AN ADAPTIVE AGGREGATION BASED DOMAIN DECOMPOSITION MULTILEVEL METHOD FOR THE LATTICE WILSON DIRAC OPERATOR: MULTILEVEL RESULTS *

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

Abstract. In lattice QCD computations a substantial amount of work is spent in solving linear systems arising in Wilson's discretization of the Dirac equations. We show first numerical results of the extension of the two-level DD-αAMG method to a true multilevel method based on our parallel `MPI-C` implementation. Using additional levels pays off, allowing to cut down the core minutes spent on one system solve by a factor of approximately 700 compared to standard Krylov subspace methods and yielding another speed-up of a factor of 1.7 over the two-level approach.

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

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

1. Introduction. In [12] we recently proposed an adaptive aggregation based domain decomposition two-level ("DD-αAMG") method to solve linear systems

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

arising in Wilson's discretization of the Dirac equations. Solving these systems makes up a large part of the compute time spent in lattice QCD simulations [2, 13]. These are among the most demanding in computational science, and thus triggered intense research activity in the construction of suitable preconditioners and, consequently, more efficient solvers for these systems in recent years [1, 5, 12, 18, 20].

Our two-level method combines a multiplicative Schwarz method (SAP) as the smoother with an aggregation based coarse-level correction, components which were also used in the construction of an "inexact deflation" approach in [18]. In contrast to the approach developed in [18], the two-level method from [12] arranged these ingredients in a "multigrid" fashion, and thus the coarse-level system needed to be solved only to very low accuracy in each iteration. This yielded the fastest run times for the two-level method and now opens the path for a true multilevel method which we present in this paper.

Another multilevel approach for (1.1), which also uses an aggregation based coarse-level correction but a different, non-stationary smoothing iteration, has been developed in [1, 5, 20]. Significant speed-ups over traditional Krylov subspace methods were reported for this approach. We replace the smoother used in [1, 5, 20] by SAP in order to be able to benefit from data locality and improve the strong scaling. In addition, the new setup routine for our adaptive multilevel domain decomposition approach differs from the one used in [20] in an important aspect: By combining the "inverse-iteration"-type approach from [18] with a bootstrap-type approach from [4] we are able to generate appropriate test vectors more efficiently.

Experiments reported in [12] show significant speed-up of the two-level DD-αAMG method over conventional Krylov-subspace methods and notable speed-up over the other hierarchical preconditioners mentioned above. Without going into detail about the specific implementation of the method in [12], we note that its error propagator is—as for many other two-level approaches—of the generic form

$$E_{2g} = (I - MD)^{\nu}(I - PD_c^{-1}RD)(I - MD)^{\mu}.$$

---

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

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

Here, M denotes the smoother, μ and ν are the number of pre- and post-smoothing iterations, and P and R the interpolation and restriction operators, respectively. Note, that in practice D_c^{-1} can and will be approximated to low accuracy.

The Wilson Dirac matrix, D, is γ_5-symmetric, i.e., $D\Gamma_5 = \Gamma_5 D^\dagger$ with Γ_5 acting as the identity on spins 1 and 2 and the negative identity on spins 3 and 4, see [12]. We chose a γ_5-symmetry preserving Galerkin construction of the coarse-level operator so that we have $R := P^\dagger$, where $P^\dagger$ denotes the conjugate transpose of P, and $D_c := P^\dagger D P$. By constructing the interpolation P to group only spin 1 and 2 as well as spin 3 and 4 variables, the γ_5-symmetry of D is preserved on the coarse-level, i.e., D_c is again γ_5-symmetric. The transition to a multilevel method is in principle straight-forward: one simply uses another two-level ansatz for the solution of the linear system involving D_c and applies this construction principle recursively until a level is reached where a direct, "exact" solution of the coarse-level system is feasible.

The remainder of this note is organized as follows. In Section 2 we introduce some notation for the multilevel method and give details about the cycling strategy used and the setup employed. Thereafter we present extensive numerical studies conducted with a three- and four-level method in Section 3, showing that the speed-ups anticipated in [12] are achieved in practice. Finally, we give some concluding remarks and an outlook on possible future progress in Section 4.

2. Domain Decomposition Adaptive Algebraic Multigrid. Our multilevel extension of the two level-approach from [12] combines the two same components, namely a multiplicative Schwarz method (SAP) [17, 22] as the smoother and a γ_5-symmetry preserving aggregation based interpolation [1, 5, 6, 12, 20], on every level. This means that the smoother as well as the interpolation together with the associated coarse-level correction are of the same type on all levels of the hierarchy. As in the two-level case, and for the same reasons (cf. [12]), the multilevel method is used as a preconditioner to a flexible Krylov subspace method, e.g., FGMRES (see [21]).

