PROCEEDINGS OF SCIENCE

Optimizing Staggered Multigrid for Exascale performance

Venkitesh Ayyar,[a,*] **Richard Brower,**[a] **M.A. Clark,**[b] **Mathias Wagner**[b] **and Evan Weinberg**[b]

[a] *Boston University, Boston, MA 02215, USA*
[b] *NVIDIA Corporation, Santa Clara, CA 95051, USA*

E-mail: vayyar@bu.edu

Adaptive multi-grid methods have proven very successful in dealing with critical slow down for the Wilson-Dirac solver in lattice gauge theory. Multi-grid algorithms developed for Staggered fermions using the Kähler-Dirac preconditioning [9] have shown remarkable success. In this work, we discuss the performance of this staggered multi-grid algorithm in four dimensions. We also demonstrate that offloading some components of a multi-shift solve to a multi-grid solver leads to a significant performance improvement in an existing MILC spectrum workflow on the Summit and Selene supercomputers.

The 39th International Symposium on Lattice Field Theory,
8th-13th August, 2022,
Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn, Germany

*Speaker

PoS(LATTICE2022)335

1. Introduction

The era of exascale computing has finally arrived. After years of improvement and innovation on the hardware level, as well as innovations in software and algorithms which more optimally utilize such hardware, we are now capable of running calculations that exceed one exaflop of computing throughput. This accomplishment is indicative of a commensurate boost in all computing workflows, and by extension a growth in scientific challenges we can pursue with a computational approach.

Unfortunately, raw flops do not alone unlock super-linear growth in the number of problems we can tackle. Cutting-edge problems of interest do not scale trivially with the increase in computing power. The path towards more precise computational science suffers from a phenomena known as critical slowing down [4], where the cost of meaningful computation scales with a higher exponent than the naïve arithmetic complexity of the algorithms in use.

One of the clearest indicators of critical slowing down in lattice gauge theory simulations is non-trivial increases in the costs of one key kernel in most workflows: the iterative solution of the Dirac linear equation as part of propagator calculations and (rational) hybrid Monte Carlo (RHMC) evolution [14, 16]. In the approach to the continuum limit, the number of iterations required to solve the Dirac linear system to a fixed tolerance grows super-linearly. This is best understood in terms of the non-linear increase in the condition number of the system as the lattice spacing a decreases.

There are many methods which stave off but do not solve critical slowing down. One is block-Krylov solvers, which improve reuse across multiples solves but do improve the condition number of the matrix [11, 23]. A second is eigenvalue deflation [20] and improvements thereof in [13, 27], to name a few, which improves the condition number of the matrix by performing an exact solve of the low space. This approach thus shifts the issue of critical slowing down to the eigensolve. The one approach that solves critical slowing down in a scalable fashion is a multi-grid (MG) algorithm.

Adaptive multi-grid preconditioning with geometric aggregation have shown remarkable success in mitigating critical slowing down in lattice gauge theory applications. The first success was with the Wilson and Wilson-clover formulations [3, 7, 25] (complimented by the similar algorithm of inexact deflation in [22]). There have also been successful extensions to twisted-mass and -clover fermions [18, 26], as well as chiral domain wall fermions [6, 8].

A remaining challenge is the development and deployment of an MG algorithm for Kogut-Susskind, or staggered, fermions [19]. The mathematical framework of a staggered MG algorithm was developed in two dimensions in [9]. In this work, we describe the extension and implementation of an MG algorithm for staggered fermions in four dimensions. We describe the details of the implementation and share performance on the Summit Supercomputer at Oak Ridge National Laboratory, as well as the Selene Supercomputer at NVIDIA. Last, we share thoughts on future directions in the development and optimal implementation of multi-grid solvers.

2. Multi-grid methods for Staggered fermions

2.1 Multi-grid algorithms

At a high level, the idea of an MG approach is to solve, in tandem, modes at all scales in a given Dirac operator. We solve this as a *K-cycle*, where the MG preconditioner is the preconditioner in an outer (flexible) Krylov solver; here we use generalized conjugate residual (GCR) [2]. The preconditioner is implemented as follows: (1) pre-smooth the current residual, (2) "restrict," or

aggregate, the current residual with a restriction operator to the coarser level, (3), iterate on the coarser level linear system to some fixed tolerance, (4), "prolongate" the error correction from the coarser level to the fine level, updating the solution, and (5) post-smoothing the new residual. This is extended to a recursive algorithm by applying an MG preconditioner to the iterative solve on the coarser level. This is described in greater detail in [9].

