Advanced Applications For Algebraic Multigrid Methods In Lattice QCD

Dissertation

Bergische Universität Wuppertal
Fakultät für Mathematik und Naturwissenschaften

eingereicht von
Artur Strebel, M. Sc.
zur Erlangung des Grades eines Doktors der Naturwissenschaften

Wuppertal, den 10. August, 2020

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-20200918-095033-4
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20200918-095033-4]

DOI: 10.25926/anzj-5192
[https://doi.org/10.25926/anzj-5192]

Acknowledgments

First of all, I would like to thank Karsten Kahl for his interesting lectures during my Masters studies which introduced me into the field of numerical linear algebra. I am also grateful to Andreas Frommer, who enabled me to do my research in his group and for creating a pleasant working athmosphere within this group. I greatly appreciate the support by Matthias Rottmann, especially during the beginning of my work.

Of course I also thank my (former) colleagues and friends, which by the time of this thesis are far away from being disjoint sets. I will always happily look back to the many great moments we had together and hope for many more to come.

Finally, I wish to thank my family for their continuous support and for always reminding me that there is a world outside of mathematics (and Wuppertal)!

Foreword

The work presented in this thesis is in parts based on the following publications:

- J. Brannick, A. Frommer, K. Kahl, B. Leder, M. Rottmann, and A. Strebel, *Multigrid preconditioning for the overlap operator in lattice QCD*, Numer. Math., 132 (2016), pp. 463–490
- A. Frommer, K. Kahl, F. Knechtli, M. Rottmann, A. Strebel, and I. Zwaan, *A multigrid accelerated eigensolver for the Hermitian Wilson-Dirac operator in lattice QCD*, (2020). arXiv:2004.08146

This work is funded by the Deutsche Forschungsgemeinschaft (DFG) the Transregional Collaborative Research Centre 55 (SFB-TR 55).

All numerical results shown in this thesis were computed at the Jülich Supercomputing Centre (JSC) using the super computers JUROPA, JURECA and JUWELS.

Contents

Chapter 1

Introduction

Quantum Chromodynamics (QCD) is a part of the standard model of particle physics and describes one of the four fundamental forces: the strong interactions between quarks and gluons within hadrons, e.g., within protons or neutrons in four-dimensional spacetime.

As it is impossible to extract information analytically from this continuous model, Lattice QCD has been introduced by Kenneth Wilson in 1974 [107]. It transports the continuous theory onto a four-dimensional lattice, where it can be solved numerically. Lattice QCD represents a non-perturbative approach and has shown to be in agreement with experimental observations, for example in [34] the authors determine the mass of the proton, neutron and other light hadrons using Lattice QCD. However lattice simulations are among the world's most demanding computational problems [8, 51], thus making the use of high-end supercomputing resources indispensable.

One of the main challenges lies in the repeated solution of an appropriate discretization of the Dirac equation, which describes the dynamics of the quarks and their interaction with gluons. Its most common discretization, i.e., the Wilson-Dirac operator, leads to large, ill-conditioned linear systems of equations when used in physically relevant simulations. Classical iterative schemes like Krylov subspace methods show a poor convergence rate for these problems [46]. When approaching the physical point, they suffer from "critical slowing down", where these methods become practically infeasible, even when using traditional preconditioning methods like odd-even preconditioning [29, 40], deflation [68] or domain decomposition [47, 67]. The ill-conditioning of the Wilson-Dirac operator in this case motivates the application of multigrid preconditioners whose convergence rate is ideally independent of the conditioning of the system, thus being favorable for this application.

Multigrid methods recently gained popularity within the Lattice QCD community leading to several algorithms, like the "AMG" method [82] implemented in the QOPQDP library [81] or the "Inexact Deflation" method [68] which can be interpreted as a 2-level multigrid method [46] and is part of the openQCD simulation package [65]. Recently Brower et al. also described a promising method to apply multigrid to the so-called staggered discretization [20]. This thesis is mostly focused on the DD-αAMG method [45, 46, 86] and aims at expanding the applicability of multigrid methods in Lattice QCD, beyond the solution of linear systems of equations involving the Wilson-Dirac operator.

We first show how to extend its use to another discretization, namely the Neuberger overlap operator. This operator respects an important physical property called chiral symmetry, which is not the case for the Wilson-Dirac operator. In practice its application is limited as it comes with a significantly larger computational cost due to the evaluation of an expensive matrix function for each matrix-vector multiplication. We propose a new preconditioner based on the idea of fictitious (or auxiliary) space preconditioning [77], which has been used for developing and analyzing multigrid preconditioners for various nonconforming finite element approximations of partial differential equations (PDEs); cf. [83, 110]. In this setting, choosing the Wilson-Dirac operator as an auxiliary space preconditioner is a viable choice, since both operators are defined on the same Hilbert space and can thus be constructed on the same finite-dimensional lattice. Numerical results show, that this preconditioning approach can substantially reduce the computational cost for solving linear systems with the overlap operator, reaching speed-ups of at least an order of magnitude over unpreconditioned solvers in realistic settings [16].

The second contribution of this thesis is motivated by a procedure known as "low-mode-averaging" [4, 7, 41], where small eigenvectors[1] of the Hermitian Wilson-Dirac operator are required to deflate the computation of the trace of its inverse. These eigenvalues are in the interior of the spectrum and as such known to be particularly expensive to compute. Preliminary results using a multigrid-accelerated Rayleigh quotient algorithm show that this technique can be beneficial for computing all-to-all propagators [5]. We propose a more advanced multigrid-accelerated eigensolver based on the generalized Davidson method and include several algorithmic adaptations specifically designed for the Hermitian Wilson-Dirac operator. This method outperforms standard algorithms like the implicitly restarted Arnoldi method [90] and shows an improved scaling behavior compared to state-of-the-art implementations of the generalized Davidson method, e.g., PRIMME [100].

The third contribution is a new approach for the setup procedure of the algebraic multigrid method. Within the setup the goal is to find so-called "test vectors" for

[1] Meaning the eigenvectors belonging to small (in magnitude) eigenvalues, cf. Section 1.2.

the construction of intergrid operators, which define the multigrid hierarchy. It has been observed that approximations to the singular vectors of the Wilson-Dirac operator, which correspond to eigenvectors of its Hermitian version, are a good choice [46] for these test vectors. In our current implementation these vectors are obtained using a simple inverse iteration algorithm, where the test vectors are treated separately, with a subsequent orthogonalization step to guarantee convergence to different singular vectors. While this method has a low computational effort it does not make efficient use of the information generated. We use our generalized Davidson algorithm to extract the optimal information from the subspace generated to approximate the smallest eigenvalues of the Hermitian Wilson-Dirac operator. During this process small generalized eigenvalue problems have to be solved, making this setup procedure computationally more expensive. However, this pays off, especially in scenarios where only a few setup steps are necessary, as for example in the Hybrid Monte Carlo algorithm.

1.1 Outline

The thesis is structured as follows. In order to keep it as self-contained as possible the basics of numerical linear algebra are reviewed in Chapter 2. We present a large class of iterative solvers named Krylov subspace methods, including the GMRES method, which is used recurrently throughout this thesis. Since multigrid methods are at the core of our developed applications, its fundamentals are reviewed here, including the DD-αAMG method. We then discuss eigenvalue problems in general and present classical algorithms. At the end of this chapter we introduce matrix functions, which are needed to define the Neuberger overlap operator.

Chapter 3 introduces the necessary basics of Quantum Chromodynamics. We define the continuous Dirac operator and derive its most commonly used discretization, the Wilson-Dirac operator. Relevant properties of this operator are discussed, as well as arising challenges and applications, which motivate the development of our methods in this thesis.

In Chapter 4 we describe the Neuberger overlap operator and elaborate on its advantage over the Wilson-Dirac operator. The new preconditioning method for this operator based on the DD-αAMG method is presented together with numerical experiments, which is also published in [16]

Chapter 5 introduces our specialized eigensolver framework for the Hermitian Wilson-Dirac operator. We first motivate our approach and then present our modifications to the generalized Davidson method, specifically designed for the Hermitian Wilson-Dirac operator and the DD-αAMG method. We conclude this section by comparing our method with two widely used software packages. This work is also published in [44].

We then apply in Chapter 6 the basic concepts of our developed eigensolver to the setup phase of the DD-αAMG method. We present multiple approaches and compare them numerically.

Chapter 7 summarizes the results achieved in this thesis and discusses possible topics for future research.

1.2 Notation and abbreviations

Throughout this thesis, we use the term "smallest/largest eigenvector/eigenpair" as a shorthand for "eigenvector/eigenpair belonging to the smallest/largest eigenvalue in absolute value". Unless stated otherwise all vector spaces and matrices are considered to be complex, and norms $\|\cdot\|$ without a subindex denote the Euclidean norm. An upper letter $\square^{(k)}$ within parenthesis denotes an iteration index. Apart from the exception of the γ matrices described below, all lower case letters denote vectors or scalars, while upper case letters are used for matrices.

The following notations and abbreviations are used throughout the thesis:

n	problem size (typically at least $\mathcal{O}(10^7)$)
m	subspace size (typically $m \ll n$)
$V = [v_1, \ldots, v_k]$	column matrix from a set of k vectors
$\mathcal{D}$	continuous Dirac operator
$D, D(m),\ D_W(m)$	(lattice) Wilson-Dirac operator (with mass m)
D_N	Neuberger overlap operator
m_0	mass shift
γ_i	generator matrices of an Clifford algebra, $i \in \{1, \ldots, 4\}$
γ_5	$\gamma_5 := \gamma_1\gamma_2\gamma_3\gamma_4$
Γ_5	lattice version of γ_5
Q	Hermitian Wilson-Dirac operator
λ	eigenvalue
Λ	set of eigenvalues
$X = [x_0, \cdot, x_n]$	set of eigenvectors
$\mathrm{spec}(A)$	spectrum of A
θ	Ritz value

Table 1.1: Notations and abbreviations

Chapter 2

Basics of numerical linear algebra

This chapter summarizes some of the basic concepts of numerical linear algebra as required for the understanding of this thesis. The reader is assumed to be familiar with undergraduate level linear algebra, analysis and general computer science as well as having a basic understanding of parallel programming.

We start by restating basic mathematical definitions, with the goal of establishing notation, symbols and basic methods used throughout this thesis. We introduce iterative methods for the solution of linear systems of equations and derive the popular GMRES method, as it is required extensively in many of the presented methods. A special focus is also put on multigrid methods, especially the DD-αAMG method, which is at the core of the new methods presented in this thesis. We define eigenvalue problems as basis for Chapters 5 and 6 and give a brief introduction into eigensolvers, explaining some standard methods for large sparse matrices. Finally we include a section about matrix functions, which are needed to define the Neuberger overlap operator in Chapter 4.

This chapter is in large parts based on books by Saad [89, 90], Trefethen and Bau [102], and Higham [58]. Proofs and further information regarding this chapter can also be found there.

2.1 Basic definitions

The main topics of this thesis are solving linear systems of equations, which can be represented by matrices, and computing eigenvalues of matrices. In both cases, special matrices with a certain structure have properties we can exploit for the development of efficient algorithms.

Definition 2.1 (special matrices).
We call a matrix $A \in \mathbb{C}^{n\times n}$

- *symmetric, if* $A = A^T$,
- *Hermitian, if* $A = A^H$,
- *unitary, if* $A^H A = I$,
- *normal, if* $A^H A = AA^H$,
- *a projection, if* $A^2 = A$,
- *sparse, if the number of non-zero entries per row is significantly smaller than* n *and independent of* n.

Remark 2.2. 1. *We can also define non-square "unitary" matrices:* $A \in \mathbb{C}^{n\times m}$ *with* $n \geq m$ *is unitary, if every column vector* a_i *has unit length and* $\langle a_i, a_j\rangle = 0$ *holds for all* $i \neq j$.

2. *If* A *is a projection then* $(I - A)$ *also defines a projection, as* $(I - A)^2 = I - 2A + A^2 = I - 2A + A = I - A$.

Eigenvalues and eigenvectors are central for Chapters 5 and 6, thus we give a definition including some of their relevant characteristics.

Definition 2.3 (eigenvalues).
Given a square matrix $A \in \mathbb{C}^{n\times n}$ *we call* $\lambda \in \mathbb{C}$ *an* eigenvalue *of* A *if and only if there exists a nonzero vector* $x \in \mathbb{C}^n$ *such that*

$$Ax = \lambda x. \tag{2.1}$$

Additional characteristics and terms related to eigenvalues:

- x *is called an* eigenvector (belonging to λ).
- *A pair* (λ, x) *of eigenvalue* λ *and its eigenvector* x *is called an* eigenpair.
- *The set of all eigenvalues of* A *is called* spectrum *of* A *and is denoted by* $\mathrm{spec}(A)$.
- *The* spectral radius of A *is defined as* $\rho(A) := \max_{\lambda\in\Lambda(A)}(|\lambda|)$.
- *Eigenvalues* λ_i *are the roots of the characteristic polynomial of* A, *i.e.,* $p_A(\lambda) := \det(A - \lambda I) = 0$.
- *The multiplicity* m_i *of an eigenvalue in* $p_A(\lambda)$ *is called* algebraic multiplicity *of* λ.

- *The* geometric multiplicity *is denoted by g_i and is the dimension of the eigenspace $(A - \lambda_i I)$.*

Definition 2.4 (eigenvalue decomposition)**.**
A square matrix $A \in \mathbb{C}^{n \times n}$ is called diagonalizable *if and only if $g_i = m_i$ for all $\lambda_i \in \Lambda$. We define in this case the* eigenvalue decomposition

$$A = XDX^{-1},$$

where each column x_i of X contains an eigenvector of A belonging to the eigenvalue $D_{i,i} = \lambda_i$ of the diagonal matrix D.

We can generalize the idea of an eigenvalue decomposition to (possibly) non-square matrices, which is known as *singular value decomposition.*

Definition 2.5 (singular value decomposition)**.**
Given a matrix $A \in \mathbb{C}^{m \times n}$ we can define the matrix decomposition

$$A = U\Sigma V^H,$$

where $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are unitary matrices and $\Sigma \in \mathbb{C}^{m \times n}$ is a diagonal matrix with non-negative entries $\sigma_{i,i}$. We call the column vectors u_i and v_i left *and* right singular vectors (of $\sigma_{i,i}$)*.*

With these definitions in place, we conclude this subsection with some remarks, which will prove useful later.

Remark 2.6.

1. *The (non-zero) singular values are the square roots of the (non-zero) eigenvalues of $A^H A$ or AA^H.*
2. *The left singular vectors u are the eigenvectors of AA^H while the right singular vectors v are the eigenvectors of $A^H A$.*
3. *The eigenvalues of a Hermitian matrix A are real, since if (λ, x) is an eigenpair of A then*
$$\begin{aligned} \langle Ax, x\rangle = \lambda &\Rightarrow \langle x, Ax\rangle = \lambda \\ &\Rightarrow \overline{\langle Ax, x\rangle} = \lambda \\ &\Rightarrow \langle Ax, x\rangle = \bar{\lambda}. \end{aligned}$$
4. *By definition subspaces $\mathcal{V} := \mathrm{span}(x_1, x_2, \ldots x_m)$ spanned by eigenvectors of A are A-invariant, i.e., $A\mathcal{V} = \mathcal{V}$.*
5. *If λ is an eigenvalue of A, then $(\lambda - \sigma)^{-1}$ is an eigenvalue of $(A - \sigma I)^{-1}$ for any $\sigma \in \mathbb{C}$.*

2.1.1 Conditioning

When numerical methods are implemented on a computer, we have to consider finite precision effects due to floating point representation. Irrational numbers as well as most rational numbers can only be stored approximately on a computer[1]. Furthermore standard operations like addition, subtraction, multiplication or division of floating point numbers introduce inaccuracies in their results. For example within the IEEE-754 standard [60], performing $c = a + b$ might yield the result $c = a$, if $a \gg b$. This leads to two key observations:

- Even in the best case, a solution obtained by numerical computation is in general an *approximation* to the exact solution.
- In the worst case, these finite precision effects accumulate and lead to an approximation which is far from the exact result.

In both cases, we are interested in a way to quantify the (in-)accuracy of a computed solution. We have to differentiate between two causes: One coming from a continuity argument of the underlying mathematical problem, which is referred to as *conditioning* of a problem. A mathematical problem, i.e., subtraction of two numbers or finding the smallest eigenvalue of a matrix, can be interpreted as a continuous function $f : X \to Y$ between normed vector spaces. Conditioning describes how the output y of a function f is affected by perturbations in the input x. It is independent of the algorithm which might implement this function on a computer. The other stems from the inability of exactly transporting this continuous problem to a computer, which is referred to as the *stability* of an algorithm. In this thesis we are mostly addressing conditioning, for further information regarding stability, see e.g., [102].

The definition of conditioning is tightly related to the notion of continuity. We interpret a problem as *well-conditioned* if small changes in the input only cause small changes in the output, and vice versa we interpret a problem as *ill-conditioned* if small changes in the input lead to big changes in the output.

Definition 2.7 (condition number).
Let $f : X \to Y$ be a problem and let $x \in X$. Let δx be some (infinitesimal) perturbation of x and $\delta f := f(x+\delta x)-f(x)$. We define the absolute condition number *$\hat{\kappa} = \hat{\kappa}(x)$ of f at x as*

$$\hat{\kappa} = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \frac{\|\delta f\|}{\|\delta x\|}. \tag{2.2}$$

[1]For example within the IEEE-754 standard for floating point arithmetic [60], 0.1 cannot be represented exactly.

The relative condition number $\kappa = \kappa(x)$ *is defined as*

$$\kappa = \sup_{\delta x} \left(\frac{\|\delta f\|}{\|f(x)\|} \Big/ \frac{\|\delta x\|}{\|x\|} \right). \tag{2.3}$$

If f is differentiable, then

$$\hat{\kappa} = \|J(x)\| \text{ and } \kappa = \frac{\|J(x)\|}{\|f(x)\|/\|x\|}, \tag{2.4}$$

where $J(x)$ is the Jacobian of f at x.

Well-conditioned problems have a small condition number, whereas ill-conditioned problems have a large one. The following examples show an analysis of some common operations based on this measure.

Example 2.8.

1. *Consider the simple problem of $f : \mathbb{C} \to \mathbb{C}$, $x \mapsto x/2$ with Jacobian $J = f' = 1/2$. Using Eq. (2.4) we find*

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|/\|x\|} = \frac{1/2}{(x/2)/x} = 1$$

   Thus this problem is well-conditioned.

2. *Another simple problem is $f : \mathbb{C}^2 \to \mathbb{C}$ with $f(x) = x_1 - x_2$. The Jacobian of f is*

$$J = \left[\frac{\partial f}{\partial x_1} \ \frac{\partial f}{\partial x_2} \right] = [1 \ -1],$$

   with $\|J\|_\infty = 1$. Using the ∞-norm, we obtain the condition number

$$\kappa = \frac{\|J(x)\|_\infty}{\|f(x)\|_\infty/\|x\|_\infty} = \frac{1}{|x_1 - x_2|/\max\{|x_1|, |x_2|\}}.$$

   Here, the condition number can get arbitrarily large for $x_1 \approx x_2$, making this seemingly harmless problem highly ill-conditioned. This phenomenon is also known as cancellation.

3. *In this thesis we are interested in the conditioning of linear systems of equations $x = A^{-1}y$, which in this context is equivalent to the conditioning of a matrix-vector multiplication $y = Ax$. With Eq. (2.3), we find*

$$\kappa = \sup_{\delta x} \left(\frac{\|A(x + \delta x) - Ax\|}{\|Ax\|} \Big/ \frac{\|\delta x\|}{\|x\|} \right) = \sup_{\delta x} \left(\frac{\|A\delta x\|}{\|\delta x\|} \Big/ \frac{\|Ax\|}{\|x\|} \right),$$

that is

$$\kappa = \|A\| \frac{\|x\|}{\|Ax\|}. \tag{2.5}$$

We can bound this to get a condition number independent of x:

$$\kappa \leq \|A\| \cdot \|A^{-1}\| \tag{2.6}$$

The product $\|A\| \cdot \|A^{-1}\|$ is referred to as the condition number of A *and is denoted by $\kappa(A)$.*

In practice, the condition number $\kappa(A)$ can be approximated without explicitly computing A^{-1}, cf. [54, 59]. It is an indicator of the achievable accuracy for many numerical methods involving this matrix.

We close this section with one important class of matrix decompositions, which is fundamental for developing more robust algorithms.

Definition 2.9 (QR decomposition)**.**
Given a matrix $A \in \mathbb{C}^{m\times n}$ with $m > n$, the QR decomposition *is given by*

$$A = QR,$$

where $Q \in \mathbb{C}^{m\times n}$ is a unitary matrix and $R \in \mathbb{C}^{n\times n}$ is a (upper) triangular matrix.

The matrix Q defines an *orthonormal* basis, i.e., a basis where $\langle q_i, q_j\rangle = 0$ if $i \neq j$ and $\|q_k\| = 1$ for all $k = 1, \ldots, n$ and describes a numerically favorable representation of a subspace. It can be obtained using the *(modified) Gram-Schmidt* algorithm, cf. Algorithm 2.1. This algorithm will generate a full set of orthonormal vectors $Q = [q_1, \ldots, q_n]$, which define a basis of the space spanned by a set of vectors $A = [a_1, \ldots, a_n]$, if the vectors $a_1, \ldots, a_n$ are linearly independent and is a numerically more stable version of the (regular) Gram-Schmidt algorithm, cf. [102].

2.2 Iterative methods for sparse linear systems of equations

For large parts of this thesis it will be required to solve linear systems of equations

$$Ax = b, \tag{2.7}$$

where $x \in \mathbb{C}^n$, $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$. One of the most prominent algorithm for the solution of Eq. (2.7) is *Gaussian elimination*, which falls under the category

Algorithm 2.1: The modified Gram-Schmidt procedure

input: set of vectors $a = [a_1, \dots, a_n]$
output: matrices $Q = [q_1, \dots, q_n]$ and R

$r_{1,1} \leftarrow \|a_1\|_2$
if $r_{1,1} = 0$ **then**
  stop
else
  $q_1 \leftarrow a_1 / r_{1,1}$
for $j = 2, \dots, n$
  define $\hat{q} := q_j$
  for $i = 1, \dots, j-1$
    $r_{i,j} \leftarrow \langle \hat{q}, q_i \rangle$
    $\hat{q} \leftarrow \hat{q} - r_{i,j} q_i$
  $r_{j,j} \leftarrow \|\hat{q}\|_2$
  if $r_{j,j} = 0$ **then**
    stop
  else
    $q_j \leftarrow \hat{q} / r_{j,j}$

of *direct solvers*. These methods perform manipulations on A in order to find the solution x. For Gaussian elimination, A is transformed into the identity matrix I, while simultaneously applying the same transformations to b. This way we implicitly compute $x = A^{-1}Ax = A^{-1}b$. However, direct solvers for sparse linear systems of equations typically have a computational complexity of $\mathcal{O}(n^2)$ and become prohibitively expensive for larger matrix dimensions. Thus, for large sparse matrices *iterative* solvers have been developed whose computational costs are typically dominated by matrix-vector products, which have a computational complexity of order $\mathcal{O}(n)$. These methods have the additional advantage that the matrix does not need to be stored in memory, but rather only requires a routine for the action Ax of the matrix A on a vector x. Also, iterative methods can be terminated early to give an approximate solution, whereas direct methods typically only yield a feasible solution at the last step of the algorithm.

The following definition gives a way to approximate the (unknown) error of an estimation $x^{(k)}$ to the solution x.

Definition 2.10 (residual equation)**.**
Defining the residual $r^{(k)} := b - Ax^{(k)}$ *and the* error $e^{(k)} := x - x^{(k)}$ *for a*

given $x^{(k)} \in \mathbb{C}^n$, *we formulate the* residual equation:

$$Ae^{(k)} = r^{(k)}. \tag{2.8}$$

Obviously, the problem of finding the error $e^{(k)}$ is equivalent to the problem of finding x. Also, since the error will typically not be available, the quality of an approximation $x^{(k)}$ can only be measured via the residual $r^{(k)}$. Looking at Eq. (2.8), we see that $\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \leq \|A^{-1}\| \cdot \|r^{(k)}\|$. This shows that if $\|A^{-1}\|$ is large, the error can still be large, despite the residual being small. Many iterative methods follow the idea of updating the iteration vector $x^{(k)}$ in every step starting from an (oftentimes random) initial vector $x^{(0)}$ by approximating $e^{(k)}$ with $\tilde{e}^{(k)}$ in some way and defining

$$x^{(k+1)} := x^{(k)} + \tilde{e}^{(k)}. \tag{2.9}$$

The following section presents three different kinds of iterative methods for solving Eq. (2.7). Simple *relaxation schemes*, such as Gauss-Seidel or SAP, *Krylov subspace methods* and *multigrid methods.*

2.2.1 Basic relaxation schemes and their block variants

Given a matrix A we can define a splitting

$$A := D - L - U \tag{2.10}$$

into its diagonal D, the strictly lower triangular part L and the strictly upper triangular part U.

With it, we can reformulate Eq. (2.7) as

$$x = D^{-1}(L + U)x + D^{-1}b. \tag{2.11}$$

Since x is unknown, we replace it with the approximation $x^{(k)}$ in the right hand side of this equation and find

$$D^{-1}(L + U)x^{(k)} + D^{-1}b = x^{(k)} + D^{-1}r^{(k)} =: x^{(k+1)}, \tag{2.12}$$

which is known as the *Jacobi method.* Looking at Equation (2.9) we can see that we use the term $\tilde{e}^{(k)} = D^{-1}r^{(k)}$ as an approximation to the error $e^{(k)}$. Alternatively, looking at Equation (2.8) we can interpret this method as A being approximated by its diagonal part D. Thus, this method intuitively works well if

A is diagonally dominant[2] but more generally it can be shown to converge linearly if the spectral radius $\rho(D^{-1}(L+U)) < 1$.

We can also use this splitting to define the *Gauss-Seidel method*. Using a similar derivation as with the Jacobi method, this method is given by the iteration formula

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b = x^{(k)} + (D-L)^{-1}r^{(k)}. \tag{2.13}$$

This time, A is approximated by $(D-L)$, which is intuitively a better approximation than D, and thus is expected to converge faster, but has the downside of being more expensive in a parallel environment [89]. Analogous to the Jacobi method, the Gauss-Seidel method converges linearly if $\rho((D-L)^{-1}U) < 1$. The proof of these convergence results is largely based on the *Banach fixed-point theorem*. For further information on splitting methods, see e.g., [89].

Instead of looking at one variable at a time, we can combine sets of variables into a *block (variable)*, which is computationally more favorable. We can formulate *block Jacobi* and *block Gauss-Seidel* in a straightforward way by defining a block decomposition of A and compatible block vectors x and b:

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,p} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p,1} & A_{p,2} & \cdots & A_{p,p} \end{bmatrix}, \; x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix},$$

where $A_{i,j} \in \mathbb{C}^{\ell_i \times \ell_j}$, $x_i \in \mathbb{C}^{\ell_i}$ and $b_i \in \mathbb{C}^{\ell_i}$ with $\sum_{i=1}^{p} \ell_i = n$.

We can define the matrix $I_i \in \mathbb{C}^{n \times \ell_i}$ with the identity on the i-th block row and zero everywhere else as the canonical injection from the i-th block column $A_{\cdot,i}$ into A and I_i^T as the trivial injection from the i-th block row $A_{i,\cdot}$ into A. This defines the block inverse $A_{i,i}^{-1} := (I_i^T A I_i)^{-1}$. Then we can apply the same formulas for both methods, where D, L and U are block diagonal respectively lower/upper block triangular matrices. These block Jacobi and Gauss-Seidel methods are also termed the *(additive)* and (multiplicative) Schwarz Alternating Procedure (SAP) [91, 95], which is a *domain decomposition* method for discretized partial differential equations. SAP is a crucial building block for the multigrid method we define in Section 2.3, so we give a description for this method in Algorithm 2.2 (block Jacobi or additive SAP) and Algorithm 2.3 (block Gauss-Seidel or multiplicative SAP).

[2] Meaning that (in absolute terms) all diagonal entries are larger than the sum of all other entries in their respective row.

Algorithm 2.2: Additive SAP (block Jacobi)

input: matrix A with blocks $A_{i,j}$, right hand side b, initial guess $x^{(0)}$
output: approximate solution x

for $k = 0, 1, 2, \ldots$
  $r^{(k)} \leftarrow b - Ax^{(k)}$
  foreach diagonal block $A_{i,i}$ **do**
    $x^{(k)} \leftarrow x^{(k)} + I_i A_{i,i}^{-1} I_i^T r^{(k)}$
  $x^{(k+1)} \leftarrow x^{(k)}$

Algorithm 2.3: Multiplicative SAP (block Gauss-Seidel)

input: matrix A with blocks $A_{i,j}$, right hand side b, initial guess $x^{(0)}$
output: approximate solution x

for $k = 0, 1, 2, \ldots$
  foreach diagonal block $A_{i,i}$ **do**
    $r^{(k)} \leftarrow b - Ax^{(k)}$
    $x^{(k)} \leftarrow x^{(k)} + I_i A_{i,i}^{-1} I_i^T r^{(k)}$
  $x^{(k+1)} \leftarrow x^{(k)}$

From Algorithm 2.3 it is apparent that multiplicative SAP has to be performed sequentially, since it requires the updated residual from the previous block solution for the next one, which is not the case for additive SAP (Algorithm 2.2). This makes additive SAP a natural choice in a parallel computing environment. However, we can decouple the sequential block solves of multiplicative SAP by using an appropriate coloring scheme leading to *red-black* SAP [46], which will be the method of choice for our multigrid method in Section 2.3.

2.2.2 Krylov subspace methods

When dealing with large sparse matrices, e.g., matrices coming from the discretization of PDEs, only the matrix-vector product $A \cdot x$ is generally available. This is due to the fact, that methods directly manipulating the matrix A, e.g., matrix decompositions, are too expensive in practice. *Krylov subspace methods* [89] only require matrix-vector multiplications and have modest storage requirements and are thus favorable in this scenario.

Definition 2.11 (Krylov subspace)**.**
*Let $A \in \mathbb{C}^{n\times n}$ and $r \in \mathbb{C}^n$. Then the m-*th Krylov subspace *is defined as*

$$\mathcal{K}_m(A, r) := \operatorname{span}\{r, Ar, A^2r, \ldots, A^{m-1}r\}.$$

If unambiguous we use the shorthand $\mathcal{K}_m$.

The dimension of a Krylov subspace grows by one each time a vector is added, until it reaches the *grade of r with respect to A* at which point it forms an invariant subspace of A. The grade of r (with respect to A) is defined as the degree of the minimal polynomial of r, i.e., the nonzero monic polynomial p of lowest degree such that $p(A)r = 0$. The following proposition states some basic properties of the m-th Krylov subspace.

Proposition 2.12.

1. *Let μ be the grade of r. Then $\mathcal{K}_\mu$ is invariant under A and $\mathcal{K}_m = \mathcal{K}_\mu$ for all $m \geq \mu$.*

2. *The Krylov subspace $\mathcal{K}_m$ is of dimension m if and only if the grade μ of r is not less than m, i.e.,*

$$\dim(\mathcal{K}_m) = m \Leftrightarrow \operatorname{grade}(r) \geq m.$$

Therefore,

$$\dim(\mathcal{K}_m) = \min\{m, \operatorname{grade}(r)\}.$$

Proof. See [89] □

Since the series of vectors $A^{k-1}r$ for $k = 1, 2, ...$ (usually) converges to the largest eigenvector of A (see Thm. 2.18), the set $\{r, Ar, A^2r, \ldots, A^{m-1}r\}$ forms a highly ill-conditioned basis. A more numerically favorable basis can be constructed using *Arnoldi's method* [2], see Algorithm 2.4. It generates an orthonormal basis V_m of the m-th Krylov subspace in an iterative fashion similar to the modified Gram-Schmidt procedure (Algorithm 2.1).