To be more specific, let L denote the number of levels to be used in the hierarchy and denote $D_1 := D$. Then the setup of our method constructs interpolation operators P_ℓ for $\ell = 1, \ldots, L-1$, which transfer information from level $\ell+1$ to level ℓ, and computes coarse-level operators $D_{\ell+1} = (P_\ell)^\dagger D_\ell P_\ell$. Given an SAP smoother, represented by its error propagation operator $I - M_\ell D_\ell$, on every level as well, the simplest multilevel cycling strategy that can be employed is the V-cycle illustrated in Algorithm 1. Here, on each level only one recursive call is used in-between pre- and post-smoothing iterations.

Algorithm 1 $z_\ell = \text{V-Cycle}(\ell, b_\ell)$

1: **if** $\ell = L$ **then**
2:     $z_\ell \leftarrow D_\ell^{-1} b_\ell$
3: **else**
4:     $z_\ell = 0$
5:     **for** $i = 1$ to μ **do**
6:         $z_\ell \leftarrow z_\ell + M_\ell(b_\ell - D_\ell z_\ell)$
7:     **end for**
8:     $b_{\ell+1} \leftarrow P_\ell^\dagger(b_\ell - D_\ell z_\ell)$
9:     $z_{\ell+1} \leftarrow \text{V-Cycle}(\ell+1, b_{\ell+1})$
10:    $z_\ell \leftarrow z_\ell + P_\ell z_{\ell+1}$
11:    **for** $i = 1$ to ν **do**
12:       $z_\ell \leftarrow z_\ell + M_\ell(b_\ell - D_\ell z_\ell)$
13:    **end for**
14: **end if**

2.1. Multilevel K-Cycles. Numerical tests with very large configurations and small quark masses have shown that a simple V-cycle is often not the ideal choice in terms of solver performance. Thus we

consider using a more elaborate cycling strategy, the K-cycle suggested in [19], in our method. Instead of only one recursive call of the coarse-level solver, a K-cycle optimally recombines several coarse-level solves. More precisely, on every level ℓ we approximate the solution of the coarse-level system by a few iterations of a flexible Krylov subspace method, which in turn is preconditioned by the K-cycle multilevel method from level $\ell + 1$ to L. In here, we deviate from the approach in [19] by using a stopping criterion based on the reduction of the associated residual rather than a fixed number of iterations. We give the specific choice of the stopping criterion used in our implementation in Section 3.

The K-cycle is illustrated in Algorithm 2. For a fixed number of iterations it can be regarded as a standard W-cycle; see e.g. [23], with adaptive re-weighting of the approximate solutions after each recursion.

Algorithm 2 $z_\ell = \text{K-Cycle}(\ell, b_\ell)$
---
perform Algorithm 1 with line 9 replaced by
$z_{\ell+1} \leftarrow$ FGMRES for matrix $D_{\ell+1}$ and r.h.s. $b_{\ell+1}$, preconditioned with K-Cycle$(\ell + 1, b_{\ell+1})$

2.2. Multilevel Setup. For the construction of the aggregation based γ_5-symmetry preserving interpolation operators P_ℓ, and with them the coarse-level operators $D_{\ell+1}$, we have extended our setup from [12] to a multilevel setup. In order to preserve the γ_5-symmetry on all levels, we use a block-spin structure for the interpolation operators on all levels. The setup process that we found to work best in practice is divided into two phases:

1. An initial phase given as Algorithm 3 which constructs an initial multilevel hierarchy solely based on the smoothing iteration starting with random test vectors.

2. An iterative phase given in Algorithm 4 and illustrated in Fig. 2.1, where the current multilevel method is used to update and improve the multilevel hierarchy by generating improved test vectors.

