*, in Multiscale and Multiresolution Methods, T. J. Barth, T. Chan, and R. Haimes, eds., vol. 20 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2002, pp. 3–95.

[11] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG.*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.
[12] 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).
[13] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM Review, 47 (2005), pp. 317–346.
[14] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders, *Towards adaptive smoothed aggregation (αSA) for nonsymmetric systems*, SIAM J. Sci. Comput., 32 (2010), pp. 14–39.
[15] R. Brower, E. Myers, C. Rebbi, and K. Moriarty, *The multigrid method for fermion calculations in quantum chromodynamics*, Tech. Report Print-87-0335, IAS,Princeton, 1987.
[16] CLS, *Coordinated lattice simulation.* `https://twiki.cern.ch/twiki/bin/view/CLS/`.
[17] T. DeGrand and C. E. Detar, *Lattice Methods for Quantum Chromodynamics*, World Scientific, 2006.
[18] T. A. DeGrand and P. Rossi, *Conditioning techniques for dynamical fermions*, Comput. Phys. Commun., 60 (1990), pp. 211–214.
[19] 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 (2008), pp. 1224–1227.
[20] S. Durr, Z. Fodor, C. Hoelbling, S. Katz, S. Krieg, et al., *Lattice QCD at the physical point: Simulation and analysis details*, JHEP, 08(2011)148 (2011).
[21] S. Durr, 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, (2011), pp. 265–268.
[22] P. Fritzsch, F. Knechtli, B. Leder, M. Marinkovic, S. Schaefer, et al., *The strange quark mass and Lambda parameter of two flavor QCD*, Nucl. Phys., B865 (2012), pp. 397–429.
[23] A. Frommer, A. Nobile, and P. Zingler, *Deflation and flexible SAP-preconditioning of GMRES in lattice QCD simulation*, tech. report, 2012. arXiv:1204.5463 [hep-lat].
[24] C. Gattringer and C. B. Lang, *Quantum Chromodynamics on the Lattice*, vol. 788 of Lect. Notes Phys., Springer, 1st ed., 2009.
[25] I. Gohberg, P. Lancaster, and L. Rodman, *Indefinite Linear Algebra and Applications*, Birkhäuser, Basel, 2005.
[26] M. Guest, G. Aloisio, R. Kenway, et al., *The scientific case for HPC in Europe 2012 - 2020*, tech. report, PRACE, October 2012. `http://www.prace-ri.eu/PRACE-The-Scientific-Case-for-HPC`, p. 75.
[27] W. Hackbusch, *Multi-Grid Methods and Applications*, vol. 4 of Springer Series in Computational Mathematics, Springer, 1st ed., 2003.
[28] K. Kahl and H. Rittich, *Analysis of the deflated conjugate gradient method based on symmetric multigrid theory.* Preprint BUW-IMACM 12/19, 2012.
[29] T. Kalkreuter, *Multigrid methods for propagators in lattice gauge theories*, J. Comput. Appl. Math., 63 (1995), pp. 57–68.
[30] A. Kennedy, *Algorithms for dynamical fermions*, arXiv:hep-lat/0607038, (2006).
[31] S. Krieg and T. Lippert, *Tuning lattice QCD to petascale on Blue Gene/P*, NIC Symposium 2010, (2010), pp. 155–164.
[32] T. Lippert, *Parallel SSOR preconditioning for lattice QCD*, Parallel Computing, 25 (1999), pp. 1357–1370.
[33] 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.
[34] M. Lüscher, *Solution of the Dirac equation in lattice QCD using a domain decomposition method*, Comput. Phys. Commun. 156, (2004), pp. 209–220.
[35] ———, *Local coherence and deflation of the low quark modes in lattice QCD*, JHEP, 07(2007)081 (2007).
[36] I. Montvay and G. Münster, *Quantum Fields on a Lattice*, Cambridge Monographs on Mathematical Physics, Cambridge University Press, 1994.
[37] J. Osborn, R. Babich, J. Brannick, R. Brower, M. Clark, et al., *Multigrid solver for clover fermions*, PoS, LATTICE2010:037 (2010).
[38] H. Rittich, *Deflation in multigrid methods*, master's thesis, Bergische Universtität Wuppertal, 2011.
[39] J. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, 1987, pp. 73–130.

[40] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, USA, 2nd ed., 2003.

[41] H. Schwarz, *Gesammelte mathematische Abhandlungen*, Vierteljahrschrift Naturforsch. Ges. Zürich, (1870), pp. 272–286.

[42] B. Sheikholeslami and R. Wohlert, *Improved continuum limit lattice action for QCD with Wilson fermions*, Nucl. Phys., B259 (1985), pp. 572–597.

[43] V. Simoncini and D. B. Szyld, *Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing.*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.

[44] 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.

[45] G. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford Applied Mathematics and Computing Science Series, Clarendon Press, 1985.

[46] 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 (2010), pp. 1715–1739.

[47] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid. With Guest Contributions by A. Brandt, P. Oswald, K. Stüben*, Academic Press, Orlando, FL, 2001.

[48] J. van den Eshof and G. L. Sleijpen, *Inexact Krylov Subspace Methods for Linear Systems.*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 125–153.

[49] J. C. Vink, *Multigrid inversion of staggered and Wilson fermion operators with $SU(2)$ gauge fields in two-dimensions*, Phys. Lett., B272 (1991), pp. 81–85.

[50] K. G. Wilson, *Quarks and strings on a lattice*, in 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, A. Zichichi, ed., vol. 321 of CLNS Reports, New York, 1977, Plenum Press.