It has a wide range of applications, including solving linear systems of equations, approximating eigenpairs (see Section 2.4) and also evaluating matrix functions (Section 2.5), which are all necessary for this work.

By closely following Algorithm 2.4 we can derive a matrix formulation of this procedure, called the *Arnoldi relation*:

Algorithm 2.4: Arnoldi's method

input: matrix A, initial vector r
output: orthonormal basis V_m of $\mathcal{K}_m$

$h_{1,1} \leftarrow \|r\|_2$
$v_1 \leftarrow r/h_{1,1}$
for $j = 1, \ldots, m$
  $w_j \leftarrow Av_j$
  for $i = 1, \ldots, j$
    $h_{i,j} \leftarrow \langle w_j, v_i \rangle$
    $w_j \leftarrow w_j - h_{i,j} v_i$
  $h_{j+1,j} \leftarrow \|w_j\|_2$
  if $h_{j+1,j} = 0$ **then**
    break
  else
    $v_{j+1} \leftarrow w_j / h_{j+1,j}$

Theorem 2.13.
Let $V_m = [v_1, \ldots, v_m]$ be the $n \times m$ column vector matrix from Arnoldi's method and $H_{m+1,m}$ the according upper Hessenberg matrix, gathering the orthogonalization coefficients $h_{i,j}$ from Algorithm 2.4. Then the following equalities hold.

$$AV_m = V_m H_{m,m} + h_{m+1,m} V_{m+1} e_m^T \tag{2.14}$$
$$= V_{m+1} H_{m+1,m} \tag{2.15}$$
$$V_m^H A V_m = H_{m,m} \tag{2.16}$$

With Arnoldi's relation, we can define the **G**eneralized **M**inimal **Res**idual (GMRES) method, which is used to solve linear systems of equations $Ax = b$.

Any vector x in $x_0 + \mathcal{K}_m$ can be written as

$$x = x_0 + V_m y,$$

where $y \in \mathbb{R}^m$. Then the residual norm $\|r\| = \|b - Ax\|$ can be transformed to

$$\begin{aligned} \|r\| &= \|b - A(x_0 + V_m y)\| \\ &= \|r_0 - AV_m y\| \\ &= \|\beta v_1 - V_{m+1} H_{m+1,m} y\| \\ &= \|V_{m+1} \left(\beta e_1 - H_{m+1,m} y\right)\| \\ &= \|\beta e_1 - H_{m+1,m} y\|, \end{aligned}$$

where $\beta := \|r_0\|$, cf. Algorithm 2.4. As the name suggests, GMRES extracts an approximation x_m by minimizing the residual norm $\|r\| = \|\beta e_1 - H_{m+1,m}y\|$, i.e.

$$
\begin{aligned}
x_m &= x_0 + V_m y_m, \text{ with} \\
y_m &= \arg\min_y \|\beta e_1 - H_{m+1,m}y\|,
\end{aligned}
$$

which requires the solution of an $(m+1) \times m$ least-squares problem. If implemented correctly $\|r\|$ can be cheaply updated in every iteration, see [89]. When $\|r\|$ passes a given threshold the iteration stops and computes the approximate solution x_m.

The basic GMRES algorithm is summarized in Algorithm 2.5.

Algorithm 2.5: GMRES

input: matrix A, right hand side b, initial guess x_0, maximum number of iterations m, residual tolerance tol

output: approximate solution x_m

1 $r_0 = b - Ax_0$, $\beta = \|r_0\|$, $v_1 = r_0/\beta$
2 **for** $j = 1, \ldots, m$
3     $w_j \leftarrow Av_j$
4   **for** $i = 1, \ldots, j$
5         $h_{i,j} \leftarrow \langle w_j, v_i \rangle$
6         $w_j \leftarrow w_j - h_{i,j}v_i$
7     $h_{j+1,j} \leftarrow \|w_j\|_2$
8   Update $\|r\|$
9   **if** $\|r\| < tol$ **then**
10         $m \leftarrow j$, break
11     $v_{j+1} \leftarrow w_j / h_{j+1,j}$
12 Compute y_m
13 $x_m \leftarrow x_0 + V_m y_m$

As a consequence of Proposition 2.12, GMRES can break down at line 11 if $h_{j+1,j} = 0$, which happens when the dimension of the Krylov subspace exceeds the grade μ of r_0. This is considered a *lucky breakdown*, since in this case the exact solution is already contained within the search space and can be extracted using lines 12 and 13.

2.2.3 Restarting

It can be shown that GMRES converges after at most grade(r_0) steps, at which point a lucky breakdown occurs.[3] However, the growing memory requirements for storing the search space and the computational cost of orthogonalization, which grows quadratically with the size of the search space, only allow for a limited amount of iterations for larger matrices.

One solution to this problem is given by *restarting* the iterative method after $k \ll n$ steps. After these k steps, the current approximation $x_{old}^{(k)}$ and its residual $r_{old}^{(k)}$ are computed and the generated subspace is discarded. Afterwards, the iterative method starts anew using the residual equation (2.8) to obtain a new approximation $x_{new}^{(0)} \leftarrow x_{old}^{(k)} + \tilde{e}_{old}^{(k)}$. We call a full cycle of k steps a *restart cycle*. For GMRES, we denote restarted GMRES by GMRES(k), where k is the *restart length*.

In theory, restarting can lead to stagnation if the same subspace is built in the new restart cycle, which would yield no improvement over the previous one. Example 2.14 illustrates a simple case of this problem.

Example 2.14.
Consider GMRES for the 4×4 linear system of equations $Ax = b$ with

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \text{and } x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Using the starting vector $x^{(0)}$ the exact solution $x = (0,0,0,1)^T$ is found after $n = 4$ steps (if the initialization is also counted as one step). Restarting after $k < n$ steps will yield no improvement since $x^{(k)} = x^{(0)}$ in this case, thus GMRES(k) does not converge for this matrix.

In practice, restarting significantly reduces orthogonalization cost if the restart length is chosen properly. This comes at the expense of an increased number of iterations, which amounts to more matrix-vector multiplications. With the orthogonalization cost oftentimes being a substantial amount of the overall computational effort restarting also typically reduces computational cost, in addition to bounding the memory requirements. We provide an example of GMRES for a matrix of dimension $n = 53{,}248$ (Configuration 2 from Table 3.1) with varying restart lengths in Figure 2.1. Here we observe, that even though GMRES(50)

[3] Similar statements hold for other Krylov subspace methods, which fulfill some sort of optimality criterion.

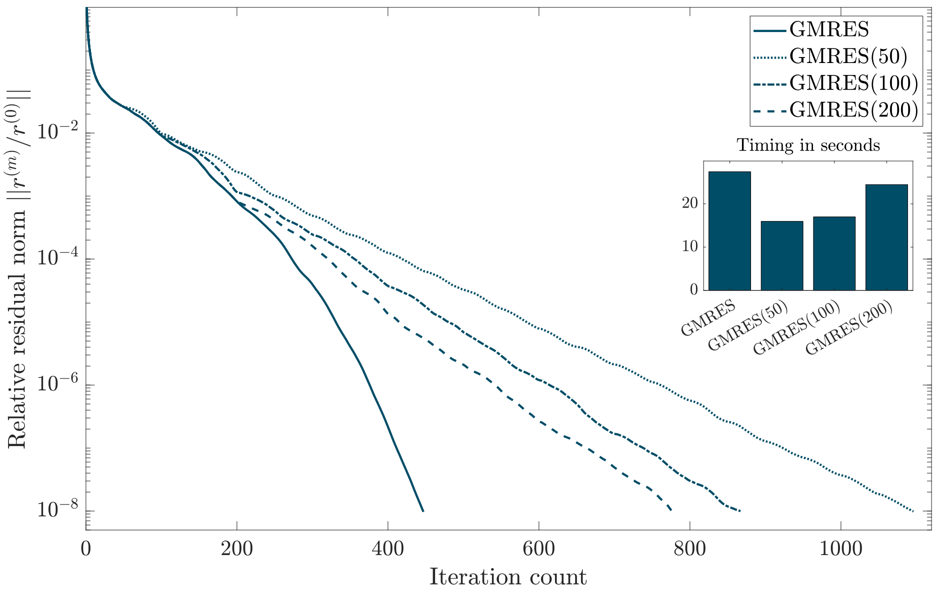


Figure 2.1: Convergence plot of GMRES(k) for a matrix of dimension $n = 53{,}248$ for different values of k. This plot was generated on a local workstation using MATLAB [71].

requires more than twice as many iterations, its execution time is nearly half of the execution time of plain GMRES. The "optimal" restart length depends on the hardware architecture and cannot be deduced a-priori, such that is has to be obtained by parameter studies.

Restarting can be implemented in various iterative methods, including eigensolvers, which are discussed in Section 2.4.

2.2.4 Preconditioning

The convergence speed of iterative methods oftentimes depends on the condition number $\kappa(A)$ of the system matrix A. The idea of *preconditioning* is to reduce the condition number by transforming the problem to an equivalent one with a smaller condition number. In general we are interested in a matrix M which is in some way close to A^{-1}, such that

$$1 = \kappa(I) \approx \kappa(MA) \ll \kappa(A).$$

We define *left preconditioning* via

$$Ax = b \Leftrightarrow MAx = Mb,$$

and *right preconditioning* via

$$Ax = b \Leftrightarrow AMy = b,$$

where $x = My$. As a consequence, in preconditioned methods every matrix vector multiplication also requires the application of the preconditioner. Thus from a practical point of view the application of M needs to be significantly cheaper compared to the solution of linear systems with A, since they are applied in every iteration, but should still be "close enough" to A^{-1} to have a notable impact on the condition number. Typically M is realized by some method, which provides a low precision solution of the original system.

In the following we discuss two different preconditioners for GMRES and show their impact on the convergence speed of linear systems of equations involving the Dirac operator, which will be discussed in Chapter 3. First note that applying non-stationary iterative schemes as preconditioners for GMRES requires to modify GMRES in order to maintain its optimality criterion of minimizing the norm of the residual in every step. This modified method is known as *flexible* GMRES (FGMRES) [88] and is described in Algorithm 2.6.

Algorithm 2.6: flexible GMRES

input: Matrix A, preconditioner M, right hand side b, initial guess x_0, maximum number of iterations m, residual tolerance tol
output: Approximate solution x_m to $Ax = b$

$r_0 = b - Ax_0$, $\beta = \|r_0\|$, $v_1 = r_0/\beta$
for $j = 1, \ldots, m$
  $z_j \leftarrow Mv_j$
  $w_j \leftarrow Az_j$
  for $i = 1, \ldots, j$
    $h_{i,j} \leftarrow \langle w_j, v_i \rangle$
    $w_j \leftarrow w_j - h_{i,j} v_i$
  $h_{j+1,j} \leftarrow \|w_j\|_2$
  Update $\|r\|$
  if $\|r\| < tol$ **then**
    $m \leftarrow j$, break
  $v_{j+1} \leftarrow w_j / h_{j+1,j}$
Compute y_m
$x_m \leftarrow x_0 + MV_m y_m$

Simple and cheap relaxation schemes like SAP are well suited as preconditioners as well as *polynomial* preconditioning. In the latter, a matrix vector multiplication is replaced by a polynomial $p(A) \approx A^{-1}$. A special case is *recursive* preconditioning, where $p(A)$ can be constructed by another application of the iterative method itself, e.g., applying a small amount of GMRES steps as a preconditioner for FGMRES, which is known as *GMRESR* [88].

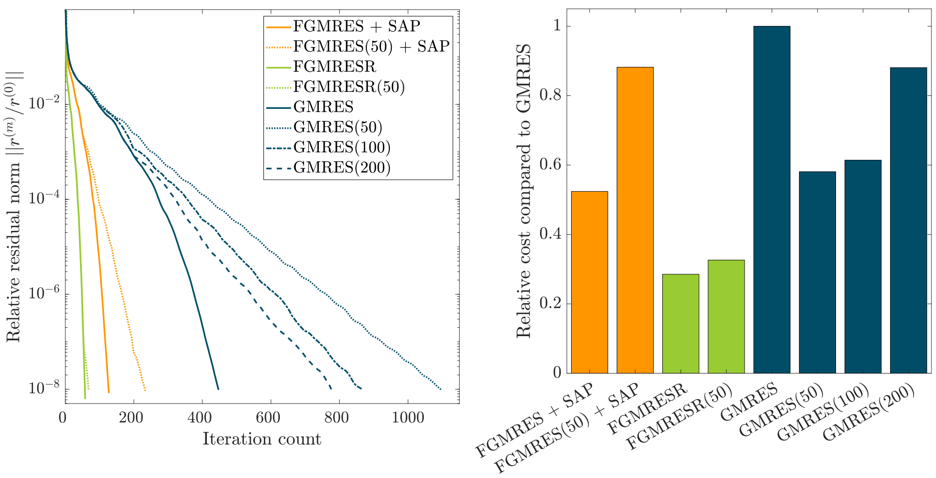


Figure 2.2: Extending Figure 2.1 with a study of different preconditioning methods. *Left:* Convergence plots of (preconditioned) GMRES(k). *Right:* Execution time relative to pure GMRES.

In addition to reducing the number of iterations needed, preconditioning also allows for the bulk of the computation to be performed in IEEE single precision, since only a low precision solution is required, which leads to faster execution times for all operations.

Figure 2.2 compares SAP preconditioned FGMRES ("FGMRES+SAP") and GMRESR with GMRES(k). In this plot, FGMRES+SAP employs one step of SAP with block size 4^4 and exact block solves, whereas GMRESR applies ten steps of GMRES as a preconditioner. Compared to plain GMRES both preconditioners significantly improve convergence while also reducing the overall computational cost. In the right plot we see that combining preconditioning with restarting does not pay off in every case. Both FGMRES + SAP and GMRES(50) reduce compute time by a factor of two, whereas FGMRES(50) + SAP is nearly as expensive as GMRES itself, which is due to the orthogonalization cost not being dominant for a small amount of iterations, thus restarting only slows down convergence. This situation changes when the matrix size or the condition number of the problem increases, which increases the iteration count and with it the orthogonalization cost.

Several other preconditioning methods, e.g., multigrid methods or auxiliary space preconditioning will be discussed in the following section and Section 4.

2.3 Multigrid methods

Multigrid methods [53, 87] arose in the last decades and became one of the most popular schemes for solving large sparse linear systems of equations, which are typically derived from the discretization of partial differential equations. Since most PDEs cannot be solved directly, they have to be formulated on a (finite) subset of the original domain called a *(finite) lattice* or *grid*, and then solved numerically.

In this thesis we only consider *cubic* lattices, which are also called *regular* grids:

Definition 2.15 (cubic lattice).
A (cubic) lattice $\mathcal{L} \subset \mathbb{R}^n$ with lattice spacing $a \in \mathbb{R} \setminus \{0\}$ is a discrete subset of $\mathbb{R}^n$, where for any two points $x, y \in \mathcal{L}$ there exists a vector $\mu \in \mathbb{Z}^n$ with

$$y = x + a \cdot \mu.$$

Given two lattices $\mathcal{L}$ and $\mathcal{L}'$, we call $\mathcal{L}'$ a coarser *lattice of the* fine *lattice $\mathcal{L}$, if $\mathcal{L}' \subset \mathcal{L}$.*

The basic idea of multigrid methods is to obtain useful information for the original fine grid from a coarser grid, i.e., a grid with less degrees of freedom, which is cheaper to solve. Multigrid methods typically exhibit a good scaling behavior with respect to problem size and are insensitive to ill-conditioning.

The main concept of multigrid methods is the utilization of two complementary preconditioning techniques: the smoother and the coarse grid correction [18, 53, 87, 103]. Starting from the original grid, the initial error is preconditioned by the smoother, followed by a coarse grid correction step, where the smoothed residual is projected to a coarser grid. On this coarser grid, the system is either solved directly, or the multigrid method is applied in a recursive fashion to further reduce the dimension of the problem.

2.3.1 Smoother

The name of this step comes from the geometrical interpretation of its main goal, i.e., *smoothing* the error. Looking at the error e of an iterate, we can decompose it into a smooth and oscillatory part in general connected to the low and high end of the spectrum respectively, i.e., $e = e_{low} + e_{high}$. These components can be represented as a linear combination of the eigenvectors of the matrix A, where e_{low} corresponds to the contribution of small eigenvectors and e_{high} corresponds to the contribution of large eigenvectors. The purpose of the smoother is to efficiently reduce e_{high}. As long as the smoother achieves this goal, the method used does not need to be convergent. We can quantify the error reduction in practice by defining the error propagator

$$E_s := (I - MA), \tag{2.17}$$

where M defines the smoother. In the upper part of Figure 2.3 we evaluate $\|E_s v_i\|$ for every normalized eigenvector v_i of a matrix of size 3072 (Configuration 1 from Table 3.1) to see the effect of a smoother on spec(D). After applying one iteration of SAP we observe the desired effect: error components belonging to larger eigenvectors are efficiently reduced, while error components belonging to the smallest eigenvectors are still largely preserved and for one eigenvector even slightly amplified. The lower part of Figure 2.3 gives a graphical representation of the error after several steps of the Gauss-Seidel method on Laplace's equation for a random initial guess. Starting on the left with a highly oscillating error we achieve a smooth surface after 20 iterations of Gauss-Seidel. This smooth error property will be crucial for the next section.

The smoother can be realized by a variety of methods, e.g., a few steps of GMRES, SAP or any other simple relaxation scheme. Looking back at Figure 2.2, we can interpret the discussed preconditioners as simple smoothing schemes, which will also be part of our multigrid implementation.

2.3.2 Coarse grid correction

In the previous section we discussed how simple relaxation schemes can be used as a preconditioner to smooth the error. Since these methods are not able to efficiently reduce the smooth error e_{low}, the smoother has to be complemented with a *coarse grid correction* step, where the problem is projected to a lower dimensional grid, solved on this smaller grid, and then transported back to the original grid to yield an error update. In this thesis, we are mostly interested in a Petrov-Galerkin approach for the coarse grid correction, in which case the error propagator can be written as

$$E_c := (I - P\underbrace{(RAP)}_{:=A_c}{}^{-1}RA), \tag{2.18}$$

where R is the *restriction operator* and P the *interpolation (or prolongation)* operator. These intergrid operators transport vectors from the fine grid to the coarse grid and vice versa. The definition of these operators can be classified into two categories, which also gives the name to the corresponding multigrid method: *geometric* and *algebraic* coarsening.

Geometric coarsening builds the coarse space by choosing a sublattice of the original lattice for the restriction operator R, e.g., every other grid point for

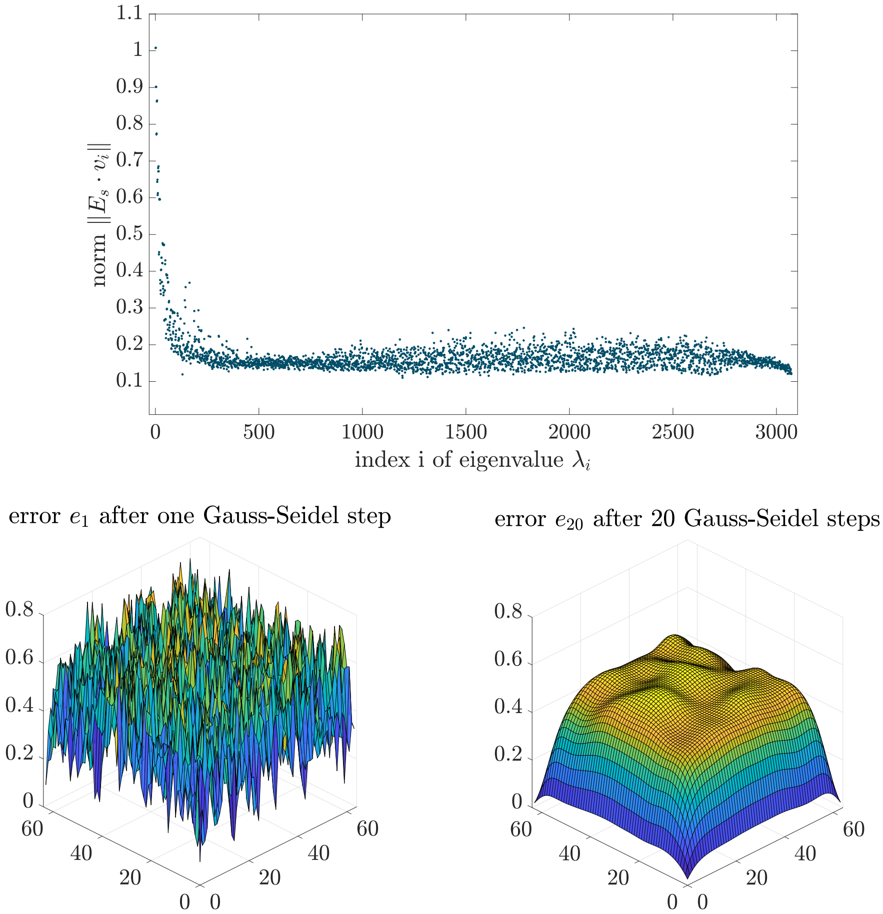


Figure 2.3: *Top:* Error reduction after one iteration of SAP for Configuration 1 from Table 3.1. *Bottom:* Error e_k of the Gauss-Seidel method to Laplace's equation with random initial guess $x^{(0)}$ and $k = 1$ iterations for the left plot and $k = 20$ iterations for the right plot.

chosen directions. The interpolation operator P can then for example be chosen such that it represents a linear (or quadratic) interpolation from the coarse grid points back to the fine grid points in between, see [103]. Geometric coarsening can be used when the underlying problem has a geometric interpretation in some way, which is oftentimes the case when discretizing PDEs.

On the other hand, algebraic coarsening is not dependent on the origin of the matrix A and thus does not require assumptions about its structure. Instead the classical approaches only rely on the entries of the matrix to construct the intergrid operators. In this thesis, we are more interested in *adaptive* algebraic multigrid methods which prescribe the smoother and then adaptively construct the intergrid operators in a *setup phase* such that the coarse grid correction step is complementary to the smoother. In most cases the setup approximates small eigenvectors, since typical smoothing methods only reduce the error components belonging to large eigenvectors. In [15] the authors describe *optimal interpolation*, which explicitly includes the smoother into the construction of the coarse grid, to get an “optimal” convergence rate of a two-level multigrid method.

Figure 2.4 shows illustrations similar to Figure 2.3. In the upper part we evaluate $\|E_c v_i\|$, this time applying one step of an algebraically constructed coarse grid correction to each normalized eigenvector v_i of a matrix of size 3072 (Configuration 1 from Table 3.1). For the intergrid operators we used a fairly extensive setup procedure to approximate the 20 smallest eigenvectors.[4] We observe that the error components belonging to small eigenvectors are efficiently reduced, which shows the complementary of the coarse grid correction step to the smoother.

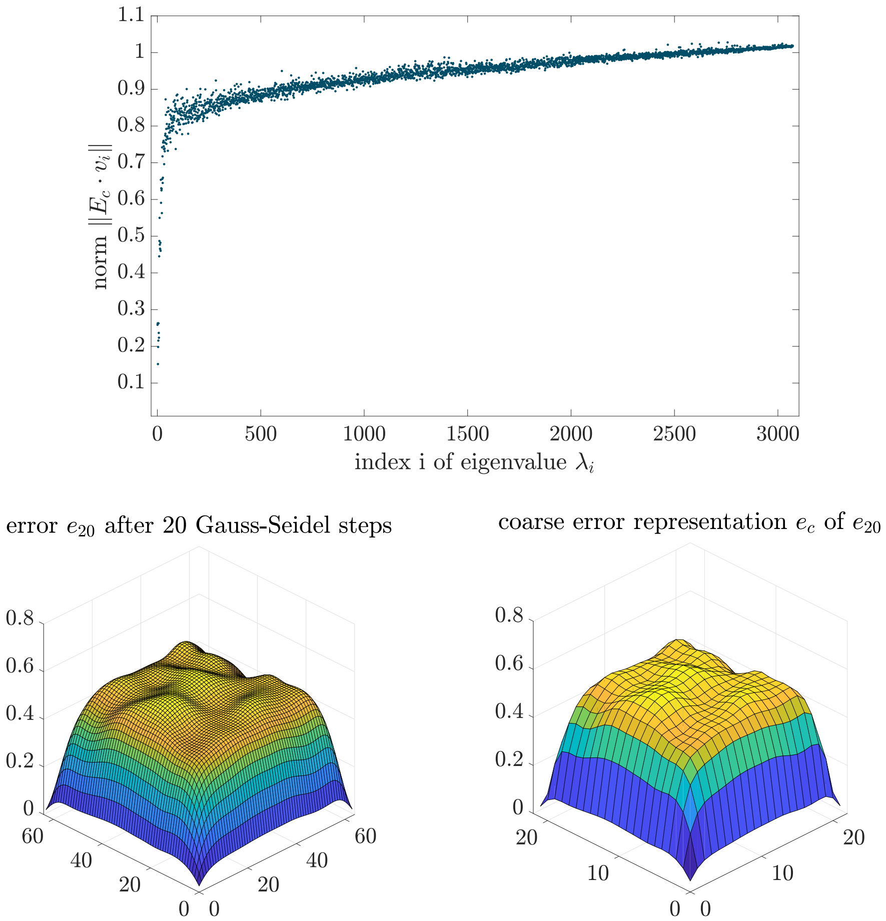


Figure 2.4: *Top:* Error reduction after one coarse grid correction step for all eigenvectors of Configuration 1 from Table 3.1. *Bottom:* Error on the fine grid after 20 steps of the Gauss-Seidel method to Laplace’s equation (*left*) and its representation on a coarser grid using full coarsening with coarsening factor 3 (*right*).

[4]The generation of the intergrid operators is discussed more thoroughly in Section 2.3.3.

The lower part of Figure 2.4 visualizes the coarsening process for a geometric multigrid approach, but conceptually also holds true for algebraic multigrid. The initial error has a highly oscillatory behavior *(left)*, which, after sufficient smoothing steps, resembles a smooth surface, where all oscillations are removed *(right)*. Further smoothing steps reduce the error inefficiently, thus one step of the coarse grid correction is required. In this example we take every third grid point for every direction ("full coarsening") of the fine grid and define them as the coarse grid points, interpolating linearly between them to approximate the fine error. The overall behavior of the error is still preserved, but now consists of 3^2 less degrees of freedom. On this smaller scale the residual equation can be solved cheaply and the resulting error update is transferred back to the fine grid. In the case where the coarse system is still too large, the multigrid approach can be applied recursively on this system.

2.3.3 Multigrid in lattice QCD

In this section we give an example of a multigrid method in the form of the *DD-αAMG* method (**D**omain **D**ecomposition aggregation-based $\boldsymbol{\alpha}$daptive **A**lgebraic **M**ulti**G**rid method), which is used to solve linear systems of equations for the Dirac operator, see Chapter 3. The new algorithms presented in this thesis are implemented within this framework. A full description of DD–αAMG and its implementation, including numerical studies and further references can be found in the PhD thesis by M. Rottmann [85] and in parts in [45, 46], thus we focus on restating the main aspects relevant for our algorithms.

The first step of this method is the setup procedure, in which we generate the eigenvector information for the intergrid operators. It consists of two phases and is mainly based on the inverse iteration algorithm, which is described in Section 2.4. In the initial phase, it performs several steps of the smoother on a set of n_{tv} random test vectors $W := [w_1, \dots, w_{n_{tv}}]$. After this, the set of vectors W is subdivided into *aggregates* $\mathcal{A}_i$, which resemble a block splitting on the lattice, as described in the following.

Analogously to a lattice block, an aggregate $\mathcal{A}_i$ is comprised of a set of lattice points, except that the variables on one lattice site can be part of different aggregates, which induces a block structure of the matrix W, cf. Figure 2.5. We then perform a QR algorithm to orthogonalize each aggregate $\mathcal{A}_i$ locally and define a "block diagonal" interpolation operator $P := \mathrm{diag}(W)$, where each diagonal block contains a (non-square) aggregate $\mathcal{A}_i$.

For the restriction operator, we set $R = P^H$ in order to preserve the structure of the problem on the coarse grid.

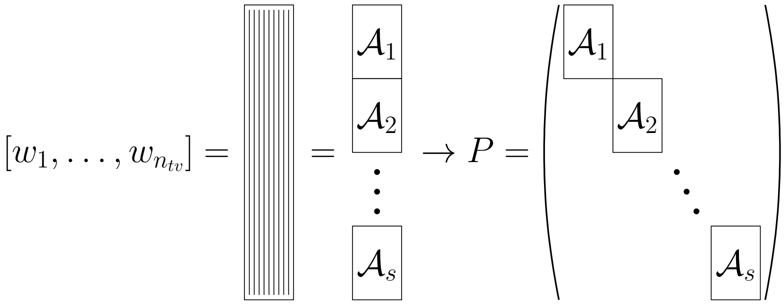


Figure 2.5: The construction of the interpolation operator P.

The iterative phase utilizes the initial multigrid hierarchy to further improve the eigenvectors. Instead of applying the smoother to the test vectors, we now perform a full AMG cycle (smoother + coarse grid correction) on each eigenvector. Afterwards, we update the multigrid hierarchy and continue on the next coarser level. Algorithm 2.7 briefly summarizes the setup procedure, skipping implementation details and algorithmic fine tuning, see [85] for further information.

Algorithm 2.7: Bootstrap AMG setup

input: smoothing method M, number of iterative phases k
output: Intergrid operators $P = R^H$ and coarse grid operator D_c

```
// Initial phase
Define set of random test vectors W = [w_1, ..., w_{n_tv}]
for j = 1, ..., n_tv
    w_j ← M w_j
construct P and D_c from W
perform initial phase for D_c
// Iterative phase
for i = 0, ..., k
    for j = 0, ..., n_tv
        w_j ← AMG(w_j)
    update P and D_c
    perform iterative phase for D_c
```

The setup procedure adaptively defines the coarse system based on a subspace which could not be efficiently treated by the smoothing method of our choice. DD-αAMG implements two different options for the smoothing method: SAP and GMRES. The former is implemented as an additive, multiplicative, two color or 16 color variant, where the block solves are performed by a few steps of the minimal residual (MR) algorithm [89], which is mathematically equivalent to GMRES(1). Numerical studies from [85] suggest that the two color SAP approach is in general the most suitable smoothing method for the Dirac operator.

In practice the convergence for the pure multigrid method cannot be guaranteed for the Dirac operator due to its structural properties, e.g., being non-normal and not necessarily positive definite. For this reason it is beneficial to use the multigrid method as a (right) preconditioner for a restarted FGMRES method. This improves the convergence of the overall method and thus speeds up the overall method at minimal cost. In summary, the three ingredients of DD-αAMG are: the FGMRES method as a wrapper, two-color SAP as the smoothing method and an adaptively determined aggregation based coarse grid correction.

Figure 2.6 illustrates the effectiveness of DD-αAMG over conventional Krylov subspace methods in a high performance computing environment. We compare a two, three and four level DD-αAMG method with an optimized Krylov subspace method, which is a mixed precision, odd-even preconditioned BiCGStab algorithm, see [85]. The left plot shows the time for a single solve of a linear system with the Dirac operator, which is dependent on a mass shift $m_0 \in \mathbb{R}$. This shift is an indicator for the ill-conditioning of the system—the smaller m_0 the more ill-conditioned the matrix. The multigrid method outperforms the Krylov subspace method by more than two orders of magnitude for physically relevant mass shifts. We also see that depending on the conditioning, it is beneficial to use additional multigrid levels. However, there is a trade-off: Using too many levels eventually slows down the application, as the method will be dominated by communication and idling processes. The right plot shows the same situation, but includes the time spent for the multigrid setup phase. Due to the bootstrap approach for the setup, the overall cost of the multigrid method is dominated by the setup cost, but is still clearly favorable over BiCGStab by one order of magnitude. Consequently, in scenarios where multiple solves are required multigrid methods are even more advantageous.

2.4 Eigenvalue problems

Eigenvalues and eigenvectors can be found in nearly all disciplines of natural science and applied mathematics. In a mathematics context eigenvectors are often used for *deflation*, a method which can be applied to improve the condition number of a problem. Engineers determine eigenfrequencies of structures like skyscrapers, bridges or wings of airplanes to make them resistant to vibrations. In quantum physics or chemistry, eigenvalues are used to determine energy levels of particles, e.g., electrons in an atom.