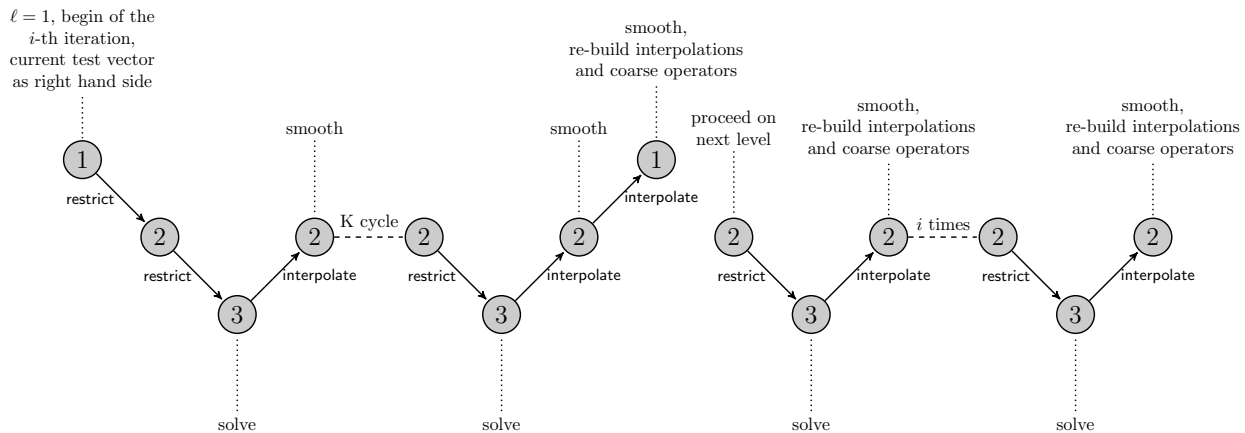Again we give our specific choices for the various parameters in the setup in Section 3.

FIG. 2.1. *Illustration of Algorithm 4 as iteration i of the iterative setup phase.*

3. Numerical Results. In this section we show extensive numerical results for our multilevel domain decomposition adaptive algebraic multigrid method (DD-αAMG), especially for the three- and four-level setting. As its predecessor in [12] the method is implemented as a parallel program in C using MPI, and we compute and apply the DD-αAMG preconditioner in single precision only. The outer FGMRES iteration

Algorithm 3 initial_setup_phase(ℓ)

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

Algorithm 4 iterative_setup_phase(ℓ,i)

1: **if** $\ell < L$ **then**
2:     **for** $j = 1$ to N **do**
3:        $z_\ell \leftarrow$ K-Cycle($\ell, v_{\ell-1}^{(j)}$)
4:        **for** $l = \ell$ to $L - 1$ **do**
5:           $v_l^{(j)} = z_l/||z_l||$ {update test vectors with the iterates of each level}
6:        **end for**
7:     **end for**
8:     **for** $l = \ell, \ldots, L - 1$ **do**
9:        construct P_l and D_{l+1}
10:     **end for**
11:     **for** $q = 1$ to i **do**
12:        iterative_setup_phase($\ell + 1, q$) {perform Algorithm 4 on next level}
13:     **end for**
14: **end if**

---

remains in double precision. The system on the coarsest level is solved via odd-even preconditioned GMRES to a given relative accuracy ϵ. The operators on all levels, including the finest-level Wilson Dirac operator, are implemented as proposed in [14]. That is, the underlying lattice is used to optimize the matrix vector multiplication.

As a new feature of the implementation, processes are now allowed to idle on the coarser levels. This is necessary since a high degree of parallelization can cause a lack of lattice sites on the coarser levels. In our implementation we assume that sites which belong to a common aggregate always share the same process. This allows to perform the computational part of interpolation and restriction without communication similarly to what has been done in the two-level approach.

Table 3.1 summarizes the default parameters used in our experiments. Note that these parameters are

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

TABLE 3.1

Parameters for the DD-αAMG multi-level method. (∗) : same in solver and setup.

id	lattice size $N_t \times N_s^3$	pion mass m_π [MeV]	CGNR iterations	shift m_0	clover term c_{sw}	provided by
1	48×48^3	135	53,932	-0.09933	1.00000	BMW-c [8, 9]
2	64×64^3	135	84,207	-0.05294	1.00000	BMW-c [8, 9]
3	128×64^3	270	45,804	-0.34262	1.75150	CLS [7, 11]
4	128×64^3	190	88,479	-0.33485	1.90952	CLS [7, 11]

TABLE 3.2

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

the same as those used for the two-level results in [12] except that we reduced the restart length of the outer FGMRES routine to $n_{kv} = 10$ due to memory limitations when using a small number of cores.

In Table 3.2 we give an overview of the configurations used in our tests. The iteration count of CGNR, i.e., CG applied to the system $D^\dagger D z = D^\dagger b$ with the residual $r = b - Dz$, can be used as an indicator for the conditioning of the respective operator. All of the configurations we use in our tests correspond to some of the most challenging linear systems encountered in state-of-the-art lattice QCD calculations.