The restriction operator R is formed from block-orthonormalized *near-null vectors*, which are vectors rich in low modes. These vectors are generated via inverse iterations on the *homogeneous system*, $A\vec{\phi}_0 = \vec{0}$, where $\vec{\phi}_0$ is seeded with random numbers. After a large number of iterations the solution vector will be rich in low modes. Each near-null vector is chirally doubled, preserving a form of "γ_5"-Hermiticity on the coarse level; the staggered chiral projector is defined in the following subsection. The process of block orthonormalization, where the block size defines the aggregation factor, increases the span of the fine space which is preserved on a coarser level.

Given a restriction operator R, we define the projection operator P via a Galerkin prescription, $P = R^\dagger$. The coarse operator is explicitly constructed as $\hat{A} = RAP$ where the $\hat{\cdot}$ notation denotes the "coarsened" version.

2.2 Staggered fermion and Kähler-Dirac preconditioning fundamentals

The lattice staggered fermion formulation [19] trades some residual fermion doubling for an exact lattice chiral symmetry and is equivalent to the Kähler-Dirac formulation [5] in the free field, where there is a one-to-one equivalence between the 2^d hypercube of degrees of freedom for staggered fermions and the 2^d degrees of freedom for Kähler-Dirac fermions. There is an exact chiral symmetry generated by $\gamma_5 \otimes \tau_5 = \epsilon(x) = (-1)^{x+y+\cdots}$, where γ and τ denote the spin and taste space, respectively, and thus the chiral projector is defined by $\frac{1}{2}(1 \pm \epsilon(x))$. The eigenspectrum of the staggered operator is maximally anti-Hermitian indefinite up to a real mass shift.

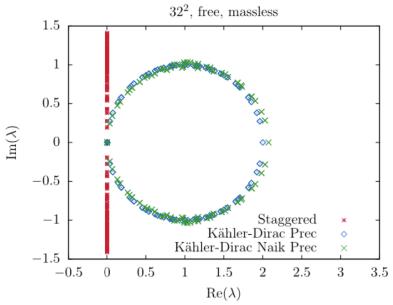
The benefits of the physics of the staggered fermion turns into a challenge for staggered fermion MG. In four dimensions, the four-fold increase in the number of degrees of freedom of a staggered fermion relative to a Wilson fermion increases the size of the necessary near-null space, with a commensurate increase in computational complexity. More fundamental is that the general adaptive MG algorithm described above, which is successful for the Wilson fermion formulation, fails due to *spurious small eigenvalues* in the coarsened staggered operator.

It was found in [9] that there is a solution to this problem: a so-called Kähler-Dirac (KD) preconditioner, which deforms the eigenspectrum of the staggered operator to a form more amendable to MG preconditioning. We consider the unimproved staggered Dirac operator as a sum of two contributions: the hopping terms that stay *within* the 2^d hypercube, denoted B, and the hopping terms that *connect* 2^d hypercubes, denoted C. In the language of the equivalent Kähler-Dirac operator, these are the internal degrees of freedom as opposed to the matrix elements connecting sites. The operator can be written as:

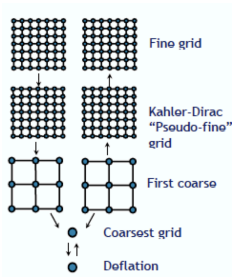
$$D_{\text{stag}} = (B + m) + C, \tag{1}$$

which constitutes a *dual-decomposition*. *In the free field*, up to a scaling, the terms $(B + m)$ and C are each anti-Hermitian (up to a real mass shift). Each obeys an identical ϵ-Hermiticity. $B^\dagger B$ and $C^\dagger C$ are proportional to the identity matrix, implying they are each separately *unitary*.

The matrix B is block-local: in the free-field case, each hypercube contains 2^d independent sites, or 2^d degrees of freedom. This makes taking the inverse of B trivial. Kahler-Dirac preconditioning



(a) Representative spectrum of the free-field staggered and KD-preconditioned operator in two dimensions [9].



(b) General schematic of the staggered MG algorithm.