Our interest in the eigenvalue problem in this thesis is twofold:

- Eigenpairs can be used to directly compute physical observables [9, 30, 39, 41, 49, 76] or as a tool for noise reduction in stochastically estimated

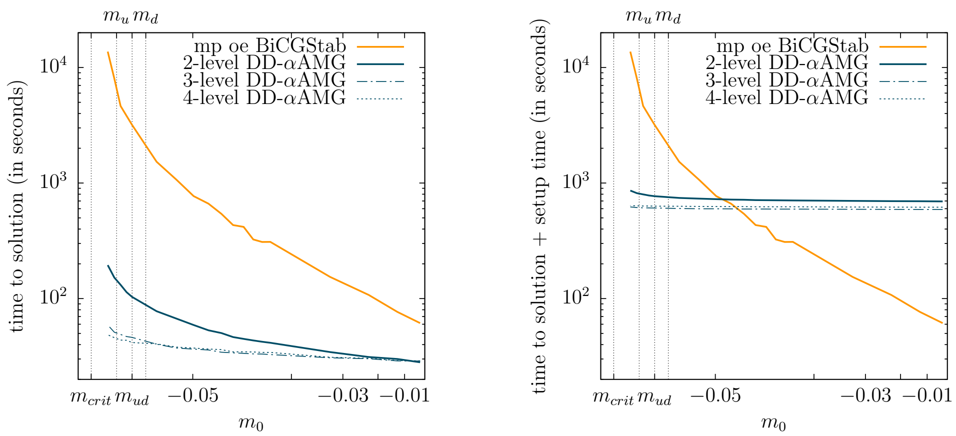


Figure 2.6: Comparing computational cost for solving linear systems with configuration 9 (see Table 3.1) using DD-αAMG and a Krylov subspace method. The left plot reports on timings for the solve only, whereas the right plot includes the multigrid setup time. Both plots were generated on the JUROPA high performance computer from the Jülich Supercomputing Centre.

quantities like disconnected fermion loops [4]. In this case we need accurate approximations and potentially a large amount of eigenvectors, such that scalability is a critical factor.

- Approximations to the eigenvectors of the (Hermitian) Dirac operator are required in order to define the interpolation and restriction operators in the setup phase of our multigrid method, cf. Chapter 2.3 or [46]. Since a rough approximation is sufficient in this scenario, a special focus is put on how to acquire a few of them as efficiently as possible.

In the following we review basic algorithms for the computation of eigenpairs. We start with simple vector iterations, where one vector is manipulated until it converges towards an eigenvector. Afterwards we introduce subspace accelerated eigensolver methods, which combine the information from all previous iterations to obtain the optimal approximations within the subspace.

2.4.1 Eigensolvers based on vector iteration

In this section we review eigensolvers, which extract one eigenpair (λ, x) by iterating on one pair $(\lambda^{(k)}, x^{(k)})$ until it converges. Note that λ and x are both unknown and knowledge of either one of them conceptually yields a simple way to obtain the other. Given λ, it is sufficient to solve the homogeneous equation

$(A - \lambda I)x = 0$ to obtain x.[5] Conversely, if x is given, we can compute the *Rayleigh quotient* of x to obtain λ, which we define in the following.

Definition 2.16 (Rayleigh quotient).
Given a matrix $A \in \mathbb{C}^{n\times n}$, the Rayleigh quotient *of a vector $x \in \mathbb{C}^n$ is defined by*

$$r(x) = \frac{x^H A x}{x^H x}. \tag{2.19}$$

If x is an eigenvector of A then $r(x) = \frac{x^H A x}{x^H x} = \frac{x^H \lambda x}{x^H x} = \lambda \frac{x^H x}{x^H x} = \lambda$. We can show that for a Hermitian matrix A and a vector x that is not necessarily an eigenvector, $r(x)$ is a *quadratically accurate* estimate of an eigenvalue.

Theorem 2.17.
Given a Hermitian matrix $A \in \mathbb{C}^{n\times n}$, an eigenvector $x_J \in \mathbb{C}^n$ of A and a vector $t \in \mathbb{C}^n$ we have:

$$r(t) - r(x_J) = \mathcal{O}(\|t - x_J\|^2), \text{ as } t \to x_J \tag{2.20}$$

Proof. Since $t = \sum_{j=1}^n a_j x_j$ is a linear combination of the eigenvectors $x_1, \dots, x_n$ of A we have that the quotient

$$r(t) = \sum_{j=1}^n a_j^2 \lambda_j / \sum_{j=1}^n a_j^2$$

is a weighted mean of the eigenvalues of A, with weights equal to the squares of the coordinates of t in the eigenvector basis. If t is sufficiently close to x_J then $a_j < \varepsilon$ for any given $\varepsilon > 0$ and $j \neq J$ and $a_J > 1 - \varepsilon$. It follows, that $|a_j/a_J| < \varepsilon$ for all $j \neq J$. Then the numerator can be bounded from above by

$$\sum_{j=1}^n a_j^2 \lambda_j \leq a_J^2 (\lambda_J + \sum_{j\neq J}^n \varepsilon^2 \cdot \lambda_j) \leq a_J^2 (\lambda_J + (n-1) \cdot \lambda_{max} \cdot \varepsilon^2).$$

Similarly we can bound the denominator from below by a_J^2, which cancels with the a_J^2 from the numerator. Together we have

$$r(t) - r(x_J) \leq (\lambda_J + (n-1) \cdot \lambda_{max} \cdot \varepsilon^2) - \lambda_J = c \cdot \varepsilon^2 = \mathcal{O}(\varepsilon^2).$$

$\square$

[5] As we will see later in this chapter solving a singular system, which can be a non-trivial task, can be avoided in practice.

With this in place, we introduce a large set of eigensolver methods, which are based on the so-called *power method*:

Theorem 2.18 (Power method)**.**
Given a matrix A with $0 \leq |\lambda_1| \leq \cdots\ |\lambda_{n-1}| < |\lambda_n|$ and an initial vector $v^{(0)}$ such that $\langle v^{(0)}, x_n \rangle \neq 0$, it holds that

$$v^{(k)} := A^k v^{(0)} \to x_n, \text{ as } k \to \infty, \tag{2.21}$$

if we assume $v^{(k)}$ to be normalized after each step.[6] *Furthermore $v^{(k)}$ and its according Rayleigh quotient $\lambda^{(k)}$ satisfy*

$$\|v^{(k)} - x_n\| = \mathcal{O}\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^k\right) \text{ and } |\lambda^{(k)} - \lambda_n| = \mathcal{O}\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^{2k}\right). \tag{2.22}$$

Proof. Writing A and $v^{(0)}$ in the eigenvector basis of A we obtain

$$A^k v^{(0)} = \sum_{i=1}^n \lambda_i^k x_i {x_i^{-1}}^T a_i x_i = \sum_{i=1}^n \lambda_i^k a_i x_i = \lambda_n^k \sum_{i=1}^n \left(\frac{\lambda_i}{\lambda_n}\right)^k a_i x_i.$$

For $k \to \infty$ all fractions within the sum except for the last one vanish, proving Eq. (2.21). Eq. (2.22) then follows directly from this result and Theorem 2.17. □

We include a pseudocode of the power method in Algorithm 2.8. Here, and in several other basic algorithms, we omit an explicit stopping criterion.

Algorithm 2.8: Power method

input: Matrix A, initial vector $v^{(0)}$ with $\|v^{(0)}\| = 1$
output: Approximation of largest eigenpair $(\lambda^{(k)}, v^{(k)})$
for $k = 1, 2, \ldots$
  $w \leftarrow Av^{(k-1)}$
  $v^{(k)} \leftarrow w/\|w\|$
  $\lambda^{(k)} \leftarrow {v^{(k)}}^T A v^{(k)}$

The convergence of the power method is guaranteed as long as $|\lambda_n| > |\lambda_{n-1}|$, but it can be arbitrarily slow if the largest eigenvalues are close to each other. Moreover, this approach only determines the largest eigenpair of a given matrix A, so in order to gain access to arbitrary eigenvectors within the spectrum we can shift and invert the matrix A to achieve convergence towards a specified

[6] Otherwise $v^{(k)} \to \infty$ if $|\rho(A)| > 1$ or $v^{(k)} \to 0$ if $|\rho(A)| < 1$.

region of the spectrum, see Remark 2.6. Choosing a shift σ sufficiently close the target eigenvalue λ_J, the power method for $(A-\sigma I)^{-1}$ will converge towards λ_J as the closest eigenvalue to σ, since $|(\sigma-\lambda_J)^{-1}| \gg |(\sigma-\lambda_K)^{-1}| > \ldots$, where λ_K is the second-closest eigenvalue to σ. This method, called *inverse iteration*, is derived from the power method by replacing the matrix vector multiplication $w = Av^{(v-1)}$ from Algorithm 2.8 with the solution of a linear system of equations $(A-\sigma I)w = v^{(v-1)}$. As might be expected, the convergence results of these methods are closely related.

Theorem 2.19 (Convergence of inverse iteration)**.**
Given a matrix A, a shift $\sigma \in \mathbb{C}$ such that $|\sigma-\lambda_J| < |\sigma-\lambda_K| \leq \ldots$ and an initial vector $v^{(0)}$ such that $v^{(0)} \not\perp x_J$ we find that after k steps of inverse iteration $v^{(k)}$ and $\lambda^{(k)}$ satisfy

$$\|v^{(k)} - x_J\| = \mathcal{O}\left(\left|\frac{\sigma-\lambda_J}{\sigma-\lambda_k}\right|^k\right) \text{ and } |\lambda^{(k)} - \lambda_J| = \mathcal{O}\left(\left|\frac{\sigma-\lambda_J}{\sigma-\lambda_k}\right|^{2k}\right). \tag{2.23}$$

Each step of inverse iteration is typically significantly more expensive compared to a single step of the power method, but it shows a superior convergence rate, if the shift σ is chosen properly. We can further improve inverse iteration by updating σ throughout the iterations with the Rayleigh quotient $r(v^{(k)})$. This method is called *Rayleigh quotient iteration* and is described in Algorithm 2.9.

Algorithm 2.9: Rayleigh quotient iteration

input: Matrix A, initial vector $v^{(0)}$ with $\|v^{(0)}\| = 1$
output: Approximation of largest eigenpair $(\lambda^{(k)}, v^{(k)})$

```
λ^(0) = v^(0)T A v^(0)
for k = 1, 2, ...
    Solve (A − λ^(k−1) I) w = v^(k−1) for w
    v^(k) ← w/‖w‖
    λ^(k) ← v^(k)T A v^(k)
```

Combining the convergence results of inverse iteration and the accuracy of the Rayleigh quotient, we end up with *cubic* convergence of the Rayleigh quotient iteration for symmetric/Hermitian matrices,[7] if $v^{(0)}$ is sufficiently close to the target eigenvector x_J.

Theorem 2.20 (Convergence of the Rayleigh quotient iteration)**.**
Given a symmetric/Hermitian matrix A, an eigenvector x_J and an initial vector

[7]For general matrices the convergence is at most quadratic, cf. [90].

$v^{(0)}$ such that $v^{(0)}$ is sufficiently close to x_J. After k steps of the Rayleigh quotient iteration $v^{(k)}$ satisfies

$$\|v^{(k)} - x_J\| = \mathcal{O}(\|v^{(k-1)} - x_J\|^3) \text{ and } |\lambda^{(k)} - \lambda_J| = \mathcal{O}(|\lambda^{(k-1)} - \lambda_J|^3).$$

Remark 2.21.
The impressive cubic convergence of the Rayleigh quotient iteration only holds locally. In practice it is therefore common to perform a few steps of inverse iteration before starting to update the shift, in order to first obtain a sufficiently accurate approximation to the target eigenvector.

2.4.2 Subspace accelerated eigensolvers

In the previous section, all presented methods determine a single eigenpair for a given matrix. Here, we introduce methods, which can obtain an arbitrary amount of eigenpairs. One classical approach known as *Wielandt's deflation* [90] is based on appropriately shifted systems, which aim at changing the position of the known eigenvalue without changing the position of the others. Considering the matrix

$$A_1 = A - \sigma x_1 v^H,$$

where x_1 is the known (right) eigenvector, v is arbitrarily chosen such that $v^H x_1 = 1$ and σ is an appropriate shift, we can show that the spectrum of A_1 is given by

$$\Lambda(A_1) = \{\lambda_1 - \sigma, \lambda_2, \ldots, \lambda_n\}$$

and that the according eigenvectors are preserved. This procedure can further be generalized to include a block of deflated eigenvalues and has several variations depending on the choice of v. However, we are still restricted to approximating one eigenvector at a time and are not making full use of the information provided, i.e., we only use the information from the current iterate, while discarding all previous ones. In the following we present subspace accelerated eigensolver methods, which will be used throughout this thesis.

Arnoldi's method for finding eigenpairs. The subspace generated by Arnoldi's method is based on the Krylov subspace $\mathcal{K}_m = \{r, Ar, A^2r, \ldots, A^m r\}$. As we have already seen in Theorem 2.18, $A^m r$ converges towards the largest eigenvector x_0 for $m \to \infty$, such that the Arnoldi basis V should contain an accurate approximation to x_0 after a few iterations. Considering the proof of Theorem 2.18, it is intuitive to assume that good approximations to several eigenvectors with eigenvalues close to λ_0 are contained in $\mathcal{K}_m$ as well, since those eigenvalues are also amplified throughout the generation of V_m.

We can extract the approximations to these eigenpairs from $\mathcal{K}_m$ by computing so-called *Ritz pairs*:

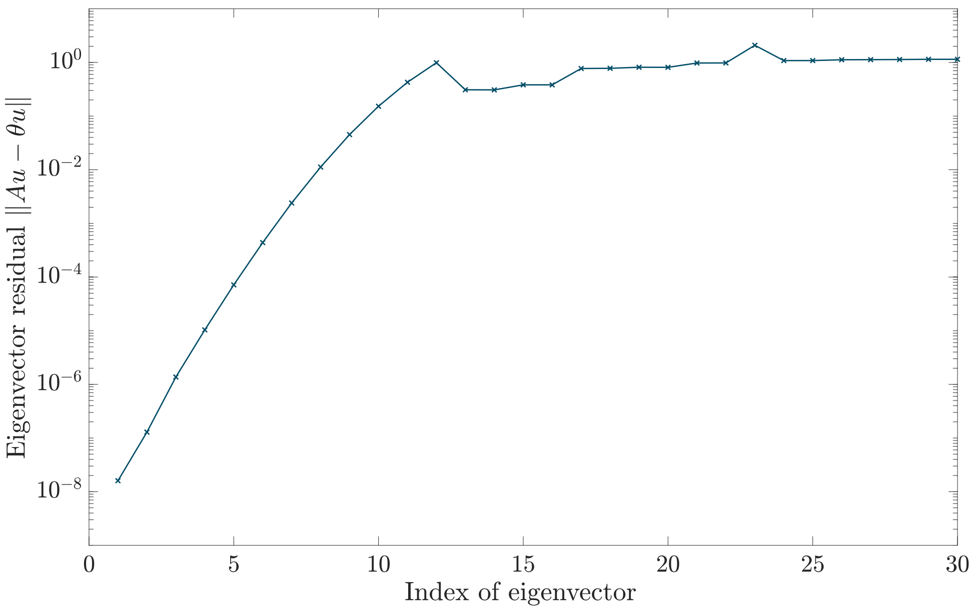


Figure 2.7: Eigenvector residual of the largest 30 eigenvectors after the largest one has converged.

Definition 2.22 (Ritz values and Ritz vectors)**.**
Let A be an arbitrary matrix and $H_{m,m} := V_m^H A V_m$ an orthogonal projection of A onto the subspace spanned by a set of m orthonormal vectors V_m. Given the eigenvalue decomposition

$$H_{m,m} = S^{-1}\Theta S$$

we call the eigenvalues θ_i of $H_{m,m}$ the Ritz values *of A with corresponding* Ritz vectors $u_i := V_m s_i$, *where s_i is the i-th column of S. The pair (θ_i, u_i) is called a* Ritz pair.

Figure 2.7 demonstrates the accuracy of Ritz vectors. After applying enough Arnoldi steps such that we obtain the first eigenvector x_0, we can see that the eigenvectors with eigenvalues close to λ_0, have eigenvector residuals ranging from 10^{-1} down to 10^{-7} already.

We can thus in principle easily obtain eigenpairs by performing Arnoldi's method and compute the Ritz pairs of the upper Hessenberg matrix $H_{m,m}$ until we found the desired amount of eigenpairs. However, there are several restrictions for this approach.

Computing the eigenvalues of $H_{m,m}$ at each step of Arnoldi's method is too expensive, instead they are only computed periodically during the algorithm, and then their accuracy has to be checked, to see whether enough eigenpairs have converged. This also entails that the search space needs to be significantly larger than the number of sought eigenpairs, which becomes prohibitively expensive for larger amounts of sought eigenpairs. Additionally, the basic method approximates exterior eigenvalues first before finding the interior ones, which limits its practicability. While we could employ the shift-and-invert technique from inverse iteration, we would need to obtain high accuracy solutions for the linear system of equations $(A-\sigma)^{-1}x = b$ to maintain an exact Krylov structure for the Arnoldi relation to hold, which is significantly more expensive than applying matrix vector multiplications.

In the next paragraph we will see how the requirement for exact solves can be eliminated, while simultaneously gaining more flexibility.

Generalized Davidson method. The *generalized Davidson* method [25, 90] is an eigensolver framework which can be seen as a generalization of Arnoldi's method [74].[8] Its advantage is that it does not rely on a Krylov subspace structure and thus offers a more flexible way of steering the search space into a desired region. In particular, the algorithm can be easily formulated such that it finds one target eigenpair, or a region of eigenpairs first. This is achieved by successively generating a set of orthogonal vectors $v_1, \ldots, v_m$, which span the search space $\mathcal{V}_m$. Similar to Arnoldi's method, the eigenpair extraction is then performed by obtaining the Ritz pairs of the matrix $H_{m,m} := V_m^H A V_m$. Afterwards the Ritz pair (θ, u), which is closest to the target eigenpair (region), is chosen. The search space is then extended by a new vector t which is obtained as a function of the matrix A and the eigenvector residual $r := Au - \theta u$. The new vector v_{m+1} is then retrieved after orthogonalizing t against $v_1, \ldots, v_m$ and normalizing it.

For this thesis we focus on obtaining t as an (approximate) solution of the *correction equation*

$$(A - \tau I)t = r, \tag{2.24}$$

where τ is an estimate for the target eigenvalue. The choice of τ steers the expansion of the search space, and with it the Ritz values, towards the desired eigenvalue regions, e.g., eigenvalues with smallest absolute value or with largest imaginary part. Similar to the Rayleigh quotient iteration, it is sensible to choose $\tau = \theta$ to improve convergence towards the desired eigenpair.

A description of the generalized Davidson method to obtain one eigenpair is given in Algorithm 2.10.

The case where we want to obtain more than one eigenpair is discussed in Section 5.1.

[8] In [74] the authors relate generalized Davidson with the Lanczos method, but the statement also holds for non-Hermitian matrices.

Algorithm 2.10: Generalized Davidson (basic)

input: initial guess t, desired accuracy ε
output: eigenpair (λ, x)

$V = \emptyset$
for $m = 1, 2, \ldots$
  $t = (I - VV^H)t$
  $v_m = t/||t||_2$
  $V = [V \mid v_m]$
  $H = V^H AV$
  get target eigenpair (θ, s) of H
  $u = Vs$
  $r = Au - \theta u$
  if $||r||_2 \leq \varepsilon$ **then**
    $\lambda = \theta$, $x = u$
    return
  compute t as a function of A, r and θ

2.5 Matrix functions

Given a scalar function

$$f : \mathbb{C} \to \mathbb{C}, z \mapsto f(z)$$

and a matrix $A \in \mathbb{C}^{n\times n}$ we are interested in a useful generalization of f to matrix arguments A, i.e.,

$$f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}, A \mapsto f(A),$$

which will be referred to as a *matrix function*. For example, linear systems of equations $Ax = b$ can be interpreted as a matrix function with $f(z) = \frac{1}{z}$. Another useful application for matrix functions can be found in graph theory: Given a graph G the matrix exponential e^A of the adjacency matrix $A(G)$ gives insight into some properties of the graph, e.g., central nodes. In this work, we are interested in the matrix sign function $\mathrm{sign}(A)$, since it is required for the definition of the Neuberger Overlap operator in Chapter 4.

Some matrix functions can be easily defined in a straight-forward fashion by substituting z with A, 1 with the identity matrix I and scalar divisions with matrix inversion.

Example 2.23.
The following examples illustrate the definition of matrix functions above:

1. $f(z) = \frac{1+z^2}{1-z} \Rightarrow f(A) = (I - A)^{-1}(I + A^2) = (I + A^2)(I - A)^{-1}$, *if* $1 \notin \Lambda(A)$

2. *If f has a convergent power series representation, such as*

$$e^z = \sum_{i=0}^{\infty} \frac{z^i}{i!}$$

we can substitute A for z in the power series and define, e.g.,

$$e^A = \sum_{i=1}^{\infty} \frac{A^i}{i!}$$

3. *The scalar sign function is defined for $z \in \mathbb{C}$ lying off the imaginary axis by*

$$\operatorname{sign}(z) = \begin{cases} -1 & , \textit{if } \operatorname{Re}(z) < 0 \\ +1 & , \textit{if } \operatorname{Re}(z) > 0. \end{cases}$$

While this definition for the sign function does not lend itself to a generalization to matrix arguments, the equivalent definition $\operatorname{sign}(z) = z/(z^2)^{1/2}$ *is a more suitable representation, as we will see later:*

$$\operatorname{sign}(A) = A(A^2)^{-1/2} \tag{2.25}$$

There exist several general definitions for matrix functions [58]. We focus on the special case of diagonalizable matrices.[9]. In this case matrix functions can be reduced to the application of the scalar function to the eigenvalues of the matrix.

Lemma 2.24.
Let $A \in \mathbb{C}^{n\times n} = X\Lambda X^{-1}$ be the eigenvalue decomposition of A and let f be any function defined on the spectrum of A. Then

$$f(A) = f(X\Lambda X^{-1}) = Xf(\Lambda)X^{-1} = X\operatorname{diag}(f(\lambda_i))X^{-1} \textit{ for } i = 1,\dots,n.$$

Lemma 2.24 gives an idea how to approximate any matrix function $f(A)$ for a given matrix A: Obtain an approximation to the eigenvalue decomposition $A = X\Lambda X^{-1}$ and evaluate $f(A) = Xf(\Lambda)X^{-1}$. This approach will be followed in the next section.

[9]For non-diagonalizable matrices a more general definition using the Jordan canonical form can be formulated. In practice it is reasonable to assume that all occurring matrices are diagonalizable.

2.5.1 Applying a matrix function to a vector

Many applications require the action of a matrix function on a vector, i.e., $f(A)b$. For large sparse matrices this task is fundamentally different from the naive approach of evaluating $f(A)$ and multiplying it with b. Even if A is a sparse matrix, $f(A)$ will in general be a dense matrix. When matrix dimensions grow larger, storing $f(A)$ thus becomes prohibitively expensive. Similar to the case of solving linear systems of equations, iterative methods have to be applied to obtain approximate solutions. This section is based on [43] and references therein.

Following the idea at the end of the last section, Theorem 2.25 shows how Arnoldi's method can be used to obtain approximations to matrix functions.

Theorem 2.25.
Given the matrices V_m and $H_{m,m}$ from m steps of the Arnoldi process on A, $f(A)b$ can be approximated by

$$f(A)b \approx f_m = V_m f(H_{m,m}) V_m^H b = \|b\|_2 V_m f(H_{m,m}) e_1. \tag{2.26}$$

Equality holds if $m \geq \deg(\psi_{A,b})$.

Since the dimension of $H_{m,m}$ is significantly smaller compared to the dimension of A, the matrix function $f(H_{m,m})$ can be efficiently evaluated via Lemma 2.24. The vector f_m can thus be obtained directly from the matrices V_m and $H_{m,m}$ of Arnoldi's method and is available with negligible additional costs.

Similar to GMRES, this method requires restarting in order to avoid growing memory and orthogonalization costs. Due to the lack of a matrix function analogue for the residual equation (2.8), the error of the Arnoldi approximation f_m has to be derived with an alternative approach [38]. There the error is approximated using an error function based on divided differences.

This approach theoretically enables restarting for Arnoldi's method, but is not practical due to numerical instability of divided differences [27]. In [43] an integral-based error representation was developed which avoids divided differences and is thus more stable.

Theorem 2.26.
Let $\Omega \subset \mathbb{C}$ be a region and let $f : \Omega \to \mathbb{C}$ be analytic with the integral representation

$$f(z) = \int_\Gamma \frac{g(t)}{t-z} \mathrm{d}t, \ \ z \in \Omega, \tag{2.27}$$

with a path $\Gamma \subset \mathbb{C}\backslash\Omega$ *and a function* $g : \Gamma \to \mathbb{C}$. *For a matrix* A *with* $\operatorname{spec}(A) \subset \Omega$ *and a vector* b, *the error of* f_m *from Theorem 2.25 is given as*

$$f(A)b - f_m = \|b\|\gamma_m \int_\Gamma \frac{g(t)}{w_m(t)} (tI - A)^{-1} v_{m+1} \mathrm{d}t =: e_m(A)v_{m+1}, \tag{2.28}$$

where $w_m(t) = (t - \theta_1)\cdots(t - \theta_m)$ *and* $\gamma_m = \prod_{i=1}^m h_{i+1,i}$.

The integral from Equation (2.28) can be evaluated in a stable way using an appropriate quadrature rule. Algorithm 2.11 shows how this approach can be implemented using an adaptive error control. For further information, see [43].

Algorithm 2.11: Quadrature-based restarted Arnoldi's method for $f(A)b$

input: Matrix A, vector b, function f, restart length m, tolerance ε
output: Approximation $f_m \approx f(A)b$

Perform m steps of Arnoldi's method to obtain $AV_m^{(1)} = V_m^{(1)} H_{m,m}^{(1)} + h_{m+1,m}^{(1)} v_{m+1}^{(1)} e_m^T$ w.r.t. A and b.
$f_m^{(1)} \leftarrow \|b\| V_m^{(1)} f(H_{m,m}^{(1)}) e_1$
$\tilde{\ell} \leftarrow 8$ and $\ell \leftarrow \operatorname{round}(\sqrt{2} \cdot \tilde{\ell})$
for $k = 2, 3, \ldots$
  Perform m steps of Arnoldi's method to obtain $AV_m^{(k)} = V_m^{(k)} H_{m,m}^{(k)} + h_{m+1,m}^{(k)} v_{m+1}^{(k)} e_m^T$ w.r.t. A and $v_{m+1}^{(k-1)}$.
  Choose sets $(\tilde{t}_i, \omega_i)_{i=1,\ldots,\tilde{\ell}}$ and $(t_i, \omega_i)_{i=1,\ldots,\ell}$ of quadrature nodes/weights.
  `accurate` $\leftarrow$ `false` and `refined` $\leftarrow$ `false`
  while `accurate` $=$ `false` **do**
    Compute $\tilde{h}_m^{(k)} = e_m^{(k-1)}(H_{m,m}^{(k)}) e_1$ by quadrature of order $\tilde{\ell}$.
    Compute $h_m^{(k)} = e_m^{(k-1)}(H_{m,m}^{(k)}) e_1$ by quadrature of order ℓ.
    if $\|h_m^{(k)} - \tilde{h}_m^{(k)}\| < \varepsilon$ **then**
      `accurate` $\leftarrow$ `true`
    else
      $\tilde{\ell} \leftarrow \ell$ and $\ell \leftarrow \operatorname{round}(\sqrt{2} \cdot \tilde{\ell})$
      `refined` $\leftarrow$ `true`
  $f_m^{(k)} \leftarrow f_m^{(k-1)} + \|b\| V_m^{(1)} h_m^{(k)}$
  if `refined` $=$ `false` **then**
    $\ell \leftarrow \tilde{\ell}$ and $\tilde{\ell} \leftarrow \operatorname{round}(\ell/\sqrt{2})$

Chapter 3

Basics of quantum chromodynamics

In this chapter we give an introduction into Quantum Chromodynamics as far as relevant for this thesis. We start with a brief overview of continuum QCD and the description of the Dirac operator, which will be central for most parts of the thesis. We present the most common discretization of the Dirac operator, the *Wilson-Dirac operator* and some properties thereof as well as derive a measure of its normality. We close this chapter with some applications, which motivated the development of the new methods derived within this thesis. Sections 3.1 and 3.2 are mostly from [46], while Section 3.3 is a result of our work published in [16].

3.1 Continuum QCD

Quantum Chromodynamics is a quantum field theory in four-dimensional space-time and a fundamental part of the standard model of particle physics today. It describes the strong interaction between quarks and gluons, the elementary particles which make up composite particles (hadrons), such as protons or neutrons. This interaction is described by the skew-adjoint continuum Dirac equation

$$(\mathcal{D} + mI)\psi = \eta, \tag{3.1}$$

where $\psi = \psi(x) \in \mathbb{C}^{12}$ and $\eta = \eta(x) \in \mathbb{C}^{12}$ are called *spinors* or *quark fields*. The twelve components $\psi_{c,\sigma}$ label the internal degrees of freedom, the so-called color $c = (1, 2, 3)$ and spin $\sigma = (0, 1, 2, 3)$ of a given spinor at a point $x = (x_0, x_1, x_2, x_3)$ in space-time. The ordering of the degrees of freedom of a spinor is given as follows:

$$\psi(x) = (\psi_{1,0}(x), \psi_{2,0}(x), \psi_{3,0}(x), \psi_{1,1}(x), \dots, \psi_{3,3}(x))^T$$

The scalar parameter m sets the quark mass of the QCD theory. The Dirac operator $\mathcal{D}$ describes the interaction between the quarks for a given gluon background field and is defined as

$$\mathcal{D} = \sum_{\mu=0}^{3} \gamma_\mu \otimes (\partial_\mu + A_\mu), \tag{3.2}$$

where ∂_μ is a shorthand for $\partial/\partial x_\mu$. The Hermitian and unitary γ-matrices $\gamma_0, \gamma_1, \gamma_2, \gamma_3 \in \mathbb{C}^{4\times4}$ generate a Clifford algebra, satisfying

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = \begin{cases} 2 \cdot I, & \mu = \nu \\ 0, & \mu \neq \nu \end{cases} \text{ for } \mu, \nu \in \{0, \dots, 3\} \tag{3.3}$$

and act nontrivially on the spin indices of the spinor and trivially on the color indices: $(\gamma_\mu\psi)(x) = (\gamma_\mu \otimes I_3)\psi(x)$. The gauge field $A_\mu(x)$ describes the connection between different (but infinitesimally close) space-time points and is defined by skew-Hermitian traceless matrices which are elements of $\mathfrak{su}(3)$, the Lie algebra of the special unitary group SU(3). It acts trivially on the spin and nontrivially on the color, i.e., $(A_\mu\psi)(x) = (I_4 \otimes A_\mu(x))\psi(x)$.

The covariant derivative $\partial_\mu + A_m u$ is a *minimal coupling extension* of the derivative ∂_μ, ensuring that $((\partial_\mu + A_\mu)\psi)(x)$ is being transformed in the same way as $\psi(x)$ under local gauge transformations, i.e., local changes of the coordinate system in color space.

The combination of covariant derivatives and the γ-matrices ensures that $\mathcal{D}\psi(x)$ transforms under the space-time transformations of special relativity in the same way as $\psi(x)$. Local gauge invariance and special relativity are fundamental principles of the standard model of elementary particle physics [46].

3.2 The Wilson discretization

In order to obtain predictions in the QCD theory, e.g., quark masses or interaction between particles, partial differential equations involving the Dirac operator have to be solved. However, these PDEs cannot be evaluated analytically, thus the QCD theory has to be formulated on a (finite) lattice and numerically approximated using high performance computing.