In what follows we explore the potential benefits of additional levels in the DD-αAMG method. Special focus is put on the consequences of additional levels in terms of the degree of parallelization, i.e., the size of the local lattice kept on each node. We tested the performance of the DD-αAMG method with different numbers of levels for a variety of cost measures and analyzed the scaling behavior as a function of the bare mass m_0. Finally, we compare the DD-αAMG approach to the recently improved version of the inexact deflation approach which now allows for inexact projection [16, 18].

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. This machine provides a maximum of 8,192 cores for a single job. For compilation we used the `icc`-compiler with the optimization flags `-O3`, `-ipo`, `-axSSE4.2` and `-m64`.

3.1. Parallel Multilevel Methods. Our first tests analyze the influence of the degree of parallelization on the performance of the two- and three-level method. This is an important aspect since using many processes causes idle times on the coarser levels. In order to reduce or even avoid these it might be promising to choose a lower degree of parallelization. This helps to even out the work-load on all levels such that the communication overhead can be mostly neglected. In this manner we aim at an overall optimal use of resources, measured in core-minutes (time to solution multiplied with the number of cores used), rather than just time to solution.

We tracked the performance of the two- and three-level DD-αAMG method for configurations 1 and 2 using different degrees of parallelization, reported as the size of the local lattice on each core. Our goal is to find the sweet spot with respect to consumed core minutes.
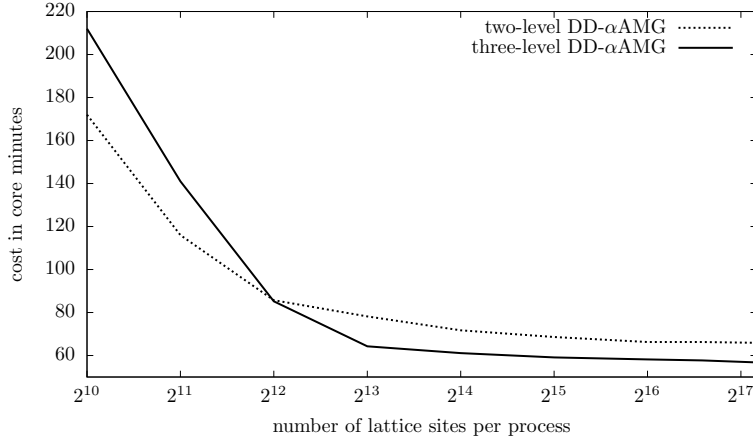

FIG. 3.1. *Estimation of the sweet spot on configuration 1.*

Figure 3.1 plots the dependence of the solver performance on the degree of parallelization for the rather small configuration 1 with lattice size 48^4. Due to memory restrictions the minimal number of processors that can be used is 36 corresponding to roughly 2^{17} lattice sites per processor. On the other end of the horizontal axis we are limited to 5,184 processors which corresponds to 2^{10} lattice sites per process on level 1.

Figure 3.1 shows that as soon as a local lattice with fewer than $8^4 = 2^{12}$ lattice sites per processors on level 1 is reached, the performance of the two-level method exceeds the performance of the three-level method. This is partly to be expected, since this is exactly the spot where idling processors on the second level cannot be avoided. To be more precise, on level two we need at least 4×2^3 lattice sites per processor in order to be able to apply SAP since we need one block of each color on each non-idling process. On level three, i.e., the coarse level, we also assume to have at least two lattice sites per non-idling process because we solve this system using odd-even preconditioning. Thus every second process is idling on level two and three. Similarly, for 2^{11} local sites three out of four processes and for 2^{10} sites 7 out of 8 processes are idling on levels two and three. Thus, for the relatively small configuration 1 the additional third level is advisable only if a relatively small degree of parallelism is used, and even then the gain over the two-level method is rather modest.