For the numerical simulation of a quantum system, we can only simulate a finite region, thus it has to be large enough such that the quarks in the interior of the system can interact without suffering from so-called *finite volume effects* or *cutoff effects*, i.e., interference of artificial boundary constraints. One common approach to limit these effects is to impose periodic boundary conditions, where each dimension is transformed into a circle, leading to a four-dimensional torus as our underlying region for the Dirac operator. In order to obtain a discretized version of the Dirac operator, it has to be reformulated to operate on a finite lattice. In particular, suitable discrete alternatives for the derivatives ∂_μ and the gauge fields A_μ have to be formulated.

We start with a periodic $N_t \times N_s^3$ lattice $\mathcal{L} \subset \mathcal{T}$ on a four-dimensional torus $\mathcal{T}$. The lattice $\mathcal{L}$ has $n_{\mathcal{L}} = N_t \cdot N_s^3$ lattice points with a lattice spacing a, where N_t denotes the number of lattice points in time dimension and N_s denotes the number of lattice points in the three space dimensions.

We define special shift vectors $\hat{\mu} \in \mathbb{R}^4$ by

$$\hat{\mu}_\nu = \begin{cases} a, & \mu = \nu \\ 0, & \mu \neq \nu, \end{cases}$$

which allow to formulate dependencies of neighboring lattice points.

The gauge fields $A_\mu(x)$, which connect infinitesimally close space-time points are replaced by *gauge links* $U_\mu(x) \in SU(3)$, which connect two neighboring lattice points x and $x + \hat{\mu}$. The opposing link from $x + \hat{\mu}$ to x is given by $U_\mu(x)^{-1}$. The unitary matrices $U_\mu(x)$ are approximations to the path-ordered exponential of the integral of A_μ along the link, i.e., $U_\mu(x) := e^{-aA_\mu(x+\frac{1}{2}\hat{\mu})} \approx e^{-\int_x^{x+\hat{\mu}} A_\mu(s)ds}$. The set of all gauge links $\{U_\mu(x) : x \in \mathcal{L}, \mu = 0, 1, 2, 3\}$ is called a *configuration*.

To discretize the covariant derivative of the continuum theory, we define forward covariant finite differences

$$(\Delta^\mu \psi_\sigma)(x) := \frac{1}{a}\big(U_\mu(x)\psi_\sigma(x + \hat{\mu}) - \psi_\sigma(x)\big)$$

and backward covariant finite differences

$$(\Delta_\mu \psi_\sigma)(x) := \frac{1}{a}\big(\psi_\sigma(x) - U_\mu^H(x - \hat{\mu})\psi_\sigma(x - \hat{\mu})\big),$$

where $\psi_\sigma(x)f \in \mathbb{C}^3$ denotes all color variables of ψ with spin σ. From the definition of the gauge links, we see that $(\Delta^\mu)^H = -\Delta_\mu$, and that the centralized covariant finite differences $(\Delta^\mu + \Delta_\mu)/2$ are skew-Hermitian, which leads to the *naive* discretization of the Dirac operator

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

This naive Dirac operator is skew-Hermitian, since the γ matrices are Hermitian, while the covariant finite differences are skew-Hermitian. However, this discretization suffers from a common phenomenon when discretizing first order derivatives using central finite differences, known as the *species doubling effect* or *red-black instability*, cf. [96]. Each eigenvalue of D_n has a 16-dimensional eigenspace, whereas only one eigenvector corresponds to an actual eigenfunction of the continuum operator. To avoid this effect, Wilson introduced a stabilization term $a\Delta_\mu\Delta^\mu$, which leads to the *Wilson-Dirac operator* [108]

$$D = D(m_0) = D_W(m_0) = \frac{m_0}{a}I + \frac{1}{2}\sum_{\mu=0}^{3}\left(\gamma_\mu \otimes (\Delta^\mu + \Delta_\mu) - aI_4 \otimes \Delta_\mu\Delta^\mu\right), \quad (3.4)$$

where similar to the continuum operator, m_0 defines the mass of the simulated quark. The Wilson-Dirac operator D inherits a nontrivial symmetry from the continuum operator leading to the Hermitian Wilson-Dirac operator Q, which is required for certain simulations, cf. [5]. By defining $\gamma_5 := \gamma_0\gamma_1\gamma_2\gamma_3$, we can see that $\gamma_5\gamma_\mu = -\gamma_\mu\gamma_5$ for $\mu = 0, 1, 2, 3$, thus $\gamma_5\gamma_\mu$ is skew-Hermitian. Thus each term $(\gamma_5\gamma_\mu) \otimes (\Delta^\mu + \Delta_\mu)$ for $\mu = 1, \ldots 4$ is Hermitian since it is the tensor product of two skew-Hermitian operators. If we define $\Gamma_5 := I_{n_\mathcal{L}} \otimes \gamma_5 \otimes I_3$, we see that the operator $Q := \Gamma_5 D$ is Hermitian, i.e., $(\Gamma_5 D)^H = \Gamma_5 D$.

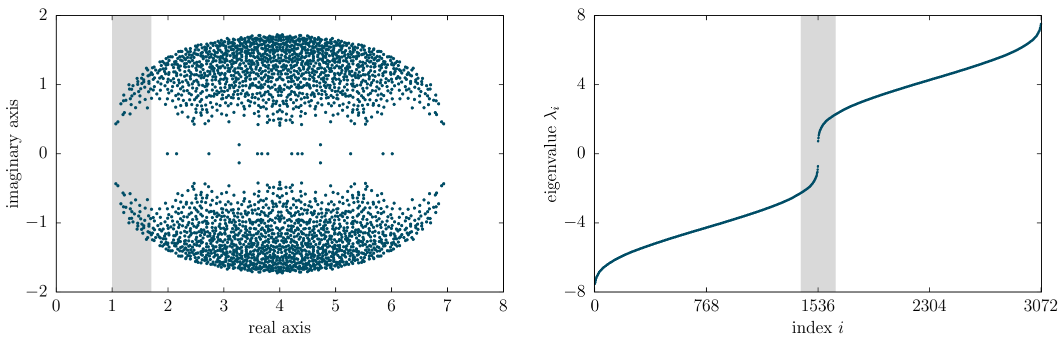


Figure 3.1: Full spectra of D and Q for configuration 1. The left plot shows the spectrum of D in the complex plane; the right plot shows the spectrum of the Hermitian operator Q and illustrates its spectral density.

Figure 3.1 shows an example of the spectrum of D and Q. For D we can observe a symmetry with respect to the real axis as well as a symmetry with respect to the line $\mathrm{Re}(z) = \frac{m_0+4}{a}$. To prove the first symmetry, let (λ, x) be an eigenpair of D and recall that $\Gamma_5 D = (\Gamma_5 D)^H$. Then we have

$$(\Gamma_5 x)^H D = x^H(\Gamma_5 D) = (\Gamma_5 D x)^H = (\Gamma_5 \lambda x)^H = \bar{\lambda}(\Gamma_5 x)^H. \quad (3.5)$$

The second symmetry follows from a reordering of the lattice sites in a red-black fashion, cf. [85].

The species doubling effect is handled by the Wilson term, which pushes the non-physical eigenvalues and its eigenvectors to the right, leaving the physical eigenpairs at the lower part of the spectrum (the shaded area in Figure 3.1). The spectrum of the Hermitian Wilson operator Q is shown in the right plot and is *maximally indefinite*, i.e., it has the same amount of positive and negative eigenvalues, with both parts being distributed in a similar way. The shaded area highlights the eigenvalues with smallest absolute value, which are of special interest for this thesis.

In Table 3.1 we provide information on the configurations used within this thesis. For proof of concept and prototypes in MATLAB [71] we used configurations 1 and 2 which were generated using our own heat-bath algorithm. Configurations 3–6 were used for the eigenvalue computations in Chapter 5 and were provided by our partners at the University of Regensburg within the Collaborative Research Centre SFB-TRR55; see [6]. For these configurations, we actually use *clover improved* (see [92]) Wilson-Dirac operators D and Q, where a block diagonal term with 6×6 diagonal blocks depending on a parameter c_{sw} is added to improve the lattice discretization error from $\mathcal{O}(a)$ to $\mathcal{O}(a^2)$. For these configurations c_{sw} is set to 1.9192. Note, that the resulting modified D is still Γ_5-Hermitian. Configurations 7 and 8 were used for the overlap simulations in Chapter 4.

ID	lattice size $N_t \times N_s^3$	kernel mass m_0	default overlap mass μ	smearing s	provided by
1	4×4^3	-0.7867	–	–	–
2	8×8^3	-0.7573	–	–	–
3	48×24^3	-0.3289	–	–	RQCD, cf. [6]
4	64×32^3	-0.3321	–	–	RQCD, cf. [6]
5	64×40^3	-0.3321	–	–	RQCD, cf. [6]
6	64×64^3	-0.3321	–	–	RQCD, cf. [6]
7	32×32^3	$-1 - \frac{3}{4}\sigma_{\min}$	0.0150000	$\{0, \ldots, 6\}$-stout [75]	cf. [31, 32]
8	32×32^3	-1.3	0.0135778	3 HEX [22]	BMW-c [12, 101]
9	64×64^3	-0.0529	–	2 HEX [22]	BMW-c [21, 35]

Table 3.1: Configurations used within this thesis together with some relevant parameters. See the references for further details. Configurations 1 and 2 are locally generated configurations.

3.3 Normality of the Wilson-Dirac operator

In the continuum case the Dirac matrix $\mathcal{D}$ is normal, i.e., $\mathcal{D}^*\mathcal{D} = \mathcal{D}\mathcal{D}^*$. Normal matrices have the numerical advantage of being unitarily diagonalizable, i.e., $A = U\Lambda U^H$, with U unitary and Λ the diagonal matrix containing the eigenvalues of A. This property is crucial for achieving a good performance of the preconditioner presented in Chapter 4. However, normality does not carry over to the discretized

Wilson-Dirac operator D, where the Frobenius norm of the commutator $||D^H D - DD^H||_F$ is non-zero in general.

One approach to reduce the non-normality is to decrease the lattice spacing for a constant volume, which implies that larger lattices are required. In this case the discretized operator approaches the continuum operator and thus becomes more normal. Since this increases computational cost significantly this approach is infeasible in practice.

Instead, we establish a connection of $||D^H D - DD^H||_F$ with the *Wilson gauge action*. In physical simulations so-called *smearing* techniques are used to reduce the Wilson gauge action and in turn reduce the non-normality of D for a given lattice. This is a result from our work published in [16].

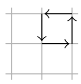
Definition 3.1 (Wilson gauge action)**.**
The Wilson gauge action $S_W(\mathcal{U})$ *for a gauge field* $\mathcal{U} = \{U_\mu(x)\}$ *is given by*

$$S_W(\mathcal{U}) := \sum_x \sum_{\mu<\nu} \mathrm{Re}(\mathrm{tr}(I - Q_x^{\mu,\nu})), \tag{3.6}$$

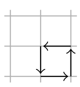
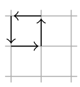
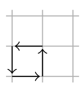
where the first sum is to be taken over all lattice sites x and $\sum_{\mu<\nu}$ is a shorthand for $\sum_{\mu=0}^{3}\sum_{\nu=\mu+1}^{3}$. The terms $Q_x^{\mu,\nu}$ are called plaquettes *and are the product of all coupling matrices along a cycle of length 4 on the lattice, i.e.,*

$$Q_x^{\mu,\nu} := U_\nu(x)U_\mu(x+\hat{\nu})U_\nu(x+\hat{\mu})^H U_\mu(x)^H. \tag{3.7}$$

The plaquettes can be represented as squares in the (μ,ν)-plane, e.g.,

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

Similarly, the other plaquettes at x are defined as

$Q_x^{\mu,-\nu} \mathrel{\widehat{=}}$ , $Q_x^{-\mu,\nu} \mathrel{\widehat{=}}$ , $Q_x^{-\mu,-\nu} \mathrel{\widehat{=}}$  . (3.8)

In the following theorem we prove that the deviation of the plaquettes from the identity matrix can be used as a measure for the non-normality of D.

Theorem 3.2.
The Frobenius norm of $D^H D - DD^H$ fulfills

$$||D^H D - DD^H||_F^2 = 16 \sum_x \sum_{\mu<\nu} \mathrm{Re}(\mathrm{tr}(I - Q_x^{\mu,\nu})) = 16 \cdot S_W(\mathcal{U}). \tag{3.9}$$

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

Table 3.2: Coupling terms in D and D^H.

Proof. We inspect the entries of $D^HD - DD^H$. We use the notation $\pi_\mu^\pm$ for the matrices

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

The relations (3.3) between the γ-matrices show that each $\pi_\mu^\pm$ is a projection and that, in addition,

$$\pi_\mu^+\pi_\mu^- = \pi_\mu^-\pi_\mu^+ = 0,\ \mu = 0,\dots,3. \tag{3.10}$$

Considering all 12 variables at each lattice site as an entity, the graph associated with the nearest neighbor coupling represented by the matrix D is the 4d-torus, and similarly for D^H. Table 3.2 gives the non-zero entries of a (block) row in D and D^H in terms of the 12×12 matrices which couple lattice site x with the sites x and $x \pm \hat{\mu}$.

The product D^HD represents a coupling between nearest and next-to-nearest lattice sites; the coupling 12×12 matrices are obtained as the sum over all paths of length two on the torus of the product of the respective coupling matrices in D^H and D. A similar observation holds for DD^H. Table 3.3 reports all the entries of D^HD, and we now shortly discuss all the paths of length 2 which contribute to these entries of D^HD.

For the diagonal position (x,x) we have 21 paths of length 2, $(x,x) \to (x,x) \to (x,x)$ and $(x,x) \to (x,x\pm\hat{\mu}) \to (x,x), \mu = 0,\dots,3$. The contribution of each of the latter 20 paths is 0 due to (3.10).

For a nearest neighbor $(x,x+\hat{\mu})$ we have the two paths $(x,x) \to (x,x) \to (x,x+\hat{\mu})$ and $(x,x) \to (x,x+\hat{\mu}) \to (x,x+\hat{\mu})$, and similarly in the negative directions. For a position $(x,x\pm 2\hat{\mu})$ there is only one path $(x,x) \to (x,x\pm\hat{\mu}) \to (x,x\pm 2\hat{\mu})$, with the product of the couplings being 0 due to (3.10). Finally, for the other next-to-nearest neighbors we always have two paths, for example $(x,x) \to (x,x+\hat{\mu}) \to (x+\hat{\mu}-\hat{\nu})$ and $(x,x) \to (x,x-\hat{\nu}) \to (x+\hat{\mu}-\hat{\nu})$.

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

This shows that in $D^HD - DD^H$ the only no-vanishing coupling terms are those at positions $(x,x+\hat{\mu}+\hat{\nu})$, $(x,x+\hat{\mu}-\hat{\nu})$ and $(x,x-\hat{\mu}-\hat{\nu})$ for $\mu \neq \nu$. They are

(x,x)	$m^2 I_{12}$
$(x,x+\hat\mu)$	$-m(\pi_\mu^+ + \pi_\mu^-)\otimes U_\mu(x)$
$(x,x-\hat\mu)$	$-m(\pi_\mu^+ + \pi_\mu^-)\otimes U_\mu(x-\hat\mu)$
$(x,x\pm 2\hat\mu)$	0
$\nu\neq\mu$:	
$(x,x+\hat\mu+\hat\nu)$	$\pi_\mu^-\pi_\nu^+\otimes U_\mu(x)U_\nu(x+\hat\mu)+\pi_\nu^-\pi_\mu^+\otimes U_\nu(x)U_\mu(x+\hat\nu)$
$(x,x+\hat\mu-\hat\nu)$	$\pi_\mu^-\pi_\nu^-\otimes U_\mu(x)U_\nu^H(x+\hat\mu-\hat\nu)+\pi_\nu^+\pi_\mu^+\otimes U_\nu^H(x-\hat\nu)U_\mu(x-\hat\nu)$
$(x,x-\hat\mu-\hat\nu)$	$\pi_\mu^+\pi_\nu^-\otimes U_\mu^H(x-\hat\mu)U_\nu^H(x-\hat\mu-\hat\nu)+\pi_\nu^+\pi_\mu^-\otimes U_\nu^H(x-\hat\nu)U_\mu^H(x-\hat\nu-\hat\mu)$

Table 3.3: Coupling terms in D^HD. The coupling terms in DD^H are obtained by interchanging all π_μ^+ and π_μ^- as well as all π_ν^+ and π_ν^-.

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

Table 3.4: Coupling terms in D^HD-DD^H.

given in Table 3.4, where we used the identities

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

By rearranging the terms we obtain the plaquettes from (3.7) and (3.8). We exemplify this for position $(x,x+\hat\mu+\hat\nu)$

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

Using the fact that for the Frobenius norm we have

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

we obtain the following for the squares of the Frobenius norms of all the coupling matrices from Table 3.4:

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

Finally for any unitary matrix Q we have

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

Now we obtain $\|D^H D - DD^H\|_F^2$ by summing the squares of the Frobenius norms of all coupling matrices. This is a sum over $24n$ coupling matrices, n being the number of lattice sites. As discussed before, groups of four of these coupling matrices refer to the same plaquette $Q_x^{\mu,\nu}$ up to conjugation in SU(3), so $\mathrm{tr}(I-Q)$ is the same for these four plaquettes Q. We can thus "normalize" to only consider all possible "first quadrant" plaquettes $Q_x^{\mu,\nu}$ and obtain

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

$\square$

As a consequence of Theorem 3.2 we conclude that D is normal in a trivial case, called the *free theory*, where all gauge links are equal to the identity. For any physically relevant configuration D we thus assume D to be non-normal. While in theory a normal Dirac operator might be desirable, in practice a "near-normal" operator is already sufficient, since in this case the field of values already excludes the origin, as we will demonstrate in Section 4.3.

In current simulations with large lattices a technique called *smearing* is extensively used either to improve the signal-to-noise ratio of physics predictions [1], or to stabilize the stochastic process of generating gauge field configurations [22, 75**?**]. This process locally averages gauge links along the plaquettes and thus reduces the Wilson gauge action, with the goal to reduce "cut-off effects" (discretization errors) related to short-distance fluctuations in the gauge field, e.g., un-physical localized eigenvectors with near zero eigenvalues. Several smearing techniques have been proposed, e.g., APE [1], HYP [56], HEX [22], while in this thesis we mainly consider *stout* smearing [75]. The following description of stout smearing follows [16].

Given a gauge field $\mathcal{U}$, stout smearing modifies the gauge links according to

$$U_\mu(x) \to \tilde{U}_\mu(x) = \mathrm{e}^{\epsilon Z_\mu^{\mathcal{U}}(x)} U_\mu(x) \tag{3.11}$$

where the parameter ϵ is a small positive number and

$$Z_\mu^{\mathcal{U}}(x) \;=\; -\frac{1}{2}(M_\mu(x) - M_\mu^H(x)) + \frac{1}{6}\,\mathrm{tr}(M_\mu(x) - M_\mu^H(x))\,, \tag{3.12}$$

where

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

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

The following result from [69, 70] relates the *Wilson flow* $\mathcal{V}_\tau = \{V_\mu(x,\tau) : x \in \mathcal{L}, \mu = 0,\dots,3\}$ defined as the solution of the initial value problem

$$\frac{\partial}{\partial\tau} V_\mu(x,\tau) = -\left\{\partial_{x,\mu} S_{\mathrm{W}}(\mathcal{V}_\tau)\right\} V_\mu(x,\tau)\,, \quad V_\mu(x,0) = U_\mu(x)\,, \tag{3.13}$$

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

Theorem 3.3.
Let $\mathcal{V}_\tau$ be the solution of (3.13). *Then*

(i) $\mathcal{V}_\tau$ is unique for all $\mathcal{V}_0$ and all $\tau \in (-\infty,\infty)$ and differentiable with respect to τ and $\mathcal{V}_0$.

(ii) The Wilson action $S_{\mathrm{W}}(\mathcal{V}_\tau)$ is monotonically decreasing as a function of τ.

(iii) One step of Lie-Euler integration with step size ϵ for (3.13)*, starting at $\tau = 0$, gives the approximation $\mathcal{V}'_\epsilon = \{V'_\mu(x,\epsilon)\}$ for $\mathcal{V}_\epsilon$ with*

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

with $Z_\mu^{\mathcal{U}}(x)$ from (3.12)

Proof. We refer to [69, 70] and also [10] for details of the proof for (i) and (ii). The idea is to show that

$$\frac{\partial}{\partial\tau} S_{\mathrm{W}}(\mathcal{V}_\tau) = 2\sum_{x,\mu} \mathrm{tr}((\partial_{x,\mu} S_{\mathrm{W}}(\mathcal{V}_\tau))^2)\,, \tag{3.14}$$

with $-\partial_{x,\mu} S_{\mathrm{W}}(\mathcal{U}) = Z_\mu^{\mathcal{U}}(x)$ skew-hermitian and thus $\mathrm{tr}((\partial_{x,\mu} S_{\mathrm{W}}(\mathcal{V}_\tau))^2) \geq 0$. Part (iii) follows directly by applying the Lie-Euler scheme; cf. [55]. $\square$

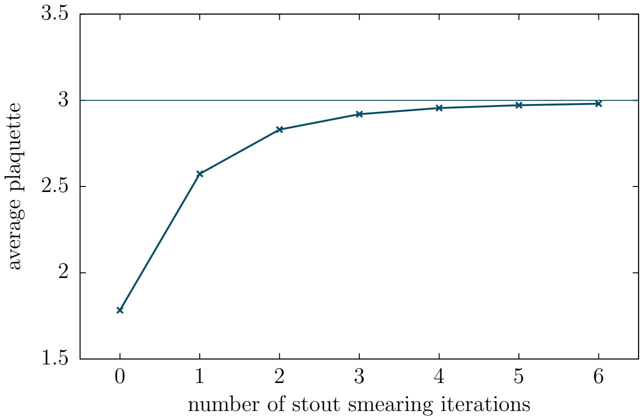


Figure 3.2: Illustration of the effect of stout smearing on the average plaquette value (3.15).

The theorem implies that one Lie-Euler step is equivalent to a step of stout smearing, with the exception that in stout smearing links are updated sequentially instead of in parallel. And since the Wilson action decreases along the exact solution of (3.13), we can expect it to also decrease for its Lie-Euler approximation, at least when ϵ is sufficiently small.

In Figure 3.2, we illustrate the relation between iterations of stout smearing and the average plaquette value Q_{avg} for configuration 7 (cf. Table 3.1). The average plaquette value is defined by

$$Q_{avg} = N_Q^{-1} \sum_x \sum_{\mu<\nu} \mathrm{Re}(\mathrm{tr}(Q_x^{\mu,\nu})), \tag{3.15}$$

where N_Q denotes the total number of plaquettes. In terms of Q_{avg}, (3.9) simplifies to

$$||D_W^H D_W - D_W D_W^H||_F^2 = 16 N_Q (3 - Q_{avg}).$$

Figure 3.2 shows that the Wilson action, and with it the non-normality of D, decreases rapidly in the first iterations of stout smearing and approaches $\mathrm{tr}(I) = 3$.

3.4 Applications

The following sections illustrate some possible applications for the newly developed methods from this thesis, which are presented in Chapters 4, 5 and 6. Section 3.4.1 presents a "controversial" alternative discretization for the continuum Dirac operator, where our work from Chapter 4 was fundamental in validating this approach, see [13]. The development of the eigensolver in Chapter 5 was motivated by an eigenvector deflation approach for the computation of $\mathrm{tr}(Q^{-1})$ and is based on previous work from [5]. This is summarized in Section 3.4.2. Our new setup strategy from Chapter 6 is developed with the hybrid Monte Carlo algorithm in mind, which we present in Section 3.4.3.

3.4.1 Validation of the staggered Wilson discretization

The Wilson-Dirac discretization is one of the first discretizations of the continuum Dirac operator, but breaks a fundamental symmetry called *chiral symmetry*, due to the stabilization term $aI_4\Delta_\mu\Delta^\mu$. In [64] the authors derive the *Kogut-Susskind* (or more commonly referred to as *staggered*) discretization, which respects chiral symmetry. Simulations with the staggered Wilson operator have smaller systematic errors and thus allow for a more controlled continuum extrapolation, compared to simulations using the Wilson-Dirac operator. In addition the computational cost of both operators are comparable, making the staggered formalism a more favorable discretization in many cases. The following derivation of the staggered operator is based on [20].

Similarly to the Wilson-Dirac discretization it uses the forward and backward covariant finite differences Δ^μ and Δ_μ, but instead of adding an stabilization term to avoid the species doubling effect, it uses a spin diagonalization of the gauge links by redefining the quark fields via

$$\psi(x) \to \Omega(x)\psi(x),$$

with $\Omega(x) = \gamma_1^{x_1} \cdot \gamma_2^{x_2} \cdot \gamma_3^{x_3} \cdot \gamma_4^{x_4}$.

Using local transformations, the γ-matrices can be transformed into local (negative) identities for each direction μ

$$\begin{aligned} &\psi^H(x)\gamma_\mu \otimes [U(x)\psi(x+\mu) - U(x-\mu)\psi(x-\mu)] \\ \to\, &\psi^H(x)\eta_\mu(x) \otimes [U(x)\psi(x+\mu) - U(x-\mu)\psi(x-\mu)]\,, \end{aligned}$$

where $\eta_\mu(x) := \Omega(x)^H\gamma_\mu\Omega(x+\mu) = (-1)^{(x_1+x_2+x_3+x_4)}$.

With this transformation the staggered Dirac operator can be written as

$$D_s = D_s(m_0) = \frac{m0}{a}I + \frac{1}{2}\sum_\mu \eta_\mu(x) \otimes (\Delta^\mu + \Delta_\mu). \tag{3.16}$$

Similar to the naive Dirac operator, the staggered Dirac operator is skew-Hermitian up to the mass shift m_0. The spin diagonalization partially solves the species doubling effect, since it leaves four identical spin systems, and thus reduce the amount of doublers from 16 to 4. The remaining doublers can be removed by taking the forth root of D_s, see e.g., [26]. However, this rooting procedure is highly debated in the physics community, and arguments against this method were made, e.g., in [23].

In order to experimentally validate the staggered discretization, it is compared to the (Neuberger) overlap discretization in [13]. Simulations with the overlap operator also exhibit an exact chiral symmetry on the lattice, but are significantly more expensive compared to other discretizations. For this reason, efficient numerical methods have to be developed to allow for affordable continuum extrapolations with the overlap operator, which is done in Chapter 4.

3.4.2 Low-mode averaging

When computing certain physical observables, so-called *n-point (Green) functions* are oftentimes employed. For example in the two-flavor $n_f = 2$ theory, the eta-meson $C_\eta(x, y)$ correlator is a two-point function of the form

$$\begin{aligned} C_\eta(x, y) &= \langle O_x^\eta, \bar{O}(y)^\eta \rangle \\ &\propto \operatorname{tr}(Q^{-1}_{x,y} Q^{-1}_{y,x}) - n_f \operatorname{tr}(Q^{-1}_{x,x}) \operatorname{tr}(\underbrace{Q^{-1}_{y,y}}_{=Q^{-1}}), \end{aligned} \tag{3.17}$$

where $\bar{O}(y)^\eta$ and O_x^η are the *creation* and *destruction* operators, which create a state (in this case an eta meson) at point y and destroy it at point x, cf. [5]. In order to compute $C_\eta(x, y)$, the traces in Eq. (3.17) have to be evaluated. We can exploit translation invariance to reduce the evaluation of $\operatorname{tr}(Q^{-1}_{x,y} Q^{-1}_{y,x})$ to the computation of 12 inversions of the Wilson-Dirac operator, which can be performed relatively cheaply. However, the second term involves $\operatorname{tr}(Q^{-1})$ which is prohibitively expensive to evaluate exactly in practice, due to the size of Q.

Instead, we can employ a stochastic Monte Carlo approach to approximate $\operatorname{tr}(Q^{-1})$. For large $N \in \mathbb{N}$ and random vectors $v_i \in (\mathbb{Z}_2 + i\mathbb{Z}_2)^n$, $i = 1, \ldots, N$ the following approximation holds:

$$\operatorname{tr}(Q^{-1}) \approx \frac{1}{N} \sum_{i=1}^{N} v_i^H Q^{-1} v_i. \tag{3.18}$$

The convergence of this method is of order $\mathcal{O}(\operatorname{Var}(R)/\sqrt{N})$, where $\operatorname{Var}(R)$ is the variance of the random variable $v_i^H Q^{-1} v_i$. This implies that a large amount of inversions $Q^{-1} v_i$ have to be performed, which is particularly expensive. In order to reduce the computational work of this Monte Carlo approach, a technique called low-mode averaging [7, 41] can be applied, which splits the operator $Q^{-1} =$

$Q_{low}^{-1} + Q_{high}^{-1}$ into two parts. Q_{low}^{-1} is comprised of the N_{eig} lowest eigenmodes of Q, i.e., the eigenpairs (λ_i, x_i) with eigenvalues closest to zero

$$Q_{low}^{-1} = \sum_{i=1}^{N_{eig}} \frac{1}{\lambda_i} x_i^H x_i, \tag{3.19}$$

while $Q_{high}^{-1} := ((I - XX^H)Q)^{-1}$ is the remaining part, where X is the set of eigenvectors x_i, $\imath = 1, \ldots, N_{eig}$. With this, the computation of $\mathrm{tr}(Q^{-1})$ can be reduced to the computation of the traces of Q_{low}^{-1} and Q_{high}^{-1}. The trace $\mathrm{tr}(Q_{low}^{-1})$ can be directly evaluated from the already computed eigenpairs:

$$\mathrm{tr}(Q_{low}^{-1}) = \sum_{i=1}^{N_{eig}} \frac{1}{\lambda_i}$$

For large enough N_{eig} the computation of $\mathrm{tr}(Q_{high}^{-1})$ becomes significantly cheaper compared to $\mathrm{tr}(Q^{-1})$ since the condition number of Q_{high}^{-1} is smaller, thus the inversions become cheaper and less iterations for the Monte Carlo algorithm have to be performed, since this approach also reduces the variance of the random variable. However, low-mode averaging comes with the additional cost of having to compute the N_{eig} smallest eigenpairs, which can be particularly expensive. In Chapter 5 we thus introduce an efficient eigensolver for this purpose.

3.4.3 Hybrid Monte Carlo

Most numerical experiments in lattice QCD are related to stochastic approaches, thus they have to be repeated multiple times with different configurations in order to obtain predictions with controlled statistical errors. These configurations are typically generated using the so-called *hybrid Monte Carlo* algorithm (HMC) [33]. This algorithm generates a set of configurations using *Markov chains*, where each configuration is an "evolution" of the previous one. We give an outline of the fundamental idea, see [52], and briefly discuss some details afterwards, as a comprehensive description of this algorithm is out of the scope for this thesis.

1. Choose initial configuration U_0 and set $i = 1$.
2. Generate random momentum fields P conjugate to U_{i-1}.
3. Evolve the configuration U_{i-1} to obtain new candidate U'.
4. Accept $U_i = U'$ with some probability P_{acc}, otherwise set $U_i = U_{i-1}$, $i \leftarrow i+1$ and go to step 2.
5. If U_i is *thermalized*, save it.

6. Go to step 2 until enough configurations are generated.

For the initial configuration, two common approaches are a *cold start*, where all gauge links are set to the identity, or a *hot start*, where all gauge links are random elements of $SU(3)$. The momentum fields P can be interpreted as "directions" in which the gauge fields are being changed in every step. In step 4 the acceptance rate P_{acc} of the new configuration is determined using the *Metropolis(-Hastings)* algorithm [57, 72], which enables us to generate a prescribed equilibrium distribution. This is important to ensure that physically more likely configurations are also more likely to be produced by the HMC algorithm. The term "thermalized" in step 5 can be interpreted as a "converged" configuration. Given enough steps of the HMC algorithm, the configuration space reaches an *equilibrium state* in which the distribution of the gauge links follows the prescribed equilibrium distribution, such that new physical configurations can be generated by going back to step 2 and repeat the procedure.

In this thesis step 3 of the HMC algorithm is of special interest. For the evolution of the gauge field U_{i-1}, a matrix inversion, typically involving the (Hermitian) Dirac operator, has to be performed. This can be performed using a multigrid algorithm, but this comes with a significant downside. The expensive multigrid setup needs to be redone in every step, as the configuration changes in every step. With the setup being the most expensive part of the multigrid algorithm, it is important to implement efficient setup procedures, which will be explored in Chapter 6.

Chapter 4

Auxiliary space preconditioning for the overlap operator in lattice QCD

The Wilson-Dirac discretization is a widely used operator in lattice QCD. However, it lacks a fundamental property of the continuous theory called *chiral symmetry*, which is only recovered in the continuum case, i.e., for lattice spacings approaching zero. This makes a controlled continuum extrapolation particularly demanding, since large lattices have to be used to reduce cutoff effects [11, 13].

In the following section, we review the *(Neuberger) overlap operator*, an alternative to the Wilson-Dirac discretization, which respects chiral symmetry. The cost for solving linear systems with this operator are as much as two orders of magnitude larger compared to standard discretizations, since every matrix vector multiplication with the overlap operator requires the evaluation of the matrix sign function, which in practice leads to two nested iterative methods.

We propose a new preconditioning technique, which accelerates linear solvers for this operator by at least a factor of ten compared to standard methods. The fundamental principle of our preconditioner is to use a standard discretization of the Dirac equation to form a preconditioner for the overlap operator. This may be regarded as a variant of the fictitious (or auxiliary) space preconditioning technique [77] that has been used for developing and analyzing multilevel preconditioners for various nonconforming finite element approximations of PDEs; cf. [83, 110]. In this context, a mapping from the original space to a fictitious space is formulated, which yields an equivalent problem that is easier to solve. Preconditioning is then done by (approximately) solving this equivalent problem. The convergence properties of auxiliary space preconditioning depend on the choice of the fictitious space and its computational efficiency additionally depends on the efficiency of the solver used in that space; cf. [77]. For our application the

Wilson-Dirac operator is a natural choice as an auxiliary space preconditioner, since it is defined on the same Hilbert space and since there are efficient solvers known for this problem. The validity of this choice is evaluated with a broad range of numerical experiments at the end of this chapter.

This work has been published in [16], and our description is based on this publication.

4.1 Chiral operators in lattice QCD

Chiral symmetry is an important property in lattice QCD, since it allows for more precise simulations in which electromagnetic fields are involved. In mathematical terms it translates to that γ_5 and $\mathcal{D}$ are anti-commuting, i.e., $\gamma_5\mathcal{D} + \mathcal{D}\gamma_5 = 0$. Since γ_5 is Hermitian it implies that any skew-Hermitian $\mathcal{D}$ fulfills this criterion. While the naive Dirac operator is skew-Hermitian and thus chiral, the Wilson-Dirac discretization breaks chiral symmetry due to its stabilization term. Finding an alternative discretization, which additionally respects chiral symmetry, is a difficult task, due to the *Nielsen-Ninomiya no-go theorem* [79]. We recast it into a mathematical notation:

Theorem 4.1 (Nielsen-Ninomiya no-go theorem).
No lattice discretization $\widehat{D}$ of the continuum Dirac operator $\mathcal{D}$ can hold all of the following properties simultaneously:

1. *$\widehat{D}$ is local*
2. *$\widehat{D}$ is translation invariant*
3. *$\widehat{D}$ is a doubler free discretization*
4. *$\widehat{D}$ anti-commutes with γ_5*

Proof. See [79] □

$\widehat{D}$ being local means that interaction between quarks only happen at a local scale, i.e., the spinor on any given lattice point only depends on its neighboring spinors. In mathematical terms a local $\widehat{D}$ implies a sparse matrix representation, which is necessary to efficiently apply iterative methods for the solution of linear systems of equations with this operator.

Since translation invariance is a fundamental requirement for a sensible discretization, this theorem implies that in practice we have to choose between chiral symmetry or a freedom of doublers. Fortunately, the limitation of this theorem can be overcome by defining an alternative lattice variant of chiral symmetry on the lattice:

Theorem 4.2 (Ginsparg-Wilson relation [48])**.**
If a Dirac operator $\widehat{D}$ *satisfies the* Ginsparg-Wilson relation

$$\Gamma_5 \widehat{D} + \widehat{D}\Gamma_5 = a\widehat{D}\Gamma_5\widehat{D}, \tag{4.1}$$

it exhibits an exact chiral symmetry of the fermion action.

Proof. See [66] $\square$

In [78] Neuberger successfully constructed a discretization respecting the Ginsparg-Wilson relation. The essentials of the arguments in [78] are summarized in the following theorem.

Theorem 4.3 (Neuberger's overlap operator)**.**
The Neuberger overlap operator

$$D_N = \frac{1}{a}\left(\rho I + D_W(m_0^{ker})\Big(D_W(m_0^{ker})^H(D_W(m_0^{ker})\Big)^{-\frac{1}{2}}\right)$$

fulfills Eq. (4.1) *for* $\rho = 1$*, has local discretization error* $\mathcal{O}(a)$*, and is a stable discretization provided* $-2 < m_0^{ker} < 0$*.*

Proof. We write $\mathcal{D}_\mathcal{L}$ for the restriction of the continuum Dirac operator $\mathcal{D}$ from Eq. (3.1) to the lattice $\mathcal{L}$, i.e., $\mathcal{D}_\mathcal{L}$ is the finite-dimensional operator that takes the same values as $\mathcal{D}$ at the points from $\mathcal{L}$. The fact that the Wilson-Dirac operator has first order local discretization error can then be expressed as [1]

$$\mathcal{D}_\mathcal{L} = D_W(0) + \mathcal{O}(a),$$

implying

$$\mathcal{D}_\mathcal{L} + \frac{m_0}{a} I = D_W(m_0) + \mathcal{O}(a) \tag{4.2}$$

for any mass parameter m_0.

To construct D_N, we first note that any operator $\widehat{D}$ that is Γ_5-symmetric and fulfills Eq. (4.1) can be parametrized by

$$a\widehat{D} = I + \Gamma_5 S, \tag{4.3}$$

[1] For simplicity, we consider here the "naive" limit $a \to 0$. In the full quantum theory one has $\mathcal{D}_\mathcal{L} = D_W(m_0(a)) + \mathcal{O}(a)$ with the mass $m_0(a)$ of order $1/\log(a)$; see [73].

with $S^H = S$ and $S^2 = I$. Both conditions are fulfilled for

$$S = \Gamma_5 D_W(m_0^{ker})\Big(D_W(m_0^{ker})^H (D_W(m_0^{ker})\Big)^{-\frac{1}{2}}, \quad -m_0^{ker} \in \mathbb{R} \setminus \operatorname{spec}(D_W(0)).$$

Using (4.2), we obtain

$$S = \Gamma_5\Big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\Big)\Big(\big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)^H \big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)\Big)^{-\frac{1}{2}}.$$

Since $\mathcal{D}$ is skew-adjoint, we have $\mathcal{D}_\mathcal{L}^* = -\mathcal{D}_\mathcal{L}$ and, thus,

$$\begin{aligned}
&\Big(\big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)^* \big(\mathcal{D}_\mathcal{L} + \tfrac{m_0^{ker}}{a} I + \mathcal{O}(a)\big)\Big)^{-\frac{1}{2}} \\
&= \tfrac{a}{|m_0^{ker}|}\Big(\big(\tfrac{a}{m_0^{ker}}\mathcal{D}_\mathcal{L} + I + \mathcal{O}(a^2)\big)^* \big(\tfrac{a}{m_0^{ker}}\mathcal{D}_\mathcal{L} + I + \mathcal{O}(a^2)\big)\Big)^{-\frac{1}{2}} \\
&= \tfrac{a}{|m_0^{ker}|}\Big(I - \big(\tfrac{a}{m_0^{ker}}\big)^2 \mathcal{D}_\mathcal{L}^2 + \mathcal{O}(a^2)\Big)^{-\frac{1}{2}} \;=\; \tfrac{a}{|m_0^{ker}|} I + \mathcal{O}(a^3),
\end{aligned}$$

which in turn yields

$$S = \Gamma_5\Big(\frac{a}{|m_0^{ker}|}\mathcal{D}_\mathcal{L} + \operatorname{sign}(m_0^{ker}) I + \mathcal{O}(a^2)\Big). \tag{4.4}$$

Combining (4.4) with (4.3), we find that

$$a\widehat{D} = I + \frac{a}{|m_0^{ker}|}\mathcal{D}_\mathcal{L} + \operatorname{sign}(m_0^{ker}) I + \mathcal{O}(a^2).$$

Thus, for $m_0^{ker} < 0$ we have

$$\widehat{D} = \frac{1}{|m_0^{ker}|}\mathcal{D}_\mathcal{L} + \mathcal{O}(a).$$

This shows that $\widehat{D}$ is a first order discretization of $\mathcal{D}$. In addition if $-2 < m_0^{ker} < 0$, $\widehat{D}$ is also stable, see [78]. Finally, note that $D_N = \widehat{D} + \frac{\rho-1}{a} I$, so $\rho - 1$ defines the quark mass (see Eq. (3.4)) up to a renormalization factor. □

The argument within the sign function is called the *kernel* of the overlap operator. Using the Wilson-Dirac operator is the most popular choice for the kernel, although other kernel operators have been investigated as well [28, 36]. With the definition of the matrix sign function from Section 2.5, we can formulate the Neuberger overlap operator as

$$D_N = \rho I + \Gamma_5 \operatorname{sign}\big(\Gamma_5 D_W(m_0^{ker})\big). \tag{4.5}$$

Since $\Gamma_5 D_W(m_0)$ is Hermitian (see Section 3.2), the matrix $\mathrm{sign}(\Gamma_5 D_W(m_0^{ker}))$ is also Hermitian. Since $\Gamma_5^2 = I$, we also see that the overlap operator satisfies the same Γ_5-symmetry as its kernel D_W:

$$\left(\Gamma_5 D_N\right)^H = \Gamma_5 D_N. \tag{4.6}$$

The next theorem characterizes spectral properties of the overlap operator:

Theorem 4.4.
The overlap operator D_N is normal. Its spectrum is symmetric to the real axis and part of the circle with midpoint ρ and radius 1, i.e.,

$$\lambda \in \mathrm{spec}\left(D_N\right) \Rightarrow \overline{\lambda} \in \mathrm{spec}\left(D_N\right) \text{ and } |\lambda - \rho| = 1 \text{ for } \lambda \in \Lambda(D_N).$$

Proof. Since the sign function is its own inverse and since $\Gamma_5 D_W(m_0)$ is Hermitian, $\mathrm{sign}(\Gamma_5 D_W(m_0))$ is its own inverse and Hermitian, thus unitary. Its product with the unitary matrix Γ_5 is unitary as well, implying that all its eigenvalues have modulus one. As a unitary matrix, this product is also normal. The term ρI in Eq. (4.5) preserves normality and shifts the eigenvalues by ρ.

It remains to show that $\mathrm{spec}(D_N)$ is symmetric with respect to the real axis, which follows from the Γ_5-symmetry (4.6) of the overlap operator in the same manner as for the Wilson-Dirac operator, cf. Eq (3.5). □

4.2 Multigrid preconditioning for the overlap operator

The motivation for our preconditioning method stems from the spectral properties of the Wilson-Dirac and the Neuberger overlap operator, see Figure 4.1. The spectral gaps to be observed as four discs with relatively few eigenvalues in the left part of Figure 4.1 are typical for the spectrum of the Wilson-Dirac operator and become even more pronounced as lattice sizes increase. In practice, the mass parameter m_0 that appears in the definition of the kernel $D_W(m_0^{ker})$ of the overlap operator is chosen such that the origin lies in the middle of the leftmost of these discs. For this choice of m_0^{ker}, we now motivate why the Wilson-Dirac operator $D_W(m_0^{prec})$ with adequately chosen mass m_0^{prec} provides a good preconditioner for the overlap operator.

To do so, we investigate the connection of the spectrum of the overlap operator and the Wilson-Dirac operator in the special case that $D_W(0)$ is normal. This is

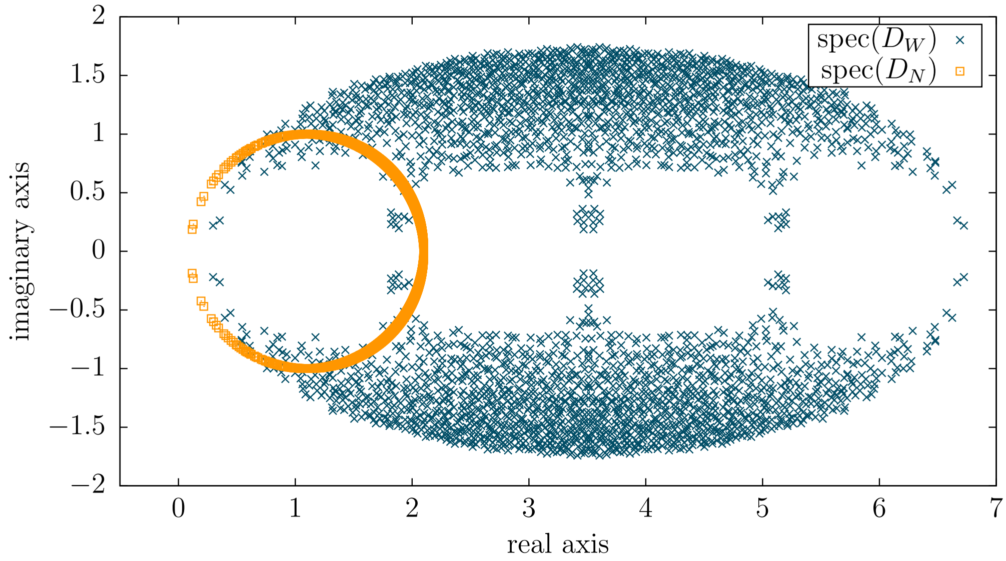


Figure 4.1: Typical spectra of the Wilson-Dirac and the overlap operator for a 4^4 lattice.

equivalent to $D_W(0)$ being unitarily diagonalizable with possibly complex eigenvalues, i.e.,

$$D_W(0) = X\Lambda X^H, \text{ with } \Lambda \text{ diagonal and } X \text{ unitary.} \tag{4.7}$$

Trivially, then, $D_W(m_0)$ is normal for all mass parameters m_0 and

$$D_W(m_0) = X(\Lambda + m_0 I)X^H. \tag{4.8}$$

To formulate the resulting non-trivial relation between the eigenvalues of D_N and its kernel $D_W(m_0^{ker})$, we use the notation $\operatorname{csign}(z)$ for a complex number z to denote its "complex" sign, i.e.,

$$\operatorname{csign}(z) = z/|z| \text{ for } z \neq 0.$$

Theorem 4.5.
Assume that $D_W(0)$ is normal, so that $D_W(m)$ is normal as well for all $m \in \mathbb{C}$, and let X and Λ be from Eq. (4.7). Then we have

$$D_N = X\big(\rho I + \operatorname{csign}(\Lambda + m_0 I)\big)X^H. \tag{4.9}$$

Proof. With Δ_m diagonal and W_m unitary, let

$$\Gamma_5 D_W(m) = W_m \Delta_m W_m^H \tag{4.10}$$

be the eigendecomposition of the Hermitian matrix $\Gamma_5 D_W(m)$. We have two different representations for the singular value decomposition of $\Gamma_5 D_W(m)$:

$$\begin{array}{rcll} \Gamma_5 D_W(m) & = & \big(\Gamma_5 X \,\mathrm{csign}(\Lambda + mI)\big) \cdot |\Lambda + mI| \cdot X^H & \text{(from (4.8))}\,, \\ \Gamma_5 D_W(m) & = & \big(W_m \,\mathrm{sign}(\Delta_m)\big) \cdot |\Delta_m| \cdot W_m^H & \text{(from (4.10))}\,. \end{array}$$

Thus, there exists a unitary matrix Q such that

$$W_m = XQ \text{ and } W_m \,\mathrm{sign}(\Delta_m) = \Gamma_5 X \,\mathrm{csign}(\Lambda + mI)Q. \tag{4.11}$$

Using the definition of D_N in (4.5), the relations (4.11) give

$$\begin{aligned} D_N &= \rho I + \Gamma_5 \,\mathrm{sign}(\Gamma_5 D_W(m)) \\ &= \rho I + \Gamma_5 W_m \,\mathrm{sign}(\Delta_m) W_m^H \\ &= \rho I + \Gamma_5 \Gamma_5 X \,\mathrm{csign}(\Lambda + mI) Q (XQ)^H \\ &= X(\rho I + \mathrm{csign}(\Lambda + mI))X^H. \end{aligned}$$

□

We remark in passing that, as an implicit consequence of the proof above, we have that the eigenvectors of $\Gamma_5 D_W(m) = \Gamma_5 D_W(0) + m\Gamma_5$ are independent of m. Thus, if D_W is normal, Γ_5 and $\Gamma_5 D_W$ admit a basis of common eigenvectors.

The result from this theorem implies that $D_N = \rho I + \Gamma_5 \,\mathrm{sign}(\Gamma_5 D_W(m_0^{ker}))$ and $D_W(0)$ share the same eigenvectors and that

$$\mathrm{spec}(D_N) = \{\rho + \mathrm{csign}(\lambda + m_0^{ker}), \lambda \in \mathrm{spec}(D_W(0))\}.$$

Taking $D_W(m_0^{prec})$ as a preconditioner for D_N, we would like eigenvalues of D_N that are small in modulus to be mapped to eigenvalues close to 1 in the preconditioned matrix $D_N D_W(m_0^{prec})^{-1}$. Since $D_W(m_0^{prec})$ and D_N share the same eigenvectors, the spectrum of the preconditioned matrix is

$$\mathrm{spec}\left(D_N D_W(m_0^{prec})^{-1}\right) = \left\{\frac{\rho + \mathrm{csign}(\lambda + m_0^{ker})}{\lambda + m_0^{prec}}, \lambda \in \mathrm{spec}(D_W(0)\right\}.$$

For $\omega > 0$ and $m_0^{prec} = \omega\rho + m_0^{ker}$, the mapping

$$g : \mathbb{C} \to \mathbb{C}, z \mapsto \frac{\rho + \mathrm{csign}(z + m_0^{ker})}{z + m_0^{prec}}$$

sends $C(-m_0^{ker}, \omega)$, the circle with center $-m_0^{ker}$ and radius ω, to one single value $\frac{1}{\omega}$. We thus expect $D_W(m_0^{prec})$ to be a good preconditioner if we choose m_0^{prec} in such a manner that the small eigenvalues of $D_W(m_0^{prec})$ lie close to $C(-m_0^{ker}, \omega)$.

Let $\sigma_{\min} > 0$ denote the smallest real part of all eigenvalues of $D_W(0)$. Assuming for the moment that $\sigma_{\min}$ is actually an eigenvalue, it must lie exactly on $C(-m_0^{ker}, \omega)$ if we take

$$\omega = \omega^{def} := -m_0^{ker} - \sigma_{\min} \text{ and thus } m_0^{prec} = m_0^{def} := \omega^{def}\rho + m_0^{ker}. \qquad (4.12)$$

For physically relevant parameters, ω^{def} is close to 1. We will take m_0^{def} from (4.12) as our default choice for the mass parameter when preconditioning with the Wilson-Dirac operator, although a slightly larger value for ω might appear adequate in situations where the eigenvalues with smallest real part come as a complex conjugate pair with nonzero imaginary part.

Although $D_W(0)$ is non-normal in physically relevant situations, we expect the above reasoning to also lead to an effective Wilson-Dirac preconditioner in these settings, and particularly so when the deviation of $D_W(0)$ from normality becomes small. This is so, e.g., when the lattice spacing is decreased while keeping the physical volume constant, i.e., in the "continuum limit", since the Wilson-Dirac operator then approaches the continuous Dirac operator, which is normal. Moreover, as we have seen in Section 3.3, smearing techniques can be applied to a given gauge configuration $U_\mu(x)$ to decrease the deviation of $D_W(0)$ from normality. Figure 4.2 shows the spectrum for the preconditioned matrix with the choice (4.12) for m_0^{prec} for the same 4^4 configuration as in Figure 4.1. The matrices in these tests are not normal; nonetheless, the spectrum of the preconditioned matrix tends to concentrate around 0.7.

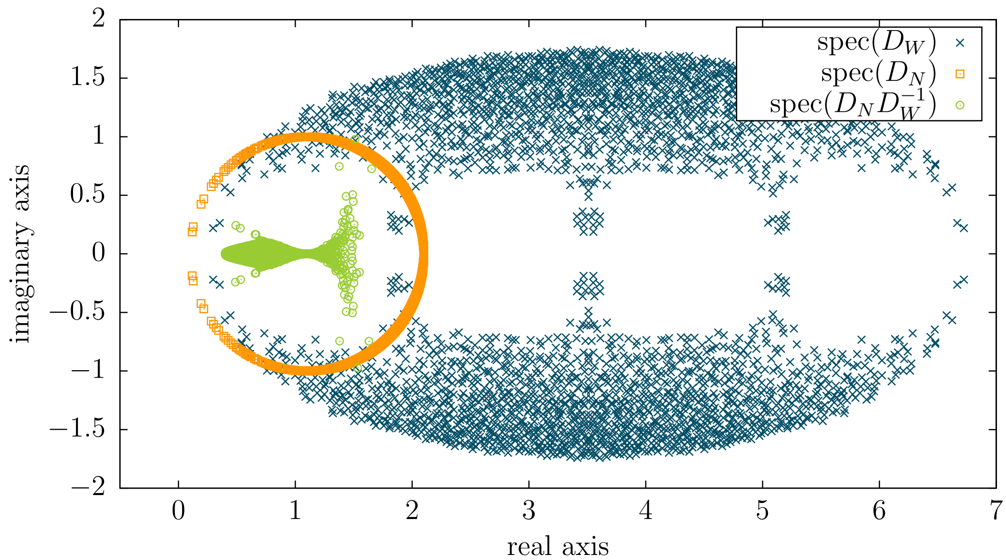


Figure 4.2: Spectra for a configuration of size 4^4

In the normal case, the singular values are the absolute values of the eigenvalues, and the singular vectors are intimately related to the eigenvectors. This relation was crucial to the proof of Theorem 4.5. In the non-normal case, relation (4.9), which uses the eigenvectors of $D_W(0)$, does not hold. For the sake of completeness, we give, for the general, non-normal case, the following result, which links the overlap operator to the singular value decomposition of its kernel $D_W(m)$.

Lemma 4.6.
Let $\Gamma_5 D_W(m) = W_m \Delta_m W_m^H$ denote an eigendecomposition of the Hermitian matrix $\Gamma_5 D_W(m)$, where Δ_m is real and diagonal and W_m is unitary. Then

(i) A singular value decomposition of $D_W(m)$ is given as

$$D_W(m) = U_m \Sigma_m V_m^H \text{ with } V_m = W_m, \Sigma_m = |\Delta_m|, U_m = \gamma_5 W_m \operatorname{sign}(\Delta_m).$$

(ii) The overlap operator with kernel $D_W(m)$ is given as

$$D_N = \rho I + \Gamma_5 \operatorname{sign}\big(\Gamma_5 D_W(m)\big) = \rho I + U_m V_m^H.$$

Proof. Since $\Gamma_5^{-1} = \Gamma_5$, we have the factorization $D_W(m) = \Gamma_5 W_m \Delta_m W_m^H = \Gamma_5 W_m \operatorname{sign}(\Delta_m)|\Delta_m|W_m^H$, in which $\Gamma_5 W_m \operatorname{sign}(\Delta_m)$ and W_m are unitary and $|\Delta_m|$ is diagonal and non-negative. This proves (i). To show (ii), just observe that, for the Hermitian matrix $\Gamma_5 D_W(m)$, we have $\operatorname{sign}(\Gamma_5 D_W(m)) = W_m \operatorname{sign}(\Delta_m) W_m^H$ and use (i). ☐

4.3 Numerical results

In this section, we report numerical results obtained on relatively large configurations used in current simulations involving the overlap operator, detailed in Table 3.1. Configuration 7 is available with different numbers $s = 0, \ldots, 6$ of stout smearing steps applied. Note that s influences $\sigma_{\min}$, the smallest real part of all eigenvalues of $D_W(0)$. The given choice for m_0^{ker} as a function of $\sigma_{\min}$, used in $D_N = \rho I + \Gamma_5 \operatorname{sign}(\Gamma_5 D_W(m_0^{ker}))$, places the middle of the first 'hole' in the spectrum of $D_W(m_0^{ker})$ to be at the origin. Configuration 8 was obtained using 3 steps of HEX smearing in a simulation similar in spirit to [12, 101] with its physics results published in [13]. The value $m_0^{ker} = -1.3$ is the one used in the simulation. The middle of the first 'hole' in $D_W(m_0^{ker})$ is thus close to but not exactly at the origin. To be in line with the conventions from [12], we express the parameter $\rho \geq 1$ used in the overlap operator D_N as

$$\rho = \frac{-\mu/2 + m_0^{ker}}{\mu/2 + m_0^{ker}},$$

where $\mu > 0$ is yet another "overlap" mass parameter.

In our experiments, we frequently consider a whole range for μ rather than just the default value from Table 3.1. The default value for μ is chosen such that it fits other physically interpretable properties of the respective configurations, like e.g., the pion mass m_π. For both sets of configurations used, m_π is approximately twice as large than the value observed in nature, and the ultimate goal is to drive m_π to its physical value, which very substantially increases the cost for generating the respective configurations. We would then use smaller values for μ, and the results of our experiments for such smaller μ hint at how the preconditioning would perform in future simulations at physical parameter values. Note that smaller values for μ make ρ closer to 1, so D_N becomes more ill-conditioned.

All results were obtained 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 [62]. This machine provides a maximum of 8,192 cores for a single job, from which we always use 1,024 in our experiments. For compilation, we used the `icc`-compiler with the optimization flags `-O3`, `-ipo`, `-axSSE4.2`, and `-m64`. In all tests, our code ran with roughly 2 Gflop/s per core, which amounts to $8-9\%$ of peak performance. The multigrid solver used to precondition with $D_W(m_0^{prec})$ (see below) performs at roughly 10% peak.

4.3.1 Accuracy of the preconditioner and influence of m_0^{prec}

In the first series of experiments, we solve the system

$$D_N\psi = \eta \tag{4.13}$$

on the one hand without any preconditioning, using GMRES(100). On the other hand, we solve the same system using D_W^{-1} as a (right) preconditioner, solving the linear systems with D_W using our DD-αAMG method.[2] In our approach, preconditioning is done by iterating with DD-αAMG until the relative residual is below a prescribed bound ε^{prec}. The setup has to be done only once for a given Wilson-Dirac operator D_W, so its cost becomes negligible when using DD-αAMG as a preconditioner in a significant number of GMRES iterations.[3] We use GMRES with odd-even preconditioning [82] as a solver for the coarsest system. The whole DD-αAMG preconditioning iteration is non-stationary, which is accounted for by using *flexible* restarted GMRES (FGMRES) [89] to solve (4.13) instead of GMRES. The restart length for FGMRES is again 100.

Figure 4.3 presents results for configuration 7 with $s = 3$ stout smearing steps and the default overlap mass μ from Table 3.1. We scanned the values for m_0^{prec} in steps

[2]Any other efficient solver for the Wilson-Dirac operator as, e.g., the "AMG" solver or the "Inexact Deflation" method, could be used as well.

[3]In all our experiments, setup never exceeded 2% of the total execution time, so we do not report timings for it.

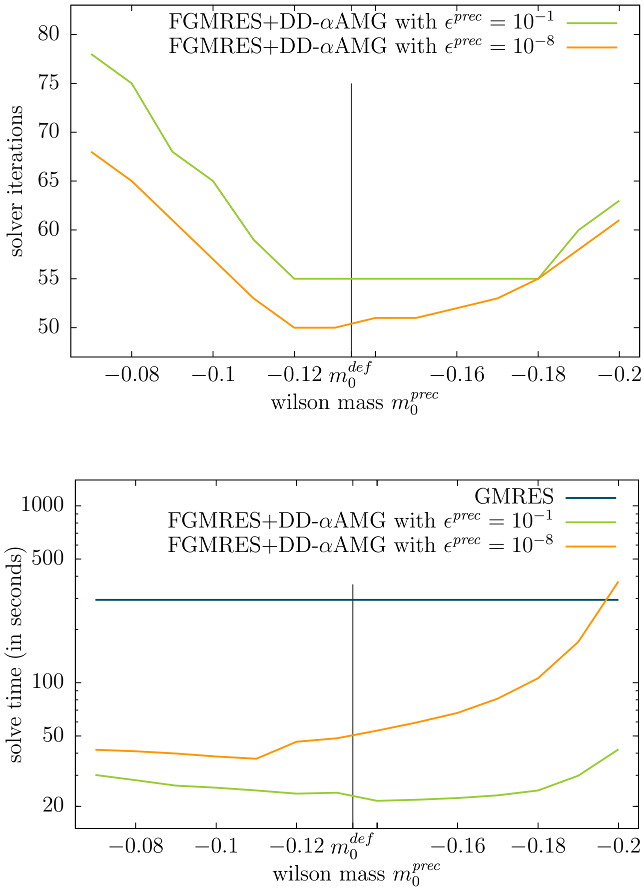


Figure 4.3: Preconditioner efficiency as a function of m_0^{prec} for two accuracies for the DD-αAMG solver (configuration 7, $s = 3$). Top: number of iterations, bottom: execution times.

of 0.01 and report the number of preconditioned FGMRES iterations necessary to reduce the initial residual by a factor of 10^{-8} for each of these values. We chose two different values of ε^{prec} for the residual reduction required in the DD-αAMG iteration in the preconditioning. The choice $\varepsilon^{prec} = 10^{-8}$ asks for a relatively accurate solution of the systems with $D_W(m_0^{prec})$, whereas the choice $\varepsilon^{prec} = 10^{-1}$ requires an only quite low accuracy and, thus, only a few iterations of DD-αAMG. The upper part of Figure 4.3 shows that there is a dependence of the number of FGMRES iterations on m_0^{prec}, while at the same time there is a fairly large interval around the optimal value for m_0^{prec} in which the number of iterations required is not more than 20% larger than the minimum. These observations hold for both accuracy requirements for the DD-αAMG solver, $\varepsilon^{prec} = 10^{-8}$ and $\varepsilon^{prec} = 10^{-1}$. The number of iterations needed without preconditioning was 973, such that our new preconditioned method requires roughly a factor of 20 less iterations.

The lower part of Figure 4.3 shows that similar observations hold for the execution times. However, the smaller iteration numbers obtained with $\varepsilon^{prec} = 10^{-8}$ do not translate into smaller execution times, since the time for each DD-αAMG solve for the preconditioner is substantially higher than for $\varepsilon^{prec} = 10^{-1}$. This turned out to hold in all our experiments, so, from now on, we invariably report results for $\varepsilon^{prec} = 10^{-1}$. We also observe that the value of m_0^{def} from (4.12) lies within an interval in which iteration numbers and execution times (for both values for ε^{prec}) are quite close to the optimum. The execution time without preconditioning was 294s, resulting into a speedup of one order of magnitude.