The picture changes, however, when we investigate the same dependence for a larger configuration. In Figure 3.2 we see an almost constant absolute gain when going from the two-level to the three-level DD-αAMG method. For any of the considered degrees of parallelization, the three-level method outperforms the two level method, and the sweet spot is taken again for 2^{17} lattice sites per process on the finest level, i.e., for 128 processes and a 32×16^3 local lattice on each process. For this particular case the three-level method shows a speed up factor of 1.7 over the two-level method. At the opposite end with a local lattice size of 2^{11} and 8,192 processes, the three-level method still gains a factor of about 1.3.
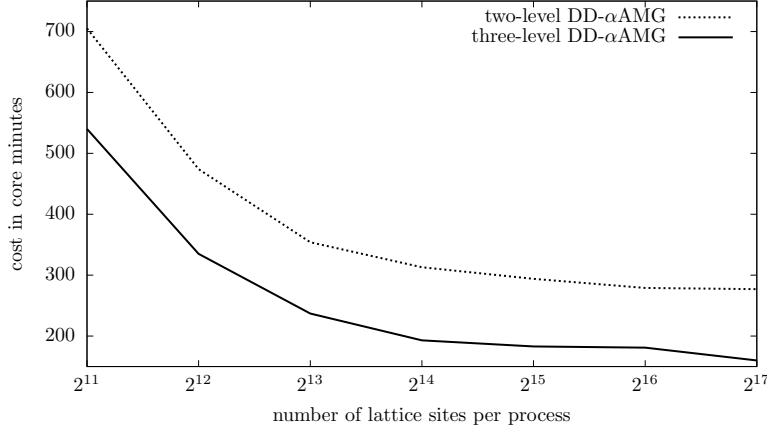
FIG. 3.2. *Estimation of the sweet spot on configuration 2.*

3.2. Two, Three and Four Levels. Now we compare DD-αAMG methods with two, three and four levels for all four configurations, using only small numbers of cores, i.e., working with large local lattices.

	configuration	1	2	3	4
	lattice size	48×48^3	64×64^3	128×64^3	128×64^3
	pion mass m_π	$135\,\mathrm{MeV}$	$135\,\mathrm{MeV}$	$270\,\mathrm{MeV}$	$190\,\mathrm{MeV}$
two levels	setup time	316s	736s	630s	701s
	solve time	48.6s	130s	113s	141s
	solve iter	23	22	24	28
three levels	setup time	374s	744s	719s	948s
	solve time	42.6s	75.2s	74.0s	79.0s
	solve iter	24	21	22	24
four levels	setup time	–	806s	755s	1,004s
	solve time	–	79.8s	75.7s	79.1s
	solve iter	–	22	22	24
	processes	81	128	256	256
local lattice	level 1	16×16^3	32×16^3	32×16^3	32×16^3
	level 2	4×4^3	8×4^3	8×4^3	8×4^3
	level 3	2×2^3	4×2^3	4×2^3	4×2^3
	level 4	–	2×1^3	2×1^3	2×1^3

TABLE 3.3

Comparison of DD-αAMG with two, three and four levels for a small number of processes, parameters from Tables 3.1 and 3.2.

Table 3.3 shows that the three-level method outperforms the other variants in terms of consumed core minutes in all tests except for configuration 1 by factors between 1.5 and 1.7, and it also outperforms the four-level method. Note that the time spent in the setup naturally increases when using additional levels, since additional operators need to be set up and only in one test this additional work amortized immediately in one solve. While we still believe that we can improve the setup routine and with it the additional overhead to be paid for additional levels, the overhead has to be kept in mind when choosing the number of levels, depending on the number of right-hand-sides to be solved. Interpreting Table 3.3 with respect to the system size, we see that the performance gain of additional levels grows with the system size. This indicates that

we can expect the gain of three- or potentially four-level approaches to grow for even larger lattices.

	configuration	1		2	
	lattice size	48×48^3		64×64^3	
	pion mass m_π	135 MeV		135 MeV	
	levels	2	3	2	3
	setup time	14.9s	40.0s	24.5s	46.1s
	solve time	2.69s	3.26s	5.21s	3.95s
	solve iter	23	24	23	22
level 1	consumed time	0.880s	0.895s	1.04s	0.930s
	wait time	0.110s	0.115s	0.124s	0.135s
level 2	consumed time	1.81s	1.50s	4.17s	1.48s
	wait time	0.0972s	0.0557s	0.171s	0.0430s
level 3	consumed time	–	0.865s	–	1.54s
	wait time	–	0.0993s	–	0.0942s
summarized	total wait time	0.207s	0.270s	0.295s	0.272s
	processes	2,592	2,592	8,192	8,192
local lattice	level 1	4×8^3	4×8^3	4×8^3	4×8^3
	level 2	1×2^3	4×2^3	1×2^3	4×2^3
	level 3	–	2×1^3	–	2×1^3

TABLE 3.4

Comparison of DD-αAMG with two and three levels for a large number of processes, parameters from Tables 3.1 and 3.2.

In Table 3.4 we show additional timings for comparatively large numbers of processes where idle times on the coarser levels occur. More precisely, for the number of processes chosen in the tests, three out of four processes idle on the second and third level within the three-level method while for the two-level variant there are no idle times at all. For configuration 1 we observe that the two-level method outperforms the three-level method with respect to setup and solve time as expected based on Figure 1. In this particular case the three-level method is unable to transfer enough work to the coarse level and level two remains the most expensive part of the solve phase. Together with the fact that for this configuration the second level can be solved quite efficiently by odd-even preconditioned GMRES alone, a third level does not pay off.

For the larger configuration 2, the situation is similar with respect to setup timings, but the second level seems to be much more ill-conditioned which yields an advantage for the three-level method regarding the solve time. Thus in situations where the increased setup time can be compensated for by solving systems with several right-hand-sides, the three-level method pays off.

Besides these observations we also note that the influence of wait times caused by nearest neighbor communication on the coarse level even with a local lattice size of 2×1^3 can be neglected. Thus the overall performance loss on the coarser levels is dominated by the percentage of idling processes and not the communication overhead.

3.3. Comparison with CGNR. We now want to study the two- and three-level method in more detail, namely, with respect to absolute cost measures and in comparison to the conventional Krylov subspace method of choice, CGNR. We choose to show absolute cost measures, e.g., flop count and core minutes rather than wall-clock time. The performance of both methods, DD-αAMG and CGNR, in terms of flop/s (floating point operations per second) can differ dramatically depending on the level of optimization of the Dirac operator and on the degree of parallelization chosen, whereas core minutes represent a measure that takes both important resources, time and hardware, into account.

In Table 3.5 we show results of our method and CGNR for configuration 2. First note that compared to CGNR the two-level method speeds up the calculation by an order of magnitude and cuts down the flop count per lattice site and overall core minute count even by more than two orders of magnitude. The addition

of a third level yields another factor of roughly 1.7 across the board with respect to the two-level method. It is noteworthy that the three-level DD-αAMG performs at 2.86 Gflop/s per core, corresponding to 12.2% peak performance, on Juropa with a pure C-code, i.e., without any machine specific optimization.

The abysmal Gflop/s per core performance of CGNR is mainly due to two facts. First we run CGNR completely in double precision, second we are limited to at most 8,192 cores per run and the corresponding local lattice size of 4×8^3 leads to a problem size exceeding the cache size.

	three-level DD-αAMG	two-level DD-αAMG	CGNR
processes	128	128	8,192
solve time	75.2s	130s	816s
consumed core minutes	160	277	111,446
consumed Mflop per site	1.64	2.97	364
Gflop/s per core	2.86	2.99	0.91

TABLE 3.5

Comparison of three-level DD-αAMG with CGNR, parameters from Tables 3.1 and 3.2.

Combined with the results reported in Section 3.2 our results suggest that in situations where total run-time is not of great importance and where several Dirac systems with different right-hand-sides must be solved, e.g., in configuration analysis, it is best to run a multilevel method with a low degree of parallelization. This gives the largest number of system solves per core minute, i.e., the largest data per Euro ratio. On the other hand, in a situation, where total run-time is of utmost importance, e.g., the generation of configurations within the HMC process, it might be advisable to use only a two-level method and a high degree of parallelization.

3.4. Scaling Tests. An important motivation for the development of multilevel preconditioners for the Wilson Dirac system has always been the removal of "critical slowing down", i.e., the observed dramatic drop in performance when the mass parameter approaches its critical value. In the next set of tests we report the time to solution of the two-, three- and four-level DD-αAMG method with respect to the bare mass m_0 for configuration 2. These tests are again carried out using only 128 cores, i.e., a low degree of parallelization.
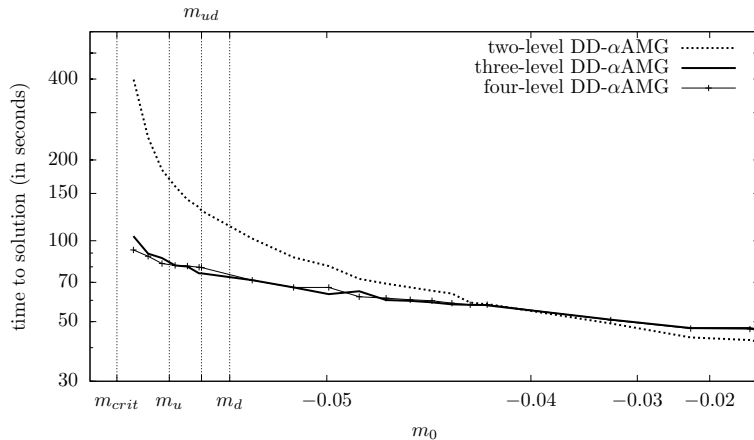