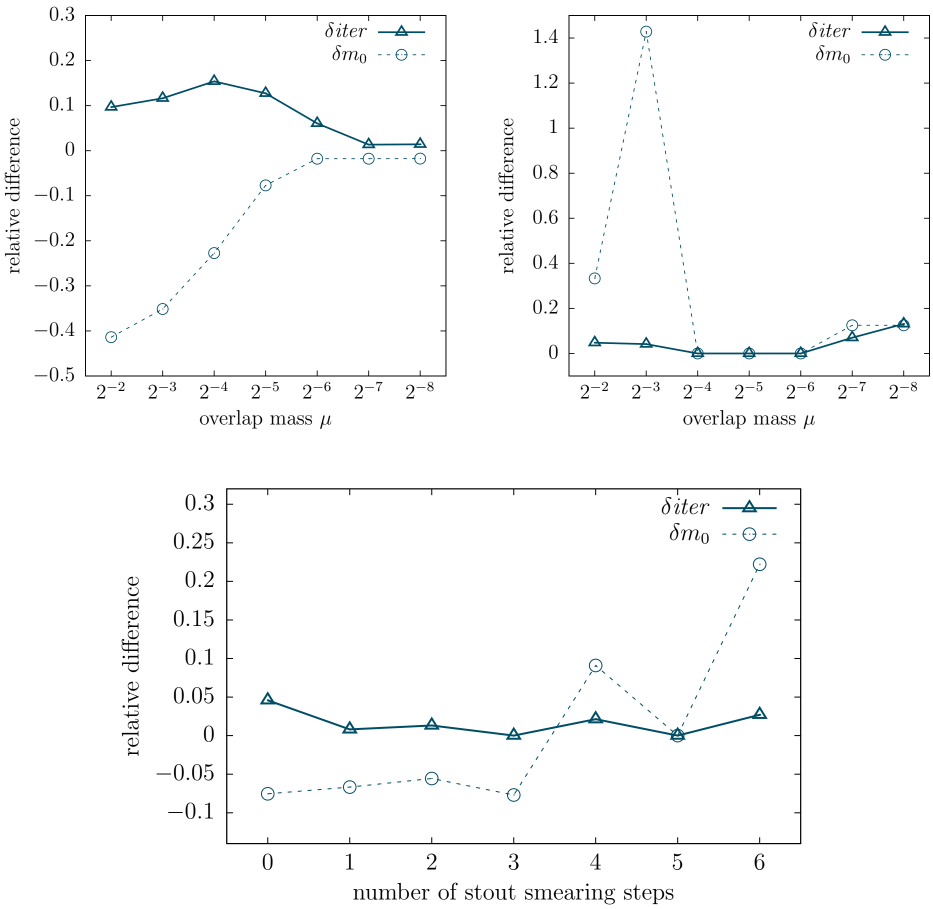


Figure 4.4: Quality of m_0^{def} without smearing (top left), with $s = 3$ steps of stout smearing (top right), and for $s = 0, \ldots, 6$ steps of stout smearing at fixed μ (bottom) for configuration 7.

Figure 4.4 reports results that show that the default value m_0^{def} is a fairly good choice in general. For two different configurations (no smearing and 3 steps of stout smearing) and a whole range of overlap masses μ, the plots at the top give the relative difference $\delta m_0 = (m_0^{opt} - m_0^{def})/m_0^{def}$ of the optimal value m_0^{opt} for

m_0^{prec} and its default value from (4.12) as well as the similarly defined relative difference $\delta iter$ of the corresponding iteration numbers. These results show that the iteration count for the default value m_0^{def} is never more than 15% off the best possible iteration count. The plot at the bottom backs these findings. We further scanned a whole range of smearing steps s at the default value for μ from Table 3.1, and the number of iterations with m_0^{def} is never more than 5% off the optimal value. The large values for δm_0 in the top right plot for $\mu = 2^{-3}$ are to be attributed to the fact that the denominator in the definition of δm_0, i.e., m_0^{def}, is almost zero in this case.

These results suggest that (4.12) is indeed a good choice for m_0^{prec}. However, $\sigma_{\min}$ needed to compute m_0^{def} from (4.12) is not necessarily known a priori, and it may be more efficient to approximate the optimal value for m_0 "on the fly" by changing its value from one preconditioned FGMRES iteration to the next.

To minimize the influence of the choice of m_0^{prec} on the aspects discussed in the following sections, we always use the optimal m_0^{prec}, computed to a precision of 10^{-2} by scanning the range $[-\widetilde{\sigma}_{\min}, 0]$, where $\widetilde{\sigma}_{\min}$ is a rough guess at $\sigma_{\min}$ that fulfills $\widetilde{\sigma}_{\min} > \sigma_{\min}$. This guess can be easily obtained by a fixed number of power iterations to get an approximation for the largest real part $\widetilde{\sigma}_{\max}$ of an eigenvalue of D and then using the symmetry of the spectrum to obtain $\widetilde{\sigma}_{\min}$ by rounding $8 - \widetilde{\sigma}_{\max}$ to the first digit.

4.3.2 Quality and cost of the preconditioner

We proceed to compare in more detail preconditioned FGMRES(100) with unpreconditioned GMRES(100) in terms of the iteration count. As before, the iterations were stopped when the initial residual was reduced by a factor of at least 10^{-8}.

Figure 4.5 gives this comparison, once as a function of the non-normality of the configuration, i.e., the number s of stout smearing steps applied, and once as a function of the overlap mass μ. We see that, for the default value of μ from Table 3.1, the quality of the preconditioner increases with the number s of stout smearing steps, ranging from a factor of approximately 5 for $s = 0$ over 12 for $s = 3$ up to 25 for $s = 6$. We also see that the quality of the preconditioner increases as μ decreases, i.e., as D_N becomes more ill-conditioned.

From the practical side, a comparison of the execution times is more important than of iteration numbers. Before giving timings of the final implementation, we must discuss relevant aspects of this implementation in some detail.

Each iteration in GMRES or preconditioned FGMRES for (4.13) requires one matrix vector multiplication by $D_N = \rho I + \Gamma_5 \operatorname{sign}(\Gamma_5 D_W)$. The matrix D_N is not given explicitly because it would be full and very large despite $\Gamma_5 D_W$

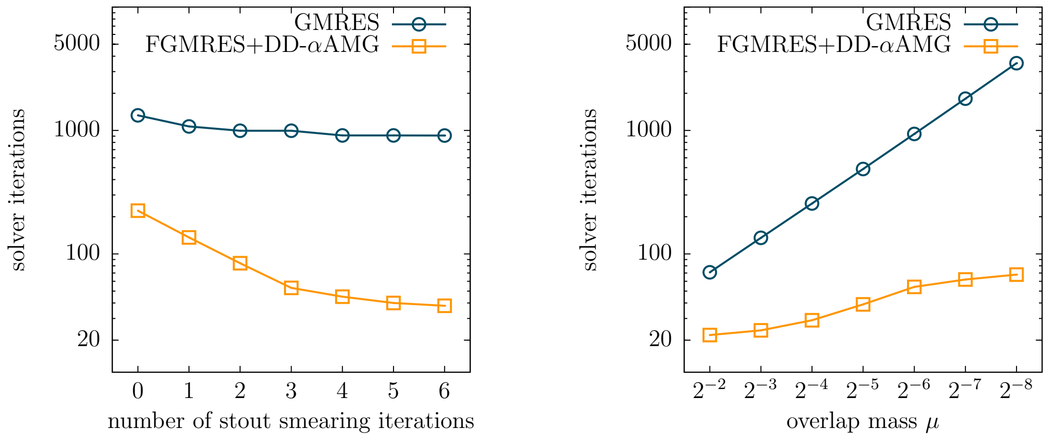


Figure 4.5: Comparison of preconditioned FGMRES(100) with unpreconditioned GMRES(100) (configuration 7). Left: dependence on the number of stout smearing steps s for default value for μ, cf. Table 3.1. Right: dependence on the overlap mass μ for $s = 3$.

being sparse. Therefore, a matrix vector multiplication $D_N\chi$ is obtained via an additional "sign function iteration" that approximates $\text{sign}(\Gamma_5 D_W)\chi$ as part of the computation of $D_N\chi$. For this we use the alternative definition of the matrix sign function $\text{sign}(A) = A(A^2)^{-1/2}$ from Example 2.23 and apply the quadrature-based restarted Arnoldi method, described in Section 2.5.1 and in [42, 43] to obtain the inverse square root of $(\Gamma_5 D_W)^2$. This method also allows for thick restarts of the Arnoldi process and has proven to be among the most efficient methods to approximate $\text{sign}(\Gamma_5 D_W)\chi$. Nevertheless, the sign function iteration still represents by far the most expensive part of the overall computation.

A first approach to reduce this cost (see [24]) is to use relaxation in the sense that one lowers the (relative) accuracy $\varepsilon_{\text{sign}}$ of the approximation as the outer (F)GMRES iteration proceeds. The theoretical analysis of inexact Krylov subspace methods in [93, 105] shows that the relative accuracy of the approximation to the matrix-vector product at iteration k should be on the order of $\varepsilon/\|r_k\|$ (with r_k the (F)GMRES residual at iteration k) to achieve a decrease of the initial residual by a factor of ε at the end of the (F)GMRES iteration. We used this relaxation strategy in our experiments.

A second commonly used approach (see, e.g., [37, 50, 104]) to reduce the cost of the sign function iteration is deflation, which is tightly related to the idea of low-mode averaging. In this approach, the k smallest in modulus eigenvalues $\lambda_1, \ldots, \lambda_k$ and their normalized eigenvectors $\xi_1, \ldots, \xi_k$ are precomputed once. With $\Xi = [\xi_1| \ldots |\xi_k]$ and $\Pi = I - \Xi\Xi^H$ the orthogonal projector on the complement of these eigenvectors, $\mathrm{sign}(\Gamma_5 D_W)\chi$ is given as

$$\mathrm{sign}(\Gamma_5 D_W)\chi = \sum_{i=1}^{k} \mathrm{sign}(\lambda_i)(\xi^H \chi)\xi_i + \mathrm{sign}(\Gamma_5 D_W)\Pi\chi.$$

The first term on the right side can be computed explicitly and the second is now easier to approximate with the sign function iteration, since the k eigenvalues closest to the singularity of $\mathrm{sign}(\cdot)$ are effectively eliminated via Π. We used this strategy as well in our experiments.

	parameter	notation	default
(F)GMRES^{dp}	required reduction of initial residual	ε_{outer}	10^{-8}
	relaxation strategy	$\varepsilon_{\mathrm{sign}}$	$\frac{\varepsilon_{outer}}{\|r_k\|} \cdot 10^{-2}$
	restart length for FGMRES	$m_{restart}$	100
DD-αAMG^{sp}	required reduction of initial residual	ε_{prec}	10^{-1}
	number of levels		2

Table 4.1: Parameters for the overlap solver. Here, *dp* denotes double precision and *sp* single precision.

Table 4.1 summarizes the default settings used for the results reported in Figure 4.6. The superscripts *dp* and *sp* indicate that we perform the preconditioning in IEEE single precision arithmetic, while the multiplication with D_N within the (F)GMRES iteration is done in double precision arithmetic. Such mixed precision approaches are a common further strategy to reduce computing times in lattice simulations.

For the results reported in Figure 4.6, we tried to keep the cost for a matrix vector multiplication with D_N independent of the number of smoothing steps that were applied to the configuration. To do so, we used the 100th smallest eigenvalue of $\Gamma_5 D_W$ for $s = 0$ as a threshold, and deflated all eigenpairs with eigenvalues below this threshold for the configurations with $s > 0$. The left plot in Figure 4.6 shows that, at fixed default overlap mass μ, we gain a factor of 4 to 10 in execution time using the preconditioner. The quality of the preconditioning improves with the numbers of smearing steps. The right part of Figure 4.6 shows that, for smaller values of μ, we can expect an even larger reduction of the execution time. For the smallest value considered, $\mu = 2^{-8}$, which is realistic for future lattice simulations, the improvement due to preconditioning is a factor of about 25.

4.3.3 Comparison of optimized solvers

Physics production codes for simulations with the overlap operator use recursive preconditioning as an additional technique to further reduce the cost for the

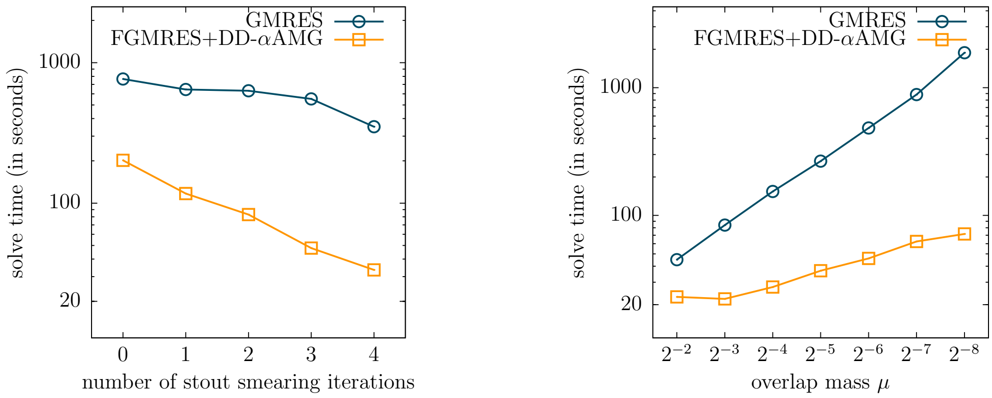


Figure 4.6: Comparison of execution times for preconditioned FGMRES and GMRES. Left: for 0 to 4 steps of stout smearing (configuration 7, default value for μ from Table 3.1), right: different overlap masses μ for configuration 7 and 3-step stout smearing.

matrix vector multiplication (MVM) with D_N; cf. [24]. This means that the FGMRES iteration is preconditioned by using an additional "inner" iteration to approximately invert D_N, this inner iteration being itself again FGMRES. The point is that we may require only low accuracy for this inner iteration, implying that all MVMs with $\mathrm{sign}(\Gamma_5 D_W)$ in the inner iteration may be approximated to low accuracy and computed in IEEE single precision only.

In this framework, we can apply the DD-αAMG preconditioner, as well, but this time as a preconditioner for the inner FGMRES iteration. In this manner, we keep the advantage of needing only a low accuracy approximation to the MVM with $\mathrm{sign}(\Gamma_5 D_W)$, while at the same time reducing the number of inner iterations and, thus, the (low accuracy) evaluations of MVMs with $\mathrm{sign}(\Gamma_5 D_W)$.

Let ε_{inner} denote the residual reduction we ask for in the unpreconditioned inner iteration and $\varepsilon_{inner}^{prec}$ the corresponding accuracy required when using the DD-αAMG iteration as a preconditioner. The inner iteration converges much faster when we use preconditioning. More accurate solutions in the inner iteration reduce the number of outer iterations and, thus, the number of costly high precision MVMs with $\mathrm{sign}(\Gamma_5 D_W)$. When preconditioning is used for the inner iteration, requiring a higher accuracy in the inner iteration comes at relatively low additional cost. It is, therefore, advantageous to choose $\varepsilon_{inner}^{prec}$ smaller than ε_{inner}. As an addition to Table 4.1, Table 4.2 lists the default values that we used for the inner iteration. They were found to be fairly optimal via numerical testing.

Figure 4.7 shows results for the solvers optimized in this way. We consider different sizes for the deflation subspace, i.e., the number of smallest eigenvalues that we deflate explicitly. The computation of these eigenvalues (via PARPACK [97])

	parameter	notation	default
inner FGMRES^{sp}	required reduction of initial residual (with preconditioning)	$\varepsilon_{inner}^{prec}$	10^{-2}
	required reduction of initial residual (without preconditioning)	ε_{inner}	10^{-1}
	relaxation strategy		$\frac{\varepsilon_{inner},\varepsilon_{inner}^{prec}}{\|r_k\|} \cdot 10^{-2}$
	restart length	$m_{restart}^{inner}$	100

Table 4.2: Parameters for the inner iteration.

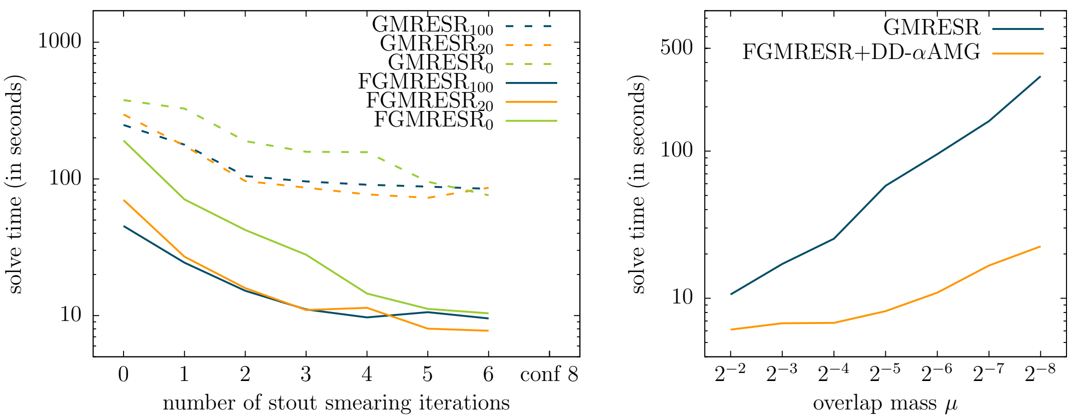


Figure 4.7: Comparison of GMRESR and FGMRESR with different deflation spaces (configuration 7 and 8 with 1,024 processes). The lower index denotes the amount of deflated eigenvectors.

is costly, so that deflating a larger number of eigenvalues is efficient only if several system solves with the same overlap operator are to be performed. The figure shows that, irrespectively from the number of deflated eigenvalues, the preconditioned recursive method outperforms the unpreconditioned method in a similar way that it did in the non-recursive case considered before. When more smearing steps are applied, the improvement grows; improvement factors reach 10 or more. The figure also shows that, in the case that we have to solve only one or two linear systems with the same matrix, it is not advisable to use deflation at all, as the cost for the computation of the eigenvalues is simply too large. We attribute this finding at least partly to the fact that the thick restart method used to approximate the sign function from [43] is particularly efficient here. While all other data in Figure 4.7 were obtained for configuration 7, the rightmost data on the left plot refer to configuration 8. We see a similar efficiency for our preconditioner as we did for configuration 7 with 3 smearing steps, an observation consistent with the fact that configuration 8 was also obtained using 3 steps of (HEX) smearing; see Table 3.1.

Chapter 5

A multigrid accelerated eigensolver framework

As described in Section 2.4, obtaining eigenpairs of the Wilson-Dirac operator is an important computational task in lattice QCD, e.g., for the computation of energy levels of particles, or to improve statistical processes, cf. Section 3.4.2. In most circumstances we are interested in a small to moderate amount of eigenvectors corresponding to the eigenvalues closest to zero, especially for the Hermitian Wilson-Dirac operator. As the Hermitian Wilson-Dirac operator is indefinite, these eigenvalues are located in the *interior* of the spectrum.

Typically, computing interior eigenvalues is particularly expensive, which is why in this chapter we derive an efficient computational method for the special case of the Hermitian Wilson-Dirac operator. Section 2.4 introduced the *shift-and-invert* algorithms which extend the basic inverse iteration approach; cf. [90] as the most prominent methods for obtaining interior eigenvalues. This included the classical Rayleigh quotient iteration (RQI) [90] and the generalized Davidson (GD) method [90]. For the GD method numerous variations like GD + k [98], Jacobi-Davidson (JD) [94] or JDCG/JDQMR [80, 98] exist. This chapter introduces an eigensolver framework which is based on the generalized Davidson method, termed GD-λAMG (**G**eneralized **D**avidson with **A**lgebraic **M**ulti**G**rid.[1] We choose the DD-αAMG method to solve the correction equation Eq. (2.24) $(A - \tau I)t = r$, since it is one of the most efficient methods for solving linear systems of equations involving the Wilson-Dirac operator, cf. Section 2.3.3. Even though multigrid methods are clearly favored over plain Krylov subspace methods, their use in this framework requires some modifications to both the eigensolver method and the

[1] The λ is the most common symbol for denoting an eigenvalue and is reminiscent to the α of the DD-αAMG method.

multigrid method in order to achieve an overall efficient algorithm. This work has been submitted for publication and is also available as a preprint [44]. The following presentation is based on this publication.

5.1 The GD-λAMG method

The method we propose for the Hermitian Wilson-Dirac operator is based on Algorithm 2.10 but incorporates several adaptations for the Hermitian Wilson-Dirac operator and the underlying DD-αAMG multigrid solver, which is originally designed for the non-Hermitian Wilson-Dirac operator D.

The first challenge is the adaptation of DD-αAMG to the Hermitian Wilson-Dirac operator Q. We need to define appropriate methods for the smoother and the coarse grid correction, which will work for Q as well. As is discussed in [3, 17], the algebraic multigrid approach for D can be transferred to one for Q if the interpolation P preserves spin structure in the sense that on the coarse grid we can partition the degrees of freedom per grid point into two groups corresponding to different spins and that we have $\Gamma_5 P = P\Gamma_5^c$, where Γ_5^c is diagonal with values ± 1, depending on the spin on the coarse grid. Putting $Q_c = \Gamma_5^c D_c$ we then have

$$I - PQ_c^{-1}P^H Q = I - PD_c^{-1}\Gamma_5^c P^H \Gamma_5 D = I - PD_c^{-1}P^H D\,, \tag{5.1}$$

showing that the coarse grid error propagator for D is identical to the coarse grid error propagator for Q if we take the same P. The SAP smoothing, cf. Section 2.2, used in DD-αAMG is identical for D and Q, as well, as can be shown by the following argument. Mathematically, one step of SAP is a product of block projections, i.e., the error propagator is given by

$$E_{\text{SAP}} := \prod_{i=1}^{b}(I - \underbrace{I_{\mathcal{L}_i} Q_i^{-1} I_{\mathcal{L}_i}^H}_{:=M_{Q_i}} Q), \tag{5.2}$$

following the notation from Section 2.2.

Algorithmically, the calculations corresponding to $I_{\mathcal{L}_i} Q_i^{-1} I_{\mathcal{L}_i}^H Q$ can be performed in parallel for all blocks i of the same color if we introduce a red-back ordering on the blocks.

With this we get the following proposition, in which we define M_{D_i} and D_i analogously to M_{Q_i} and Q_i.

Proposition 5.1.
The error propagator $E_{\text{SAP}}(Q) := \prod_{i=1}^{b}(I - M_{Q_i}Q)$ *is identical to* $E_{\text{SAP}}(D) := \prod_{i=1}^{b}(I - M_{D_i}D)$.

Proof. We first note that Γ_5 is just a local positive or negative identity, so its block restriction $\Gamma_5^i := I_{\mathcal{L}_i}^H \Gamma_5 I_{\mathcal{L}_i}$ on $\mathcal{L}_i$ not only satisfies $I_{\mathcal{L}i}^H \Gamma_5 = \Gamma_5^i I_{\mathcal{L}_i}^H$ but also $(\Gamma_5^i)^{-1} = \Gamma_5^i$. To prove the proposition we only need to show that the error propagators are identical for any given subdomain i:

$$\begin{aligned} I - M_{Q_i} Q &= I - (I_{\mathcal{L}_i} Q_i^{-1} I_{\mathcal{L}_i}^H) Q \\ &= I - (I_{\mathcal{L}_i} (I_{\mathcal{L}_i}^H \Gamma_5 D I_{\mathcal{L}_i})^{-1} I_{\mathcal{L}_i}^H) \Gamma_5 D \\ &= I - (I_{\mathcal{L}_i} (\Gamma_5^i D_i)^{-1} I_{\mathcal{L}_i}^H) \Gamma_5 D \\ &= I - (I_{\mathcal{L}_i} D_i^{-1} \Gamma_5^i I_{\mathcal{L}_i}^H) \Gamma_5 D \\ &= I - (I_{\mathcal{L}_i} D_i^{-1} I_{\mathcal{L}_i}^H \Gamma_5) \Gamma_5 D = I - M_{D_i} D. \end{aligned} \tag{5.3}$$

□

Proposition 5.1 states that SAP for Q is equivalent to SAP for D if the block inversions for the block systems Q_i are performed exactly, which together with (5.1) implies that the DD-αAMG method has the same error propagator, irrespective of whether it is applied to Q or to D. As observed in [46] SAP smoothing works well for the standard Wilson-Dirac operator D thus it also works well for the Hermitian Wilson-Dirac operator Q. However, it is computationally more efficient to only approximate block inversions in SAP, and in this situation it becomes less clear which method is to be preferred over the other; see Section 5.3.

Alternatively, instead of SAP one can use (restarted) GMRES as a smoother for Q. For the non-Hermitian Wilson-Dirac operator D this is used in the multigrid methods from [3, 14, 82], and since GMRES is also one of the most numerically stable Krylov subspace methods for indefinite systems, it is to be expected to work well as a smoother in a multigrid method for Q as well. For GMRES smoothing a connection between Q and D similar to what has just been exposed for SAP smoothing does not hold. We compare the above options for the smoothing method experimentally in Section 5.3.

The next challenge is that we are confronted with a "maximally indefinite" interior eigenvalue problem, seeking the eigenvalues closest to zero, while the operator has a nearly equal amount of positive and negative eigenvalues. The basic generalized Davidson method uses the Rayleigh Ritz procedure to determine the Ritz approximation by solving the standard eigenvalue problem for $H = V_m^H A V_m$, cf. Definition 2.22. Ritz values approximate outer eigenvalues better and faster than the interior ones [90], which is why we use *harmonic Ritz values* [84] instead.

Definition 5.2 (harmonic Ritz values).
A value $\theta \in \mathbb{C}$ is called a harmonic Ritz value *of A with respect to a linear subspace $\mathcal{V}$ if θ^{-1} is a Ritz value of A^{-1} with respect to $\mathcal{V}$.*

As the exterior eigenvalues of A^{-1} are the inverses of the eigenvalues of A of small modulus, harmonic Ritz values tend to approximate small eigenvalues well. Inverting A to obtain harmonic Ritz values can be avoided with an appropriate choice for $\mathcal{V}$ as stated in the following theorem; cf. [94].

Theorem 5.3.
Let $\mathcal{V}$ be some m-dimensional subspace with basis $v_1, \ldots, v_m$. A value $\theta \in \mathbb{C}$ is a harmonic Ritz value of A with respect to the subspace $\mathcal{W} := A\mathcal{V}$ if and only if

$$Au_m - \theta u_m \perp A\mathcal{V} \text{ for some } u_m \in \mathcal{V}, u_m \neq 0. \tag{5.4}$$

With

$$V_m := [v_1|\ldots|v_m],\ W_m := AV_m \text{ and } H_m := (W_m^H V_m)^{-1} W_m^H A V_m,$$

(5.4) *is equivalent to*

$$H_m s = \theta s \text{ for some } s \in \mathbb{C}^m, s \neq 0 \text{ and } u_m = V_m s.$$

Due to Theorem 5.3, we can obtain harmonic Ritz values by solving the generalized eigenvalue problem

$$W_m^H V_m u = \frac{1}{\theta} W_m^H W_m u. \tag{5.5}$$

The computational overhead compared to the standard Ritz procedure is dominated by m^2 additional inner products to build $W_m^H A V_m$. In our numerical tests, we have observed that this is compensated by a faster convergence of the generalized Davidson method, cf. Section 5.3.

Although the multigrid approach is viable for the Hermitian Wilson-Dirac operator Q, it is, in practice, slower than for D. For exact solves of the subdomain systems in the SAP smoother, Equation (5.1) and Proposition 5.1 implies that the convergence speeds for Q and for D are comparable as the error propagation operators are identical. Though, in computational practice, it is more efficient to do only approximate solves for the subdomain systems, using a small number of GMRES steps, for example. In this scenario the multigrid method becomes significantly slower when used for Q rather than D, see Figure 5.4 in Chapter 5.3. This slowdown can be counteracted by left-preconditioning the correction equation with Γ_5. This means that instead of solving (2.24) with Q, we can transform it equivalently according to

$$\begin{aligned} & (Q - \tau I)t &= r \qquad (5.6)\\ \iff\ & \Gamma_5(Q - \tau I)t &= \Gamma_5 r \\ \iff\ & (D - \tau \Gamma_5)t &= \Gamma_5 r. \qquad (5.7)\end{aligned}$$

The spectrum of the resulting operator $\Gamma_5 Q(\tau) := D - \tau\Gamma_5$ has similarities to that of D with some eigenvalues collapsing on the real axis. As we will see in Chapter 5.3, this simple transformation speeds up the multigrid method significantly. For reference, Figure 5.1 shows full spectra of D and $\Gamma_5 Q(\tau)$ for a configuration on a 4^4 lattice.

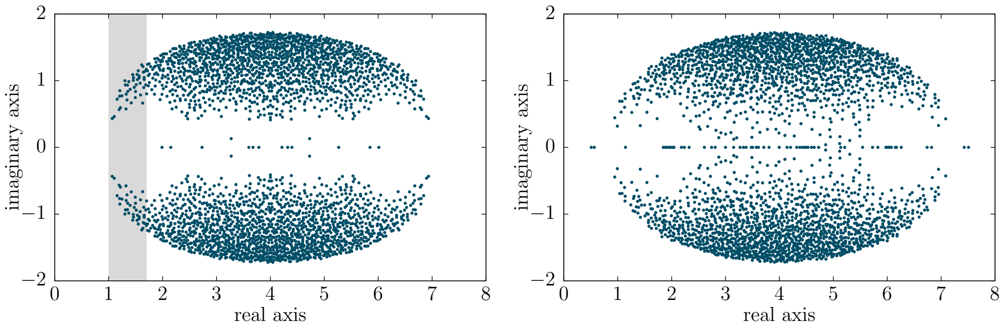


Figure 5.1: Full Spectra of D and $\Gamma_5 Q(\tau)$ for configuration 1 (see Table 3.1).

As the search space grows in every outer iteration, the storage and orthogonalization costs of the outer iteration in a generalized Davidson method eventually become prohibitively large. The following techniques reduce these costs in order to achieve a near-linear scaling in the number of computed eigenpairs. The first technique is *thick restarting* [94]. When the search space reaches a size of m_{max}, we perform a restart, similar to the one described in Section 2.2.3. However, instead of keeping only one vector, we keep the first m_{min} smallest non-converged harmonic Ritz vectors and use them to span the search space at the beginning of the next restart cycle. Using more than one vector for the next restart cycle is favorable due to the fact that, as already seen at the beginning of Section 2.4.2, the search space typically contains good approximations to more than one eigenpair, thus by carrying over more than one vector we retain more useful information and improve convergence at minimal cost.

The parameters m_{min} and m_{max} have to be chosen such that it is reasonable to assume that both positive and negative harmonic Ritz values are retained within the new search space. This way the eigensolver obtains a (nearly) equal amount of positive and negative eigenpairs in a uniform way.

With restarting, another major challenge arises. By dropping the already converged eigenvectors form the search space, it becomes mandatory to find an efficient way to avoid re-targeting those eigenpairs already found. As stated in Section 2.4.2, the approach of keeping all converged eigenvectors in the search space, even after a restart, becomes infeasible for a larger amount of sought eigenpairs k, since the maximum size of the search space depends on k, which significantly impacts the scaling behavior in this case. To remedy this, we employ the concept of *locking* converged eigenpairs [99] as a second technique. Locking keeps the search space $\mathcal{V}$ orthogonal to the space of already converged eigenvectors $\mathcal{X}$. In this manner, it is not required to keep converged eigenvectors in the search space which has the effect that the search space dimension becomes bounded independently of the number of eigenpairs sought. This in turn bounds the cost for computing the harmonic Ritz pairs. The new search direction still has to be orthogonalized against all previous eigenvectors, which leads to costs of order $\mathcal{O}(nk^2)$, since it consists of $k-1$ inner products for each of the k eigenpairs. This is responsible for the fact that, in principle, the cost of our method scales superlinearly with k, and this becomes visible when k becomes sufficiently large.

5.2 Local coherence

The strength of algebraic multigrid methods relies on an effective coarse grid correction step and thus on the construction of the interpolation operator P. As discussed in Section 2.3.3, the methods in use for the Wilson-Dirac operator are all adaptive: They require a setup phase which computes “test vectors” $w_i, i = 1, \ldots, n_{tv}$ which are collected as columns in the matrix $W = [w_1 \mid \ldots \mid w_{n_{tv}}]$. Using these test vector the setup phase constructs the interpolation operator P. By construction, the range of P contains *at least* the range spanned by the test vectors it is being built from. In [68] it has been observed that eigenvectors belonging to small eigenvalues of the Wilson-Dirac operator D are *locally coherent* in the sense that these eigenvectors are locally similar, i.e., they are similar on the individual aggregates. This is the reason why the span of an aggregation based interpolation P contains good approximations to small eigenpairs *far beyond* those which are explicitly used for its construction. This in turn explains the efficiency of such P in the multigrid method.