FIG. 3.3. *Scaling of DD-αAMG with the bare mass m_0. In here, $m_{ud} = -0.05294$ denotes the physical mass parameter for which the configuration was thermalized and $m_{crit} = -0.05419$ [8, 9] is the critical mass.*

We see in Figure 3.3 that the time to solution of the two-level method increases much more rapidly than

the time to solution of the three-level method when approaching the critical bare mass m_{crit}. Further we see that beyond m_u the four-level method starts to outperform the three-level method. Note that for the computation of observables in lattice QCD it might actually be necessary to solve the Dirac equation for mass parameters beyond m_{ud}, e.g, to account for the up-down quark mass difference [3, 10]. For configuration 2 this requires a mass parameter $m_u = -0.05347$ which leads to a factor of approximately 1.6 in the condition number compared to m_{ud}.

FIG. 3.4. *Scaling of the remaining error obtained from DD-αAMG*

We also tracked the norm of the relative error $||e||/||z^*|| = ||z^* - z||/||z^*||$ as a function of m_0 for a pre-determined solution z^*. Figure 3.4 shows that the error only slightly increases in the range of the physical masses when using more levels even though the tolerance for the K-cycle on the second level is a factor of 2 less precise than for odd-even GMRES within the two-level method. When approaching the critical bare mass, four-level DD-αAMG also provides the most stable error.

3.5. Comparison with Inexact Deflation with Inexact Projection. Recently the implementation of inexact deflation was upgraded within the openQCD code [16]. The new version of inexact deflation, termed "inexact deflation with inexact projection" is similar in spirit to our method proposed in [12], while it differs in its construction of the interpolation and the coarse-level operator. In the inexact deflation approach γ_5-symmetry is not preserved on the coarse level.

In order to account for this recent upgrade of the openQCD code, we compare it with multilevel DD-αAMG. As the openQCD code uses open boundary conditions in time direction we cannot use our set of configurations with this code. Thus we integrated the new modules containing the inexact deflation with inexact projection method into the DD-HMC [15] and compared both methods.

Table 3.6 shows that the inexact deflation with inexact projection method for configuration 2 is 2.3 times faster than without inexact projection, and also a factor of 1.35 faster than two-level DD-αAMG. This is mainly due to the fact that inexact deflation with inexact projection does not preserve the γ_5-symmetry on the coarse level. It therefore ends up with only half as many variables and a four times less complex operator on the coarse level if the same number of test vectors is used. As a consequence we found numerically that the inexact deflation with inexact projection approach has to compute more test vectors, but not twice as many, to construct a comparably "rich" coarse-level subspace which is needed to achieve the same number of iterations as the γ_5-symmetry preserving approach. Although the setup time of the three-level DD-αAMG method is slightly longer than that of two of the other methods, the reduced solve time more than compensates for this already for a single right-hand-side.

Note that the results presented so far for the γ_5-symmetry preserving interpolation show that the recur-

10

	three-level DD-αAMG	two-level DD-αAMG	inexact deflation (DD-HMC-1.2.2) [15]	inexact deflation with inexact projection [16]
test vectors N	20	20	20	32
setup bootstrap iter	6	6	10	6
SAP block size	2^4	2^4	4^4	4^4
block solver iter	3	3	4	4
setup time	744s	736s	681s	897s
solve iter	21	22	48	18
solve time	75.2s	130s	223s	96.6s

TABLE 3.6

Comparison on configuration 2 using 128 processes, parameters from Tables 3.1 and 3.2.

sive extension of DD-αAMG works and, in particular, that the SAP smoother still works on coarse levels. It is unclear whether this is still the case if γ_5-symmetry is not preserved on the coarse level, so that a recursive extension of this approach might not benefit as much from additional levels. These questions can probably only be answered experimentally.
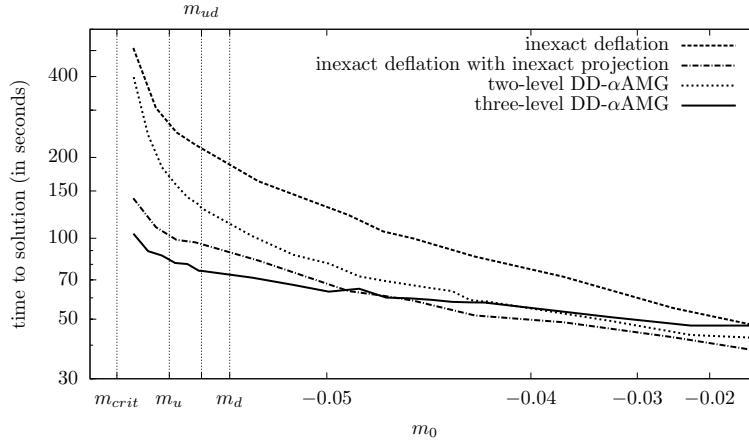