We can study local coherence using the *local coherence measure* lc of a vector v defined as

$$\mathrm{lc}(v) = \|\Pi v\|/\|v\|,$$

where Π denotes the orthogonal projection on the range of P. If $\mathrm{lc}(v)$ is close to 1, there is a good approximation to v in the range of P, implying that the multigrid coarse grid correction reduces error components in the direction of v almost to zero.

Figure 5.2 gives values for $\mathrm{lc}(v)$ for the Wilson-Dirac operator D and the corresponding Hermitian Wilson-Dirac operator Q on a 4^4 lattice. Since this lattice is so small, we can compute the full spectrum ($12 \cdot 4^4 = 3072$ eigenpairs) of both D and Q. For each matrix we then consider a partitioning of the eigenvectors into 128 sets, each set consisting of 24 consecutive eigenpairs. Here, “consecutive”

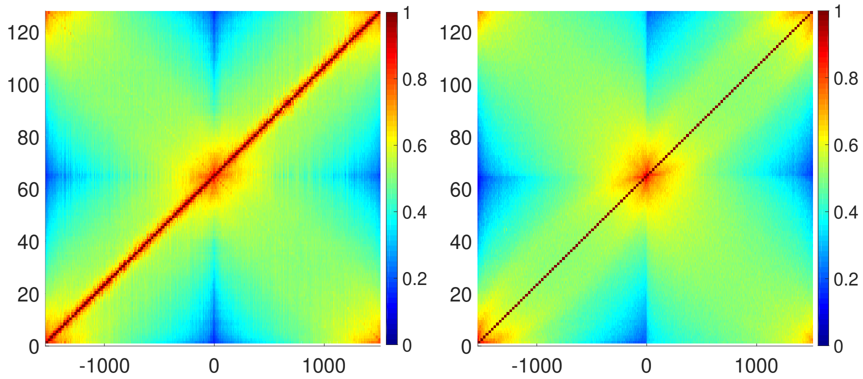


Figure 5.2: Local coherence for D (left) and Q (right) for a configuration 1, cf. Table 3.1.

refers to an ordering based on the modulus or the sign of the real part, respectively; see the next paragraph for details. For each of these "interpolation sets", the corresponding row displays the color coded value of $\mathrm{lc}(v)$ when projecting an eigenvector v with the projection Π corresponding to the aggregation-based interpolation P built with the eigenvectors from that interpolation set as test vectors. The aggregates used were based on a decomposition of the 4^4 lattice into 16 sublattices of size 2^4. Due to the spin structure preserving approach, we have two aggregates per sublattice, each built from the corresponding spin components of the 24 test vectors[2]. Of course $\mathrm{lc}(v) = 1$ (dark red) if v is from the respective interpolation set.

The numbering of the eigenvalues used in these plots is as follows: The plot for D has its eigenvalues with negative imaginary part in its left half, ordered by descending modulus and enumerated by increasing, negative indices including zero, $-1\,535, \ldots, 0$. The eigenvalues with positive imaginary part are located in the right half, ordered by ascending modulus and enumerated with increasing positive indices $1, \ldots, 1536$. For Q we just order the real eigenvalues by the natural ordering on the reals, using again negative and positive indices. Thus, for D as for Q, eigenvalues small in modulus are in the center and their indices are small in modulus, while eigenvalues with large modulus appear at the left and right ends, and their indices are large in modulus.

Although one must be careful when drawing conclusions from extremely small configurations, Figure 5.2 illustrates two important phenomena. Firstly, local coherence appears for both D and Q, but it is more pronounced for the non-Hermitian Wilson-Dirac matrix. This especially holds directly next to the interpolation sets (the diagonal in the plots). Secondly, local coherence is particularly

[2]The projection Π therefore projects onto a subspace of dimension $24 \cdot 2 \cdot 16 = 768$. If there were no local coherence at all, the expected value of lc is thus $768/(12 \cdot 4^4) = 0.25$.

strong and far-reaching when projecting on the interpolation sets corresponding to the smallest and largest eigenpairs in absolute values. In the center of both plots, we observe a star-shaped area with particularly high local coherence. This area corresponds to around 10% of the smallest eigenvalues. To a lesser extent, local coherence is also noticeable for the other parts of the spectrum, as we consistently observe higher values for $\mathrm{lc}(v)$ for eigenvectors close to the respective interpolation set.

The right part of Figure 5.3 reports similar information for the Hermitian Wilson-Dirac operator Q coming from a larger, realistic configuration on a 64×32^3 lattice. For lattices of this size we cannot compute the full spectrum, thus we show the values for the 984 smallest eigenpairs, subdivided in 41 interpolation sets, each consisting of 24 consecutive eigenpairs. The aggregates were this time obtained from 4^4 sublattices. For comparison, the left part of the figure shows a zoomed-in part of the local coherence plot for Q for the 4^4-lattice from Figure 5.2.

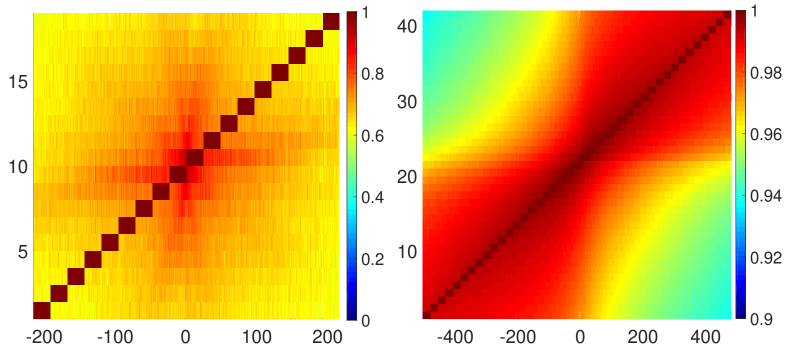


Figure 5.3: Local coherence for Q for different lattices, focusing on the eigenpairs closest to zero. *Left:* 432 eigenpairs of configuration 1. *Right:* 984 eigenpairs of configuration 4.

First note that the colors encode different values in the left and right part of Figure 5.3. Local coherence does not drop below 0.9 for the large configuration, while for the small configuration it goes down to 0.6. On the other hand, 984 eigenvalues only correspond to a minuscule fraction of roughly $4 \cdot 10^{-3}\%$ of the total of $12 \cdot 32^3 \cdot 64 = 25\,165\,824$ eigenvalues, which is much less than the roughly 10% depicted for the small configuration. For the interpolation operator of the large configuration we used aggregates corresponding to 4^4 sublattices, which give a total of $2 \times 2^{13} = 16\,384$ aggregates. In relative terms, this is several orders of magnitude finer as for the 4^4 lattice. This finer aggregation leads to interpolation operators with increased faculties to recombine information, which explains the resulting higher local coherence.

Both parts of Figure 5.3 show that local coherence drops off for eigenvectors farther away from the interpolation set. For the large configuration, we see, for example, that local coherence of the vectors from the last interpolation set with the second-to-last interpolation set is very high as indicated by the deep red color in the top right corner of the plot. The local coherence of these vectors with respect to the central interpolation set (which contains the eigenpairs with eigenvalues closest to 0) is significantly smaller, indicated by the yellow color at the middle of the right-hand boundary of the plot. In the scenario where we choose the shift τ in the correction equation (5.7) farther away from zero—as we are targeting eigenpairs close to τ—a coarse grid operator constructed using the eigenpairs closest to zero thus becomes increasingly less effective, reducing the overall convergence speed of the multigrid method significantly. To remedy this, we propose a dynamical interpolation updating approach, resulting in a coarse grid operator that remains effective on the span of the eigenvectors with eigenvalues close to the value of τ set in the outer iteration of the generalized Davidson method. In the course of the outer iteration, once enough eigenpairs are available, we therefore rebuild the interpolation, and with it the coarse grid operator, using the already converged eigenvectors which are *closest* to the currently targeted harmonic Ritz value. Once a harmonic Ritz value converged to an eigenvalue and we choose a new target value τ that has the same sign as the previous target, we replace one eigenvector from the interpolation set—the one farthest away from τ—with the newly converged eigenvector, and update the multigrid hierarchy. If the new τ has its sign opposite to the previous one we replace the full interpolation set with converged eigenvectors closest to the new τ, and again update the multigrid hierarchy. The updates of the interpolation and coarse grid operators involve some data movement and local operations, but their cost is minor compared to the cost of the other parts of the computation.

With this approach, the coarse grid is always able to treat the eigenspace closest to our current harmonic Ritz approximation efficiently and makes optimal use of the existing local coherence. This results in a significantly faster multigrid solver when larger shifts are used, i.e., when a large number of small eigenpairs has to be computed. Since the solution of these shifted systems accounts for most of the work in the eigensolver this approach improves the eigenvalue scaling to a nearly linear complexity, as seen in Section 5.3.

A summary of our eigensolver, termed GD-λAMG, a generalized Davidson method with algebraic multigrid acceleration, is given as Algorithm 5.1.

5.3 Numerical results

In this section we present a variety of numerical tests to analyze the efficiency of the GD-λAMG eigensolver.

Algorithm 5.1: GD-λAMG

input: Hermitian Dirac operator Q, no. of eigenvalues n, no. of test vectors n_{tv}, min. and max. subspace size m_{min} and m_{max}, initial guess $[v_1, \ldots, v_{n_{tv}}, t] =: [V|t]$, desired accuracy ε_{outer}

output: set of n eigenpairs (Λ, X)

$\Lambda = \emptyset$, $X = \emptyset$
for $m = n_{tv} + 1, n_{tv} + 2, \ldots$
    $t = (I - VV^H)t$, $t = (I - XX^H)t$
    $v_m = t/||t||_2$
    $V = [V|v_m]$
    get all (θ_i, s_i) with $(QV)^H(QV)s_i = \theta_i(QV)^H V s_i$
    find smallest (in modulus) $\theta_i \notin \Lambda$
    $u = Vs_i$, $r = Qu - \theta_i u$
    if $||r||_2 \leq \varepsilon_{outer}$ **then**
        //current eigenpair has converged
        $\Lambda = [\Lambda, \theta_i]$, $X = [X, u]$
        update smallest (in modulus) $\theta_i \notin \Lambda$
        $u = Vs_i$, $r = Qu - \theta_i u$
        rebuild interpolation using the n_{tv} eigenvectors x_j with eigenvalue λ_j closest to θ_i and update multigrid hierarchy
    //solve correction equation
    $t = \texttt{DD-}\alpha\texttt{AMG}((D - \theta_i\Gamma_5), \Gamma_5 r)$
    //restart
    if $m \geq m_{max}$ **then**
        get (Θ, S) as all eigenpairs (θ_i, s_i) of $(QV)^H(QV)s_i = \theta(QV)^H V s_i$
        sort (Θ, S) by ascending modulus of Θ
        for $i = 1, \ldots, m_{min}$
            $V_i = Vs_i$
            $(QV)_i = (QV)s_i$
        retain first m_{min} vectors of V and QV

Table 5.1 shows the default algorithmic parameter settings we used within GD-λAMG.

	parameter	symbol	default
DD-αAMG setup	number of test vectors	n_{tv}	24
	setup iterations		6
	(post-)smoothing steps		4
DD-αAMG solve	relative residual	ε_{inner}	10^{-1}
	maximum iterations		5
	coarse grid tolerance		$5 \cdot 10^{-1}$
eigensolver method	relative eigenvector residual	ε_{outer}	10^{-8}
	number of eigenpairs		100
	minimum subspace size	m_{min}	30
	maximum subspace size	m_{max}	50

Table 5.1: List of algorithmic *default* parameters.

The numerical results involving configurations 3–6 were obtained on the JURECA and JUWELS clusters at the Jülich Supercomputing Centre [61, 63], while results involving the other lattices were obtained on a smaller workstation. We will compare our results with the state-of-the-art library PRIMME and with PARPACK, and we start by outlining their underlying basic algorithms.

PRIMME (PReconditioned Iterative MultiMethod Eigensolver) [100, 109] implements a broad framework for different Davidson-type eigensolvers. Its performance is best if it is given an efficient routine to solve linear systems with the matrix A, and we do so by providing the Γ_5-preconditioned DD-αAMG solver. There are two key differences compared to GD-λAMG:

- The interpolation cannot be updated efficiently within the PRIMME framework (at least not without expert knowledge on the underlying data structures), hence we do not update it for this method.
- PRIMME uses a Rayleigh Ritz instead of a harmonic Ritz approach to extract eigenvalue approximations.

PRIMME has a fairly fine-tuned default parameter set, e.g., for subspace size or restart values, and is able to dynamically change the eigensolver method. We keep the default settings and provide the same multigrid solver to PRIMME as we do for GD-λAMG.

PARPACK (Parallel ARnoldi PACKage) [97] is a somewhat older but widely used software for the computation of eigenvalues of large sparse matrices. It is based on an implicitly restarted Arnoldi method, which is originally designed to find extremal eigenvalues. It is possible to transform an interior problem into an exterior one using a filter polynomial, i.e., a polynomial which is large on the k interior eigenvalues we are looking for and small on the remaining ones. To construct such a polynomial, for example as a Chebyshev polynomial, we need information on the eigenvalue λ_{max} which is largest in modulus and the $(k+1)$st smallest in modulus, λ_{k+1}. While $\lambda_{max} = 8$ is a sufficiently good estimate for the Hermitian Wilson-Dirac matrix Q, no a-priori guess for λ_{k+1} is available in realistic scenarios. For our tests, we run one of the other methods to compute the first k eigenvalues and then use a slightly larger value as a guess for λ_{k+1}. While this approach obviously costs a lot of additional work and actually makes the subsequent Arnoldi method obsolete, it is a good reference for a *near-optimally* polynomially filtered Arnoldi method. Since this approach does not require inversions of the matrix Q, the parameter set for this method is rather small. We use a degree ten Chebyshev polynomial as the filter polynomial and set the maximum subspace size to be twice the number of sought eigenpairs. The required eigenvector residual is set to 10^{-8}, as with the other methods.

5.3.1 Algorithmic tuning

Solving the correction equation. Each step of Algorithm 5.1 uses DD-αAMG in line 14 to solve the Γ_5-preconditioned correction equation $(D - \tau\Gamma_5)t = \Gamma_5 r$. More precisely, as indicated by the parameters given in the middle of Table 5.1, we stop the outer (FGMRES) iteration of DD-αAMG once the initial residual norm is reduced by a factor of 0.1 or a maximum of 5 iterations is achieved. Within each DD-αAMG iteration we require a reduction of the residual by a factor of 0.5 when solving the system on the coarsest level. Table 5.2 shows that the Γ_5-preconditioning yields indeed significant gains in compute time.

correction equation	iterations		Time
	outer	inner	in core-h.
Eq. (5.6): $(Q - \tau I)t = r$	565	10,349	83.0
Eq. (5.7): $(D - \tau\Gamma_5)t = \Gamma_5 r$	511	3,045	41.3

Table 5.2: Impact of Γ_5-preconditioning for the computation of 100 eigenpairs of configuration 3 (see Table 3.1).

A variant of generalized Davidson methods solves, instead of the correction equation (2.24), the Jacobi-Davidson projected [94] system $(I-uu^H)(A-\theta I)(I-uu^H)$, where u is the last (harmonic) Ritz vector approximation, which was used to compute the residual $r = Au - \theta u$. This will avoid stagnation in the case that the correction equation is solved *too exactly*. There are theoretically justified approaches which adaptively determine how accurately the projected system should be solved in each iteration. Since we solve the correction equation to quite low relative precision (10^{-1} only), we could not see a benefit from using the Jacobi-Davidson projected system. Indeed, even with the adaptive stopping criterion, this approach increased the compute time by approximately 15% for our implementation.

Impact of the smoother The original DD-αAMG method uses SAP as a smoother and we have shown in Section 5.1 that SAP is also applicable for the Hermitian Wilson-Dirac operator Q, yielding the same error propagation operator as long as the individual block systems are solved exactly. We now compare a cost-efficient, approximate SAP and GMRES as smoothers within the multigrid methods constructed for the matrices $D - \tau I$, $Q - \tau I$ and $D - \tau\Gamma_5$, where τ ranges from 0 to 0.5 for configuration 2. Note that $D - \tau I$ is not relevant for this work, since it would arise when computing eigenpairs for D. We still include the results here to be able to compare the performance of DD-αAMG for $Q - \tau I$ and $D - \tau\Gamma_5$ with the performance of DD-αAMG for $D - \tau I$.

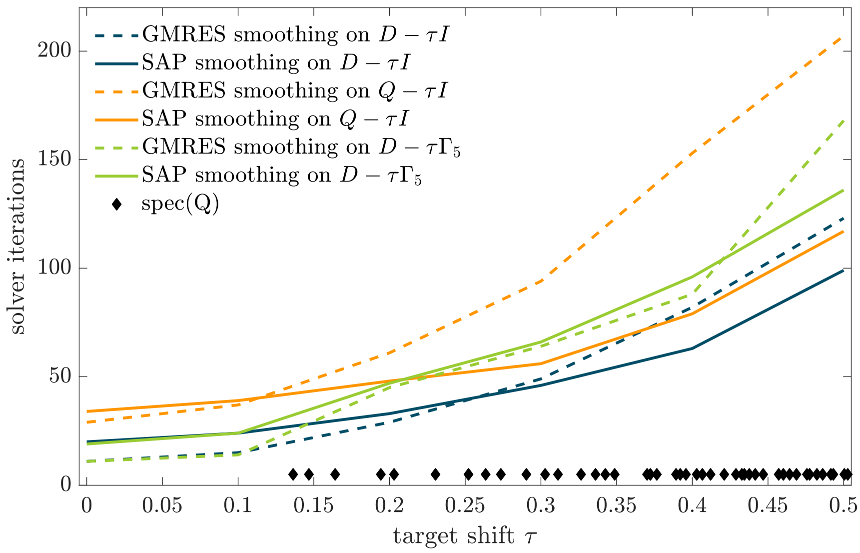


Figure 5.4: Comparison of iteration counts of the DD-αAMG method using either SAP or GMRES smoothing for configuration 2 and increasing target shifts τ. The black diamonds at the bottom depict the eigenvalue distribution of Q.

Figure 5.4 shows a scaling plot with respect to the target shift τ for configuration 2. For this plot, we used a two-level DD-αAMG method with six steps of the adaptive setup procedure to generate the coarse grid system. The GMRES smoother stops iterating if a reduction of the residual by a factor of 10^{-1} is achieved or after a maximum of ten iterations have been performed. Similarly, the SAP smoother performs three sweeps of SAP, where each block solve is performed using GMRES until a reduction of the residual by a factor of 10^{-1} is achieved for the individual block or after a maximum of ten iterations have been performed. This way the computational work for both smoothers is roughly comparable.

Figure 5.4 verifies what was stated in Section 5.1, namely that DD-αAMG converges more slowly for Q compared to D. It also shows that Γ_5-preconditioning is beneficial in the case of GMRES smoothing whereas in the case of SAP smoothing, it loses efficiency compared to Q, although only by a small margin. Comparing the two smoothing methods for $D - \tau\Gamma_5$, we see that both methods perform nearly identical up to larger shifts, where SAP starts to be slightly more favorable. We do not expect this to be relevant for larger configurations, though, since there the spectrum is much more dense. Even when aiming for a large number of small eigenvalues, we certainly do not expect to end up with τ-values as large as 0.2 already. Since the focus for this work is on finding an efficient coarse grid operator, and not on optimizing the smoother, we stick to GMRES smoothing here. Implementing SAP instead of GMRES for $D - \tau\Gamma_5$ within the DD-αAMG framework would require a more substantial remodeling of the DD-αAMG code.

Impact of the coarse grid correction For an assessment of the impact of the coarse grid correction step we compute 100 eigenvalues for configuration 3, once using DD-αAMG with GMRES smoothing to solve the correction equation, and once with a modification where we turned-off the coarse grid correction. This yields a generalized Davidson method where the Γ_5-preconditioned correction equation (5.7) is solved using FGMRES with the GMRES-steps of the smoother as a non-stationary preconditioner, i.e., GMRESR, the recursive GMRES method [106]. Note that we do not yet include updating the multigrid hierarchy as the outer iterations proceeds.

The left part of Figure 5.5 shows the FGMRES iterations spent on the correction equation for computing 100 eigenvalues for the two variants. We see that right from the beginning, including the coarse grid correction, i.e., using the multigrid method, reduces the iteration count by one order of magnitude compared to the "pure" GMRESR-Krylov subspace method. The required number of FGMRES iterations per eigenvalue stays constant at ≈ 30 for the multigrid method, whereas GMRESR starts at ≈ 300 and increases to $\approx 1,200$ for the last eigenvalues. This is also reflected in CPU time, where on JUWELS multigrid preconditioning results in 30 core-h for the entire computation, whereas 217 core-h were necessary when using GMRESR. Thus multigrid gains one order of magnitude in compute time

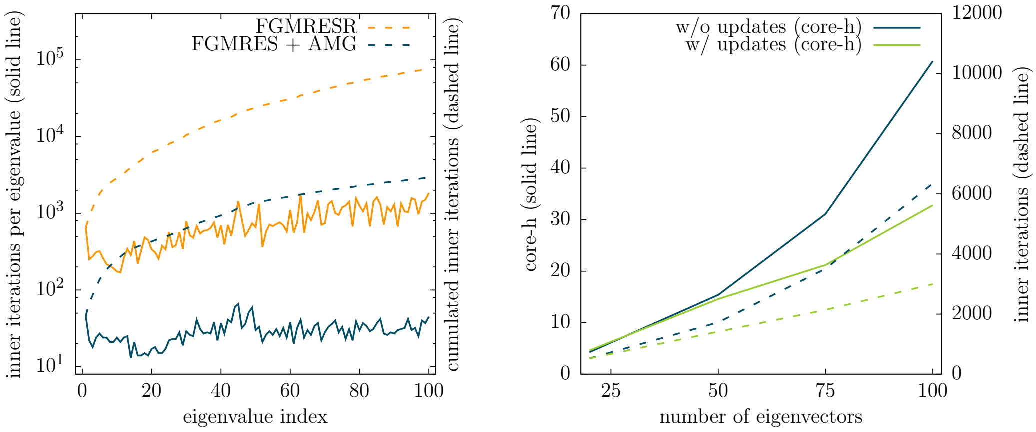


Figure 5.5: *Left:* Computation of 100 eigenvalues for configuration 3 with GMRESR and FGMRES + AMG. *Right:* Comparing eigenvalue scaling for configuration 4 depending on whether eigenvector information is provided for the interpolation operator.

and, in addition, shows an improved scaling behavior, despite the loss of local coherence for the larger eigenvalues.

The right part of Figure 5.5 now illustrates the additional benefits that we get from turning on the updating of the multigrid hierarchy, i.e., when performing full GD-λAMG as described in Algorithm 5.1. Both approaches perform similarly as long as a small amount of eigenvalues is sought. This changes substantially for already a moderate amount of eigenvalues to a point where interpolation updates save roughly a factor of two in both, number of iterations (dashed lines) and consumed core-h (solid lines). In terms of iterations it is also noteworthy that interpolation updates lead to a nearly linear scaling with respect to the eigenvalue count, whereas in the other case the scaling is closer to quadratic.

5.3.2 Scaling results

Scaling with the lattice size. We now compare GD-λAMG, PRIMME and PARPACK in terms of scaling with respect to the lattice size. For this, we report the total core-h consumed for computing 100 eigenpairs on configurations 3 to 6.

Figure 5.6 shows that PRIMME and GD-λAMG scale similarly with increasing lattice size. GD-λAMG shows some improvement in core-h compared to PRIMME, and this improvement tends to get larger when increasing the lattice size. The right part of Figure 5.6 shows, that this improvement might be partially attributed to the fact that we use a harmonic Ritz extraction. Here, we compare GD-λAMG with its default harmonic Ritz extraction to a variant where

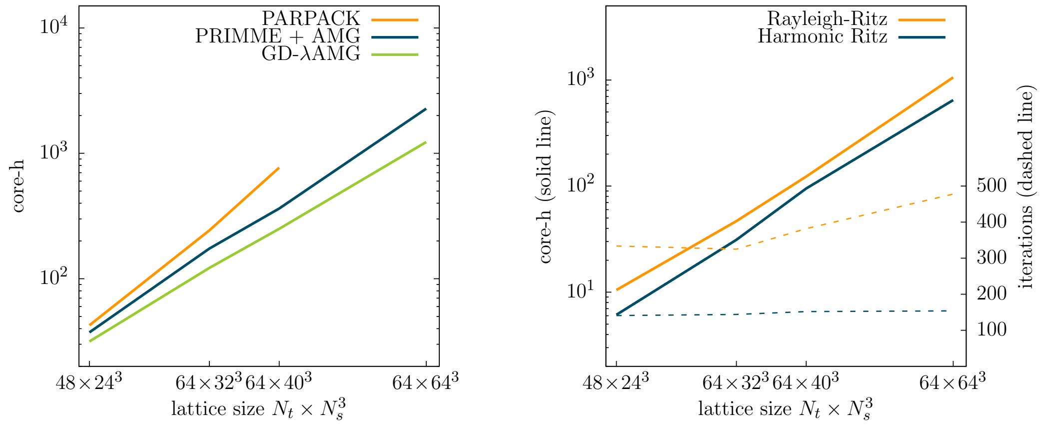


Figure 5.6: *Left:* Computation of 100 eigenvalues for 48×24^3 to 64×64^3 lattices for different methods. *Right:* Comparison of Ritz and harmonic Ritz eigenpair extraction for different lattice sizes.

we use the standard Rayleigh-Ritz extraction as is done in PRIMME. This figure shows that harmonic Ritz extractions result in substantially less inner iterations. It also yields savings in compute time, which are smaller, due to the additional cost for the inner products. Note that for larger lattices eigenvalues become more clustered. The harmonic Ritz extraction is then more favorable compared to the Rayleigh-Ritz approach, since it is able to better separate the target eigenvalue from the neighboring ones. PARPACK scales worse than the other methods, even when we use an unrealistic "near optimal" filter polynomial as we did here. In practice, i.e., when no guess for $|\lambda_{k+1}|$ is available, PARPACK's performance would fall even further behind. Applying PARPACK to Q^{-1} to make use of the efficient multigrid solver is way too costly, due to the necessity of accurate solves to maintain the Krylov structure.

Scaling with the number of eigenvalues. Figure 5.7 reports results of a scaling study obtained for configuration 4. We just compare GD-λAMG and PRIMME, since PARPACK is not competitive.

The figure shows that GD-λAMG has an advantage over PRIMME when larger numbers of eigenvalues are sought. GD-λAMG needs up to one order of magnitude less iterations, which translates to a speed-up of 1.5 for 50 eigenvalues to up to more than three for 1 000 eigenvalues. This shows that the additional effort due to the adaptive construction of the multigrid hierarchy and the harmonic Ritz extraction is beneficial with respect to the overall performance. GD-λAMG and PRIMME both scale nearly linearly with respect to the number of eigenvalues sought, up to at least 300 eigenvalues. Then PRIMME's performance starts to decrease more significantly compared to GD-λAMG. We see that the increase in

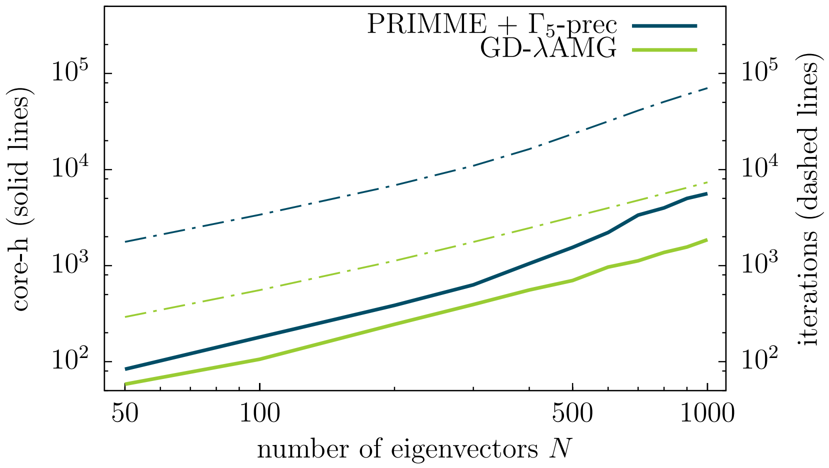


Figure 5.7: Eigenvalue scaling in the range of 50 to 1000 eigenpairs on configuration 4 with a lattice size of 64×32^3.

the overall computing time in PRIMME scales more than linearly with the number of iterations to be performed. This indicates that the non-adaptive multigrid solver used in PRIMME is getting increasingly less efficient, a situation that is remedied with the update strategy realized in GD-λAMG.

Chapter 6

A Davidson-type multigrid setup

In Chapter 5 we have established that generalized Davidson-type eigensolvers are among the most efficient algorithms for computing eigenpairs of the Hermitian Wilson-Dirac operator. Since the setup phase of an algebraic multigrid method oftentimes aims at computing approximate eigenvectors, it is thus natural to assume that a multigrid setup based on this type of eigensolvers will be efficient as well. In this chapter we introduce a Davidson-type multigrid setup based on the eigensolver from Chapter 5. Afterwards we discuss first results of this new setup procedure and compare it to the current setup, which is based on inverse iteration, cf. Section 2.3.3. In the following we will refer to the inverse iteration based setup procedure as *simple* setup.

6.1 Subspace acceleration in algebraic multigrid setup

Most current algebraic multigrid methods, like the Inexact Deflation method, the AMG method and the DD-αAMG method are based on finding approximations to the small eigenvectors of D using simple and cheap vector iteration methods, see e.g., Algorithm 2.7. There, each vector is iterated separately with some orthogonalization in order to ensure convergence towards different eigenvectors.

This approach however does not make optimal use of the available generated information. In particular each test vector starts with a random initial guess, while good initial guesses for each eigenvector (except for the first one) can be extracted using the approximation of the previous eigenvector, as has already been shown in Figure 2.7.

In this section we motivate how the GD-λAMG eigensolver can be applied as an efficient setup routine for an algebraic multigrid method. First note that GD-λAMG computes eigenvalues of Q rather than D, but simple algebra shows that the eigenvalue decomposition of Q can be transformed to a singular value decomposition of D. Its proof is similar to the proof of the first part of Lemma 4.6.

$$\begin{aligned} Q &= V\Lambda V^* \\ \Leftrightarrow \Gamma_5 Q &= \Gamma_5 V \operatorname{sign}(\Lambda)|\Lambda|V^* \\ \Leftrightarrow D &= U\Sigma V^* \end{aligned} \tag{6.1}$$

Since Γ_5, V and $\operatorname{sign}(\Lambda)$ are unitary, $U := \Gamma_5 V \operatorname{sign}(\Lambda)$ is unitary as well. The diagonal matrix $\Sigma := |\Lambda|$ has non-negative entries, thus $U\Sigma V^*$ defines a singular value decomposition of D, and we can use the eigenvectors of Q to obtain singular vectors of D.

Constructing the intergrid operators within the multigrid setup from singular vectors is a viable alternative to eigenvectors as they also inherit useful information about the matrix. A comparison between these two approaches will be given in Section 6.2.

The application of GD-λAMG within the context of a setup procedure is rather straightforward, but several algorithmic aspects which are responsible for the improved eigenvalue scaling, become obsolete. Locking and shifted inversions are not required since we are only interested in $\mathcal{O}(20)$ eigenvector approximations, whereas (thick) restarting and (non-shifted) inversions remain core parts of the modified algorithm. Analogous to the simple setup, our new *Davidson* setup is comprised of an initial phase and an iterative bootstrap phase.

In the initial phase we apply ℓ sweeps of n_{tv} Davidson steps, where n_{tv} is the number of test vectors and ℓ is a tunable parameter. Starting with an empty set of test vectors, the n_{tv} vectors from the first sweep are added to the subspace and form the first candidate set for the test vectors. After the next sweep the subspace is comprised of $2 \cdot n_{tv}$, where the new vectors come from n_{tv} additional Davidson steps. At this point a thick restart is applied, extracting the n_{tv} best eigenvector approximations within this subspace. Overall, this procedure is repeated ℓ times and the final n_{tv} eigenvector approximations are used as test vectors for the definition of the intergrid operators. With the next coarser grid defined, the initial phase can be recursively applied to the next level, until the full multigrid hierarchy is defined. The (non-shifted) inversions required within this phase are performed very approximately using the smoothing method prescribed by the multigrid method, e.g., SAP or GMRES.

After the initial setup, the first multigrid hierarchy is defined, and it is then employed in the iterative phase to further improve the test vectors. The iterative phase closely follows the procedure from the initial setup by repeatedly expanding the subspace to $2 \cdot n_{tv}$ vectors before applying a thick restart to extract the n_{tv} best eigenvector approximations. After each restart, the iterative phase is applied recursively to the next coarser grid. In the iterative phase, we thus benefit from a more effective solver by using the full multigrid method, resulting in an improved convergence behavior.

Algorithm 6.1 briefly sketches the new setup strategy. In this algorithm, one Davidson step refers to one `for`-loop of the generalized Davidson method, cf. Algorithm 2.10, while the thick restarting procedure follows the one described in Algorithm 5.1.

Algorithm 6.1: Davidson setup

input: number of initial Davidson sweeps ℓ, number of iterative phases k
output: Intergrid operators $P = R^H$ and coarse grid operator D_c
initialize random initial guess $W = [w_0]$
// Initial phase: inversions performed by smoother
for $j = 1, \dots, \ell$
  Extend W by n_{tv} Davidson steps
  Thick restart to extract n_{tv} test vectors, gathered as columns of W
construct P and D_c from W
perform initial phase for D_c
// Iterative phase: inversions performed by full AMG method
for $i = 0, \dots, k$
  Extend W by n_{tv} Davidson steps
  Thick restart to extract n_{tv} test vectors, gathered as columns of W
  update P and D_c
  perform iterative phase for D_c

In addition to the Davidson setup we also formulate a *hybrid* approach where the initial phase is performed by the Davidson setup, whereas for the iterative phase we switch to the simple setup. Even though these two methods aim at approximating different types of vectors, i.e., singular vectors and eigenvectors respectively, we will see in the following section that this approach is a competitive alternative to both methods in certain scenarios.

6.2 Numerical results

In this section we present numerical results where we compare the three different setup approaches. First, we investigate whether in the setup phase approximations of the eigenvectors of D or approximations of the singular vectors of

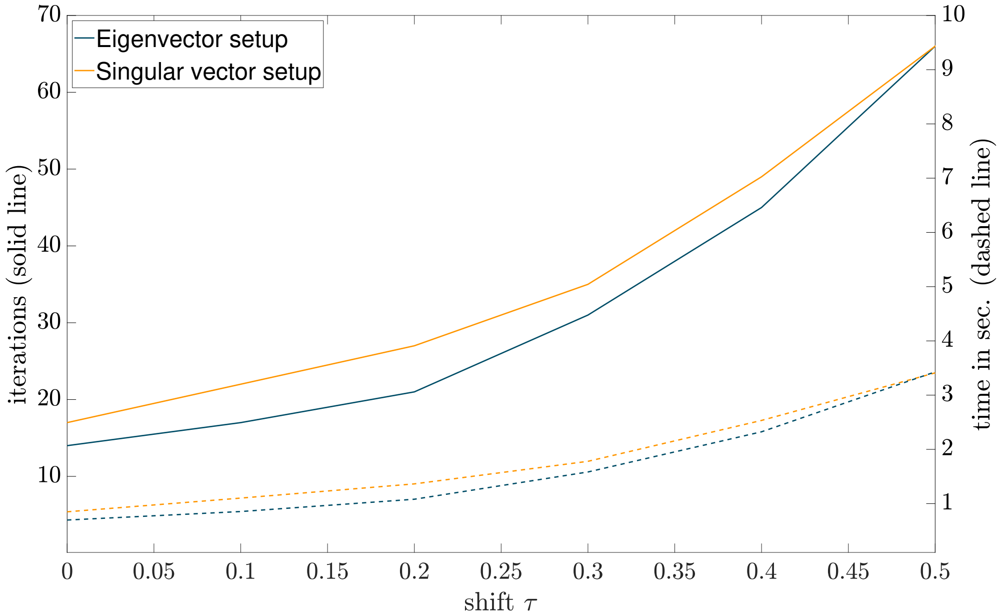


Figure 6.1: Iteration count and compute time for the solution of $(D - \tau I)x = b$ with the DD-αAMG method using eigenvector and singular vector approximations during the setup phase. These results were generated using Configuration 1 from Table 3.1.

D—which correspond to eigenvectors of Q—are more beneficial for the overall convergence of the DD-αAMG method.

Figure 6.1 shows convergence results for increasingly ill-conditioned systems $D - \tau I$ with configuration 1 using a two-level DD-αAMG method. For the setup parameters, we use 20 test vectors, one step of the initial simple setup, three steps of the iterative simple setup and four steps of multiplicative SAP as a post-smoother. We use the simple setup for both approaches to have a fair comparison between the effectiveness of eigenvectors and singular vectors for the construction of the intergrid operators.

This graph shows that eigenvectors are slightly favored over singular vectors as candidates for test vectors by about 10%, independent of the conditioning of the problem. This difference vanishes for larger shifts, but this is mostly not relevant in practice, since those shifts are not required for physical simulations.

Table 6.2 shows the iteration count of DD-αAMG as well as the execution time for the solve itself, the setup (where we separate between the time spent in the initial setup and the iterative part) and the overall run time for the whole method. The $+k$ in the setup type column indicates the number of iterative setup steps performed for the respective method. We use a physically relevant 64^4 configuration, i.e., configuration 9 from Table 3.1.

setup type	solver		setup time		overall time
	iter.	time	initial	iterative	solve + setup
simple + 0	251	204s	9s	-	213s
Davidson + 0	33	58s	17s	-	75s
simple + 1	28	77s	9s	28s	114s
hybrid + 1	12	40s	17s	38s	95s
Davidson + 1	9	23s	16s	60s	99s
simple + 2	8	13s	9s	102s	124s
hybrid + 2	7	13s	17s	152s	182s
Davidson + 2	9	18s	17s	229s	264s
simple + 3	7	12s	9s	223s	244s
hybrid + 3	7	11s	17s	268s	296s
Davidson + 3	9	17s	16s	361s	394s
simple + 4	7	12s	9s	304s	325s
hybrid + 4	6	10s	17s	350s	377s
Davidson + 4	9	17s	16s	527s	560s
simple + 5	7	12s	9s	420s	441s
hybrid + 5	6	10s	17s	453s	480s
Davidson + 5	9	17s	16s	704s	737s

Figure 6.2: Comparison of the three different setup procedure for varying iterative setup steps denoted by the $+k$ in the first column.

We observe that the fastest overall time for the setup and one solve is achieved using the initial Davidson setup without any iterative phase (Davidson +0). It is about 50% faster compared to the fastest simple inverse iteration based setup, which uses one step of the iterative phase (simple +1). Davidson +0 leads to a faster solver as well as being cheaper compared to simple +1 even though the solver requires a few iterations more in this case.

In a scenario where multiple solves per configuration are required a faster solve time becomes more important compared to the overall time, which includes the setup time. In this case more investment into the setup is beneficial. As we see in Table 6.2, the solve times quickly decrease for all three methods once we employ one or more steps of the iterative phase. Especially the simple setup benefits from the iterative phase. For larger amounts of iterative steps the hybrid and the simple setup have comparable solver iterations, whereas the Davidson setup slightly falls back. This is attributed to the fact that the Davidson setup aims at eigenvectors of Q, which correspond to singular vectors of D, while the other methods approximate eigenvectors of D, showing again that an eigenvector based setup is preferred over a singular vector based setup. In any case, applying more than three steps of any setup procedure yields only minor improvements and is only recommended if an extremely large amount of solves per configuration is

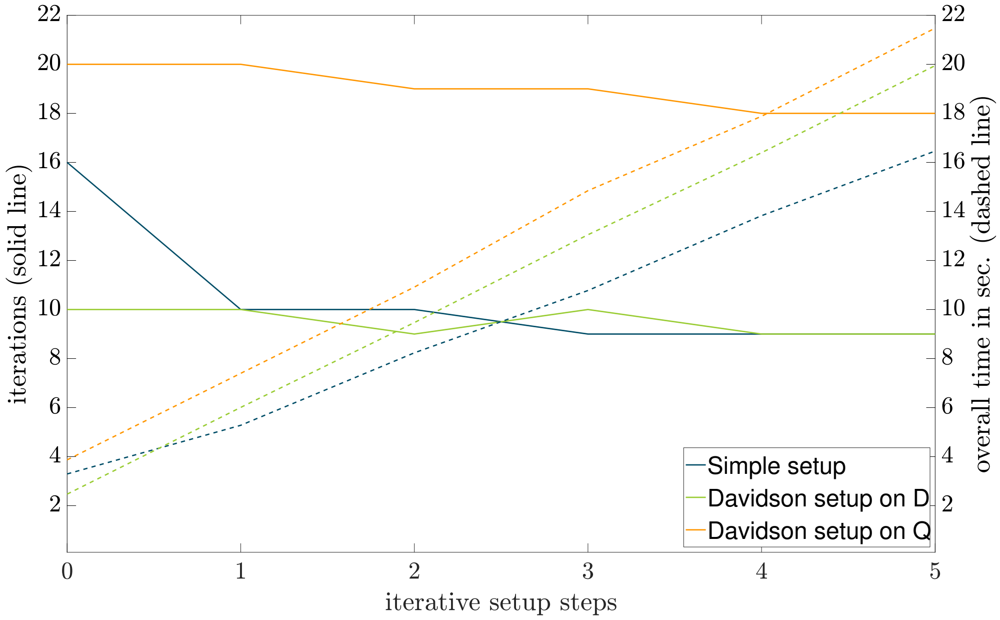


Figure 6.3: Iteration count and compute time for the solution of $Dx = b$ with the DD-αAMG method including the setup time for the simple setup and the Davidson setup for Q and for D. These results were generated using Configuration 1 from Table 3.1.

required.

In terms of overall time, it can also be observed that even though the Davidson setup is based on a Hermitian operator, the computational cost of its iterative part is significantly larger compared to the simple iterative setup, which is due to the Davidson setup involving more inner products and the solution of small dense eigenvalue problems.

Thus, overall, the choice of the most beneficial setup procedure depends on whether only one solve per operator is required or multiple ones. In the first case, e.g., within the HMC algorithm (cf. Section 3.4.3), a cheap setup procedure like Davidson +0 performs best, while for multiple solves, e.g., for eigensolvers like GD-λAMG, more investment into a simple or hybrid setup is recommended, depending on how many solves are required.

Finally, we provide a small MATLAB experiment in Figure 6.3, where we not only compare the simple setup with the Davidson setup for Q but also a proof-of-concept implementation of the Davidson setup directly for D, i.e., a Davidson method which approximates eigenvectors of D rather than singular vectors.

Similar to the results from Table 6.2 the Davidson setup on Q leads to a larger iteration count compared to the other setup methods based on D. In this case the Davidson +0 setup also requires more compute time compared to the simple

+0 setup. However, the Davidson +0 setup for D leads to a notably better solver compared to the simple +0 setup, bringing down the iteration count from 16 to 10 for this example. This shows again that the initial phase of the Davidson setup is more efficient compared to the simple setup. If we include several steps of the iterative setup, this advantage is equalized due to the increased cost of the iterative phase of the Davidson setup. This finding suggests that further research in this direction is appropriate, by developing a new eigensolver for the non-Hermitian Wilson-Dirac operator and applying it as a setup routine. Additionally, the Davidson setup would greatly benefit from more efficient strategies for the iterative phase.

Chapter 7

Conclusion & Outlook

In this chapter we summarize the results obtained within this thesis, and outline future possible research.

7.1 Conclusion

Due to the big success of algebraic multigrid methods in lattice QCD, especially the DD-αAMG method, we intended to broaden the scope of possible applications of these methods. First, we looked at the Neuberger overlap operator, which is a more faithful discretization of the Dirac equation compared to the commonly used Wilson discretization. Due to nested iterative methods, numerical simulations with this operator are about two orders of magnitude more expensive compared to simulations using the Wilson-Dirac operator. In this work, we showed how a new preconditioner can be used to reduce the run time by at least one order of magnitude. Since the overlap operator involves the matrix sign function, a direct application of multigrid methods is not feasible, instead we employed the (cheap) Wilson-Dirac operator as a preconditioner, which allowed us to indirectly benefit from multigrid methods. In numerical tests we have shown that this approach significantly improved the convergence speed for this operator. This discretization has been instrumental in experimentally validating the highly debated staggered discretization.

As a second contribution, we implemented an eigensolver for the Hermitian Wilson-Dirac operator based on the generalized Davidson method, termed GD-λAMG. The shifted linear systems of equations, which appear in every step of this method, are solved using the DD-αAMG method. In order to obtain a more favorable convergence speed, several adaptations were made. The main feature consisted of a synergy between the multigrid method and the Davidson method, where the eigenvectors computed within the Davidson method were used to improve the multigrid hierarchy, thus accelerating its convergence. In turn the multigrid method became able to efficiently generate new search directions for the Davidson method. Together with other state-of-the-art techniques, like thick restarting, harmonic Ritz extraction and locking, the GD-λAMG method not only outperforms other eigensolvers by up to a factor of three for our numerical tests, but also showed an improved scaling behavior when computing many eigenpairs for larger lattices.

With the success of the GD-λAMG method, we realized that this method could be suitable for improving the setup phase of algebraic multigrid methods, which is in general significantly more expensive than the solve phase itself. The setup phase has previously been implemented by cheap and simple vector iteration schemes, like inverse iteration. In our first (preliminary) numerical tests we have shown that the use of subspace acceleration is beneficial in scenarios where only a few solves are required for a given operator, which is the case for, e.g., the hybrid Monte Carlo algorithm. In this case, we were able to improve the run time of the whole method by nearly a factor of two.

7.2 Outlook

For now, we do not plan to further improve on the solver for the overlap operator, since the current focus of research in lattice QCD is on more computationally favorable discretizations, like the staggered operator or the so-called *(5d) domain-wall* operator, which is a rational approximation of the overlap operator. In recent publications, a direct application of multigrid methods to those operators has been introduced [19, 20], which could provide possibilities for further investigation in this area.

While the GD-λAMG eigensolver is fairly well-tuned at this point, improvements in robustness and scalability could be topics of future research. One interesting remaining question is whether it is possible to set an upper bound for the number of locked eigenvectors, i.e., only keeping at most k converged eigenvectors instead of all of them. This would lead to a fully linear complexity with respect to the number of sought eigenvectors n_{ev}, as in this case the inner products required for the locking procedure would be bounded by $k \cdot n_{ev}$ instead of n_{ev}^2. Apart from this approach, significant improvement of the performance of GD-λAMG can be achieved by improving the multigrid solver, in this case the DD-αAMG method, as the solution of the shifted linear systems is in most cases the bottleneck of the eigensolver. For this, we want to explore the possibility of GPU acceleration, where we are currently working on a GPU implementation of the SAP smoother, but plan to extend this approach to the whole multigrid method. This work is currently done by our working group as part of the PhD project by Gustavo Ramírez.

For the Davidson-type setup procedure we plan to further investigate the most beneficial choice for the test vectors. Our preliminary results suggest that eigenvectors of D are slightly more effective than singular vectors. At the same time, singular vectors can be computed as eigenvectors of the Hermitian operator Q, and are as such typically cheaper to compute. Thus, a thorough investigation including an efficient implementation of the Davidson algorithm for the computation of eigenvalues of D should be the next step for this research topic.

List of Figures

List of Tables

List of Algorithms & Scripts

Bibliography

[1] M. Albanese, F. Costantini, G. Fiorentini, F. Flore, M. P. Lombardo, P. B. R. Tripiccione, L. Fonti, E. Remiddi, M. Bernaschi, N. Cabibbo, L. A. Fernandez, E. Marinari, G. Parisi, G. Salina, S. Cabasino, F. Marzano, P. Paolucci, S. Petrarca, F. Rapuano, P. Marchesini, P. Giacomelli, and R. Rusack, *Glueball masses and string tension in lattice QCD*, Phys. Lett., B192 (1987), pp. 163–169.

[2] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math, 9 (1951), pp. 17–29.

[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 (2010), p. 201602.

[4] G. Bali, L. Castagnini, and S. Collins, *Meson and baryon masses with low mode averaging*, PoS, LATTICE2010 (2010), p. 096.

[5] G. Bali, S. Collins, A. Frommer, K. Kahl, I. Kanamori, B. Müller, M. Rottmann, and J. Simeth, *(approximate) low-mode averaging with a new multigrid eigensolver*, PoS, LATTICE2015 (2015), p. 350.

[6] G. S. Bali, S. Collins, B. Glässle, M. Göckeler, J. Najjar, R. H. Rödl, A. Schäfer, R. W. Schiel, A. Sternbeck, and W. Söldner, *The moment $\langle x \rangle_{u-d}$ of the nucleon from $n_f = 2$ lattice QCD down to nearly physical quark masses*, Phys. Rev., D90 (2014), p. 074510.

[7] G. S. Bali, H. Neff, T. Duessel, T. Lippert, and K. Schilling, *Observation of string breaking in QCD*, Phys. Rev. D, 71 (2005), p. 114513.

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

[9] B. Blossier, M. Della Morte, N. Garron, G. von Hippel, T. Mendes, H. Simma, and R. Sommer, *HQET at order $1/m$: II. spectroscopy in the quenched approximation*, JHEP, 05 (2010), p. 074.

[10] C. Bonati and M. D'Elia, *A comparison of the gradient flow with cooling in SU(3) pure gauge theory*, Phys. Rev., D89:105005 (2014).

[11] V. G. Bornyakov, R. Horsley, S. M. Morozov, Y. Nakamura, M. I. Polikarpov, P. E. L. Rakow, G. Schierholz, and T. Suzuki, *Probing the finite temperature phase transition with $N_f = 2$ nonperturbatively improved Wilson fermions*, Phys. Rev. D, 82 (2010), p. 014504.

[12] S. Borsanyi, Y. Delgado, S. Dürr, Z. Fodor, S. D. Katz, S. Krieg, T. Lippert, D. Nogradi, and K. K. Szabo, *QCD thermodynamics with dynamical overlap fermions*, Phys. Lett., B713 (2012), pp. 342–346.

[13] S. Borsanyi, Z. Fodor, S. D. Katz, S. F. Krieg, T. Lippert, D. Nogradi, F. Pittler, K. K. Szabo, and B. C. Toth, *Qcd thermodynamics with continuum extrapolated dynamical overlap fermions*, (2015). arXiv:1510.03376.

[14] 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).

[15] J. Brannick, F. Cao, K. Kahl, F. R.D., and X. Hu, *Optimal interpolation and compatible relaxation in classical algebraic multigrid*, SIAM J. Sci. Comp., 40 (2018), pp. A1473–A1493.

[16] J. Brannick, A. Frommer, K. Kahl, B. Leder, M. Rottmann, and A. Strebel, *Multigrid preconditioning for the overlap operator in lattice QCD*, Numer. Math., 132 (2016), pp. 463–490.

[17] J. Brannick and K. Kahl, *Bootstrap algebraic multigrid for the 2d Wilson Dirac system*, SIAM J. Sci. Comp., 36 (2014), pp. B321–B347.

[18] 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.

[19] R. C. Brower, M. Clark, D. Howarth, and E. S. Weinberg, *Multigrid for chiral lattice fermions: Domain wall*, (2020). arXiv:2004.07732.

[20] R. C. Brower, E. Weinberg, M. A. Clark, and A. Strelchenko, *Multigrid algorithm for staggered lattice fermions*, Phys. Rev. D, 97 (2018), p. 114513.

[21] Budapest-Marseille-Wuppertal collaboration, S. Dürr, Z. Fodor, C. Hoelbling, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, K. K. Szabó, and G. Vulvert, *Lattice QCD at the physical point: Simulation and analysis details*, JHEP, 2011 (2011), p. 148.

[22] S. Capitani, S. Dürr, and C. Hoelbling, *Rationale for UV-filtered clover fermions*, JHEP, 0611(2006)028 (2006).

[23] M. Creutz, *Why rooting fails*, PoS, LATTICE2007 (2007), p. 007.

[24] N. Cundy, J. van den Eshof, A. Frommer, S. Krieg, and K. Schäfer, *Numerical methods for the QCD overlap operator. III: Nested iterations*, Comput. Phys. Commun., 165 (2005), pp. 221–242.

[25] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Physik, 17 (1975), pp. 87–94.

[26] C. Davies et al., *High precision lattice QCD confronts experiment*, Phys. Rev. Lett., 92 (2004), p. 022001.

[27] C. De Boor, *Divided differences*, Surv. Approx. Theory, 1 (2005), pp. 46–69.

[28] P. de Forcrand, A. Kurkela, and M. Panero, *Numerical properties of staggered overlap fermions*, PoS, LATTICE2010:080 (2010).

[29] T. A. DeGrand and P. Rossi, *Conditioning techniques for dynamical fermions*, Comput. Phys. Commun., 60 (1990), pp. 211–214.

[30] T. A. DeGrand and S. Schaefer, *Improving meson two point functions in lattice QCD*, Comput. Phys. Commun., 159 (2004), pp. 185–191.

[31] L. Del Debbio, L. Giusti, M. Lüscher, R. Petronzio, and N. Tantalo, *QCD with light Wilson quarks on fine lattices (i): First experiences and physics results*, JHEP, 02(2007)056 (2007).

[32] ——, *QCD with light Wilson quarks on fine lattices (ii): DD-HMC simulations and data analysis*, JHEP, 0702(2007)082 (2007).

[33] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth, *Hybrid monte carlo*, Phys. Lett. B, 195 (1987), pp. 216–222.

[34] 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.

[35] S. Durr, Z. Fodor, C. Hoelbling, S. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, K. Szabo, and G. Vulvert, *Lattice QCD at the physical point: Light quark masses*, Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics, 701 (2011), pp. 265–268.

[36] S. Durr and G. Koutsou, *On the suitability of the Brillouin action as a kernel to the overlap procedure*, (2017). arXiv:1701.00726.

[37] R. G. Edwards, U. M. Heller, and R. Narayanan, *A study of practical implementations of the overlap Dirac operator in four-dimensions*, Nucl. Phys., B540 (1999), pp. 457–471.

[38] M. Eiermann and O. G. Ernst, *A restarted Krylov subspace method for the evaluation of matrix functions*, SIAM J. Numer. Anal., 44 (2006), pp. 2481–2504.

[39] E. Endress, C. Pena, and K. Sivalingam, *Variance reduction with practical all-to-all lattice propagators*, Comput. Phys. Commun., 195 (2015), pp. 35–48.

[40] S. Fischer, A. Frommer, U. Glassner, T. Lippert, G. Ritzenhofer, and K. Schilling, *A parallel SSOR preconditioner for lattice QCD*, Comput. Phys. Commun., 98 (1996), pp. 20–34.

[41] J. Foley, K. Jimmy Juge, A. O'Cais, M. Peardon, S. M. Ryan, and J. Skullerud, *Practical all-to-all propagators for lattice QCD*, Comput. Phys. Commun., 172 (2005), pp. 145–162.

[42] A. Frommer, S. Güttel, and M. Schweitzer, *Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1602–1624.

[43] ——, *Efficient and stable Arnoldi restarts for matrix functions based on quadrature*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 661–683.

[44] A. Frommer, K. Kahl, F. Knechtli, M. Rottmann, A. Strebel, and I. Zwaan, *A multigrid accelerated eigensolver for the Hermitian Wilson-Dirac operator in lattice QCD*, (2020). arXiv:2004.08146.

[45] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *An adaptive aggregation based domain decomposition multilevel method for the lattice Wilson Dirac operator: Multilevel results*, 2013. arXiv:1307.6101.

[46] ——, *Adaptive aggregation based domain decomposition multigrid for the lattice Wilson-Dirac operator*, SIAM J. Sci. Comp., 36 (2014), pp. A1581–A1608.

[47] A. Frommer, A. Nobile, and P. Zingler, *Deflation and flexible SAP-preconditioning of GMRES in lattice QCD simulations*, (2012). arXiv:1204.5463.

[48] P. H. Ginsparg and K. G. Wilson, *A remnant of chiral symmetry on the lattice*, Phys. Rev. D, 25 (1982), pp. 2649–2657.

[49] L. Giusti, P. Hernandez, M. Laine, P. Weisz, and H. Wittig, *Low-energy couplings of QCD from current correlators near the chiral limit*, JHEP, 04 (2004), p. 013.

[50] L. Giusti, C. Hoelbling, M. Lüscher, and H. Wittig, *Numerical techniques for lattice QCD in the epsilon regime*, Comput. Phys. Commun., 153 (2003), pp. 31–51.

[51] M. Guest, G. Aloisio, R. Kenway, et al., *The scientific case for HPC in Europe 2012 - 2020*, tech. rep., PRACE, October 2012. http://www.prace-ri.eu/PRACE-The-Scientific-Case-for-HPC, p. 75.

[52] T. R. Haar, *Optimisations to Hybrid Monte Carlo for Lattice QCD*, PhD thesis, Adelaide U., 2019.

[53] W. Hackbusch, *Multi-Grid Methods and Applications*, vol. 4 of Springer Series in Computational Mathematics, Springer, 1985.

[54] W. W. Hager, *Condition estimates*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 311–316.

[55] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration*, vol. 31 of Springer Series in Computational Mathematics, Springer, Heidelberg, 2010.

[56] A. Hasenfratz and F. Knechtli, *Flavor Symmetry and the Static Potential With Hypercubic Blocking*, Phys. Rev. D, 64 (2001), p. 034504.

[57] W. Hastings, *Monte carlo sampling methods using Markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109.

[58] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[59] N. J. Higham and F. Tisseur, *A block algorithm for matrix 1-norm estimation with an application to 1-norm pseudospectra*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1186–1201.

[60] IEEE, *IEEE standard for floating-point arithmetic*, IEEE Std 754-2008, (2008), pp. 1–70.

[61] Jülich Supercomputing Centre, *JURECA - Jülich research on exascale cluster architectures.* `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html`.

[62] ——, *JUROPA - Jülich research on petaflop architectures.* `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUROPA/JUROPA_node.html`.

[63] ——, *JUWELS - Jülich wizard for european leadership science.* `http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUWELS/JUWELS_news.html`.

[64] J. B. Kogut and L. Susskind, *Hamiltonian formulation of Wilson's lattice gauge theories*, Phys. Rev. D, 11 (1975), pp. 395–408.

[65] M. Lüscher, *OpenQCD simulation package.* `http://luscher.web.cern.ch/luscher/openQCD/`.

[66] M. Lüscher, *Exact chiral symmetry on the lattice and the Ginsparg-Wilson relation*, Phys. Lett., B428 (1998), pp. 342–345.

[67] ——, *Solution of the Dirac equation in lattice QCD using a domain decomposition method*, Comput. Phys. Commun., 156 (2004), pp. 209–220.

[68] M. Lüscher, *Local coherence and deflation of the low quark modes in lattice QCD*, JHEP, 2007 (2007), p. 081.

[69] M. Lüscher, *Properties and uses of the Wilson flow in lattice QCD*, JHEP, 1008(2010)071 (2010).

[70] ——, *Trivializing maps, the Wilson flow and the HMC algorithm*, Commun. Math. Phys., 293 (2010), pp. 899–919.

[71] MATLAB, *version 7.10.0 (R2010a)*, The MathWorks Inc., Natick, Massachusetts, 2010.

[72] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, *Equation of state calculations by fast computing machines*, J. Chem. Phys., 21 (1953), pp. 1087–1092.

[73] I. Montvay and G. Münster, *Quantum Fields on a Lattice*, Cambridge Monographs on Mathematical Physics, Cambridge University Press, 1994.

[74] R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 817–825.

[75] C. Morningstar and M. J. Peardon, *Analytic smearing of SU(3) link variables in lattice QCD*, Phys. Rev., D69:054501 (2004).

[76] H. Neff, N. Eicker, T. Lippert, J. W. Negele, and K. Schilling, *On the low fermionic eigenmode dominance in QCD on the lattice*, Phys. Rev., D64 (2001), p. 114509.

[77] S. Nepomnyaschikh, *Mesh theorems on traces, normalizations of function traces and their inversion*, Soviet J. Numer. Anal. Math. Modelling, 6 (1991), pp. 223–242.

[78] H. Neuberger, *Exactly massless quarks on the lattice*, Phys. Lett. B, 417 (1998), pp. 141–144.

[79] H. B. Nielsen and M. Ninomiya, *No go theorem for regularizing chiral fermions*, Phys. Lett., 105B (1981), pp. 219–223.

[80] Y. Notay, *Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 9 (2002), pp. 21–44.

[81] J. Osborn, *QOPQDP software*. `https://github.com/usqcd-software/qopqdp`.

[82] J. C. Osborn, R. Babich, J. Brannick, R. C. Brower, M. A. Clark, S. D. Cohen, and C. Rebbi, *Multigrid solver for clover fermions*, PoS, LATTICE2010 (2010), p. 037.

[83] P. Oswald, *Preconditioners for nonconforming discretizations*, Math. Comp., 65 (1996), pp. 923–941.

[84] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1998), pp. 115–133.

[85] M. Rottmann, *Adaptive Domain Decomposition Multigrid for Lattice QCD*, PhD thesis, University of Wuppertal, 2016.

[86] M. Rottmann, A. Strebel, S. Heybrock, S. Bacchio, B. Leder, and I. Kanamori, *DD-αAMG software*. `https://github.com/DDalphaAMG/DDalphaAMG`.

[87] J. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, USA, 1987, pp. 73–130.

[88] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput, 14 (1992), pp. 461–469.

[89] ——, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, USA, 2nd ed., 2003.

[90] ——, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, PA, USA, 2nd ed., 2011.

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

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

[93] 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.

[94] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[95] 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.

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

[97] D. Sorensen, R. Lehoucq, C. Yang, and K. Maschhoff, *PARPACK*. `http://http://www.caam.rice.edu/software/ARPACK`, used version: 2.1, September 1996.

[98] A. Stathopoulos, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. part I: Seeking one eigenvalue*, SIAM J. Sci. Comput., 29 (2007), pp. 481–514.

[99] A. Stathopoulos and J. R. McCombs, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. part II: Seeking many eigenvalues*, SIAM J. Sci. Comput., 29 (2007), pp. 2162–2188.

[100] A. Stathopoulos and J. R. McCombs, *PRIMME: PReconditioned Iterative MultiMethod Eigensolver: Methods and software description*, ACM Transactions on Mathematical Software, 37 (2010), pp. 21:1–21:30.

[101] B. C. Toth, *QCD thermodynamics with dynamical overlap fermions*, PoS, LATTICE2013:163 (2013).

[102] L. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1st ed., 1997.

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

[104] J. van den Eshof, A. Frommer, T. Lippert, K. Schilling, and H. A. van der Vorst, *Numerical methods for the QCD overlap operator. I: Sign-function and error bounds*, Comput. Phys. Commun., 146 (2002), pp. 203–224.

[105] 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.

[106] C. Vuik and H. van der Vorst, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[107] K. G. Wilson, *Confinement of quarks*, Phys. Rev. D, 10 (1974), pp. 2445–2459.

[108] ——, *Quarks and strings on a lattice*, in New Phenomena in Subnuclear Physics: Proceedings, International School of Subnuclear Physics, Erice, Sicily, Jul 11-Aug 1 1975. Part A, 1975, p. 99. [,0069(1975)].

[109] L. Wu, E. Romero, and A. Stathopoulos, *PRIMME_SVDS: A high-performance preconditioned SVD solver for accurate large-scale computations*, SIAM J. Sci. Comput., 39 (2016), pp. S248–S271.

[110] J. Xu, *The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids*, Comput., 56 (1996), pp. 215–235.