FIG. 3.5. *Scaling with the bare mass m_0 of inexact deflation and DD-αAMG on configuration 2 using 128 processes.*

As the behavior of solvers approaching the critical mass is an important bench-mark, we also investigate the scaling of the new inexact deflation with inexact projection method. Figure 3.5 shows the scaling behavior of all the methods reported in Table 3.1 as a function of the bare mass m_0. Inexact deflation with inexact projection and two-level DD-αAMG show a similar scaling behavior until m_{ud}. Beyond this mass, two-level DD-αAMG tends to scale similarly to the ordinary inexact deflation approach. Besides that we note that the upgrade of the inexact deflation method also leads to an improved scaling behavior. Thus the ability to solve the coarse system equations inexactly and the ability to use more test vectors and still having a cheap coarse-level operator can fundamentally influence the scaling behavior of the two-level method. Though the overall best scaling curve belongs to three-level DD-αAMG and the third level already pays off for heavier masses than m_d and will pay off even more in the future when even larger lattices will be used.

4. Conclusions and Outlook. We successfully extended our two-level method to a true multilevel method including a multilevel setup. For certain cases our three-level implementation obtains speed-ups of up to a factor of 1.7 compared to the two-level version for physical masses. The scaling behavior with respect to the bare mass and the lattice size shows great potential for future lattice QCD computations on

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

REFERENCES

[1] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, T. A. Manteuffel, S. F. McCormick, J. C. Osborn, and C. Rebbi, *Adaptive multigrid algorithm for the lattice Wilson-Dirac operator*, Phys. Rev. Lett., 105:201602 (2010).

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

[3] S. Borsanyi, S. Dürr, Z. Fodor, J. Frison, C. Hoelbling, et al., *Isospin splittings in the light baryon octet from lattice QCD and QED*, arXiv:1306.2287, (2013).

[4] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG.*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.

[5] 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).

[6] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM Review, 47 (2005), pp. 317–346.

[7] CLS, *Coordinated lattice simulation.* `https://twiki.cern.ch/twiki/bin/view/CLS/`.

[8] S. Durr, Z. Fodor, C. Hoelbling, S. Katz, S. Krieg, et al., *Lattice QCD at the physical point: Simulation and analysis details*, JHEP, 08(2011)148 (2011).

[9] S. Durr, Z. Fodor, C. Hoelbling, S. D. Katz, S. Krieg, T. Kurth, L. Lellouch, T. Lippert, K. K. Szabo, and G. Vulvert, *Lattice QCD at the physical point: Light quark masses*, Phys. Lett. B701, (2011), pp. 265–268.

[10] J. Finkenrath, F. Knechtli, and B. Leder, *One flavor mass reweighting in lattice QCD*, arXiv:1306.3962, (2013).

[11] P. Fritzsch, F. Knechtli, B. Leder, M. Marinkovic, S. Schaefer, et al., *The strange quark mass and Lambda parameter of two flavor QCD*, Nucl. Phys., B865 (2012), pp. 397–429.

[12] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *Adaptive Aggregation Based Domain Decomposition Multigrid for the Lattice Wilson Dirac Operator*, arXiv:1303.1377, (2013).

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

[14] S. Krieg and T. Lippert, *Tuning lattice QCD to petascale on Blue Gene/P*, NIC Symposium 2010, (2010), pp. 155–164.

[15] M. Lüscher, *DD-HMC algorithm for two-flavour lattice QCD.* `http://luscher.web.cern.ch/luscher/DD-HMC`, used version: DD-HMC-1.2.2, September 2008.

[16] ———, *openQCD simulation program for lattice QCD with open boundary conditions.* `http://luscher.web.cern.ch/luscher/openQCD/`, used version: openQCD-1.2, May 2013.

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

[18] ———, *Local coherence and deflation of the low quark modes in lattice QCD*, JHEP, 07(2007)081 (2007).

[19] Y. Notay and P. S. Vassilevski, *Recursive Krylov-based multigrid cycles*, Numerical Linear Algebra With Applications, 15 (2008), pp. 473–487.

[20] J. Osborn, R. Babich, J. Brannick, R. Brower, M. Clark, et al., *Multigrid solver for clover fermions*, PoS, LATTICE2010:037 (2010).

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

[22] 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.

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