Figure 1: On the left, a comparison of the spectrum of the KD-preconditioned and non-preconditioned staggered operator. The application of the KD preconditioner converts the staggered spectrum into an overlap-esque spectrum. On the right, a visual representation of the structure of the five-level staggered-type MG algorithm. Of note, the second level is the KD-preconditioned operator which still acts on the fine space, and the fifth level is a deflation space.

involves left- (or right-)multiplying D_{stag} with B^{-1} to obtain

$$A = B^{-1} D_{\text{stag}} = \mathcal{I} + B^{-1} C = \mathcal{I} + (\epsilon B)^{-1} (\epsilon C) \,. \tag{2}$$

Since ϵB and ϵC are each unitary in the free field, their product is also unitary, and thus the final Kähler-Dirac preconditioned operator has a shifted-circular, a.k.a. overlap-esque, structure. This is shown on the left-hand side of Fig. 1. We perform an MG aggregation on this operator.

The observations we make here break down in the interacting case as neither operator is unitary. Further, in the four-dimensional case, the more relevant operator is the HISQ [17, 24] operator. Here the dual decomposition breaks down due to the introduction of a Naik term. However, everything noted above remains *approximately* true in these cases, so we put our blinders on and carry on with the KD-preconditioned operator.

2.3 Kähler-Dirac preconditioned multigrid

We now describe the high-level implementation of our algorithm. A complementary sketch of the process is given in the right-hand side of Fig. 1. First, our outer-most operator is the full HISQ stencil containing the fat and long links. The next recursive level is the Kähler-Dirac preconditioned operator. This is an abuse of language since this next "level" does not include a thinning of degrees of freedom. We nonetheless insert it into the same algorithmic structure—including a pre- and post-smoother surrounding it—and denote this level as "pseudo-fine" to reflect its nature.

Subsequent levels are constructed as expected: generate near-null vectors, perform a chiral doubling by applying $\frac{1}{2}(1 \pm \epsilon(x))$, perform a block orthonormalization, construct a prolongator, restrictor, and coarse operator, then recurse. As described in previous publications, we take the non-traditional approach of generating near-null vectors with the coarse operator and then coarsening the *block-preconditioned* version of said operator. In general this *improves* the condition number on each level, analogous to even-odd preconditioning for the Wilson-clover operator [15].

As a final optimization, we perform a deflation with the low-lying singular values on the coarsest level. This is inspired by deflation of a Hermitian positive definite operator in [28], however, the key

innovation in the QUDA library is using singular value decomposition deflation instead. The use of deflation in the workflow is an acknowledgement that recursing to an arbitrarily small problem size can be inefficient. This means that we do *not* have a perfect MG algorithm, and there is residual critical slowing down on the coarsest level. The deflation addresses this, with the benefit that because it is a much smaller operator, generating the eigenspace is more efficient than generating the eigenspace for the fine operator.

3. Implementation and Results

3.1 Details of the implementation

For our investigations here, we utilize the efficient implementation of MG workflows in the QUDA library for GPUs [12]. Originally developed for NVIDIA GPUs, it now features a performance-portable abstraction that includes formal support for HIP to target AMD GPUs, as well as near-complete support for SYCL and work-in-progress support for OpenMP device targets.

QUDA's solvers utilize reduced-precision methods and gauge-link compression based on their inherent symmetries. HISQ long links, being in the $U(3)$ group, can be compressed as 9 or 13 real numbers. Conjugate Gradient (CG) is run in mixed precision, here mixed double/half, where "half" refers to a 16-bit fixed-point format with a per-site fp32 norm. The staggered and HISQ operators use GPU-initiated communications via NVSHMEM support when run on NVIDIA GPUs.

The coarse stencil operator is also highly optimized; the coarse links are stored in 16-bit fixed point format, and additional parallelism is exposed in the matrix-vector application to compensate for reduction in parallelism possible by the size of the local volume alone. Halo exchanges for the coarse spinor fields utilize NVSHMEM packing kernels on NVIDIA GPUs. This is an important optimization given that communications on the coarse level are inherently latency limited.

Our implementation of the KD-preconditioned operator has evolved significantly over the course of this work. We initially performed a unitary transformation of the staggered operator, acting on all V sites on the lattice with $N_c = 3$ degrees of freedom per site, into an operator acting on $\frac{V}{2^d}$ "super-sites" with 2^d times $N_c = 3 \longrightarrow 48$ degrees of freedom. The stencil of this operator included many zeroes in deterministic locations. Next, we constructed the inverse of the block-local term B and pre-applied it to each "hopping" term. In the HISQ case, distance-three Naik terms on the KD-preconditioned operator were dropped. The justification being perturbative: the leading coefficient of the Naik term is 5% of the magnitude of the fat link terms. This required storing nine 48^2 matrices per 2^d super-site: eight for the pre-computed hopping terms and one for the KD inverse term $(B+m)^{-1}$. This corresponds to $9 \times 48^2/16 = 1296$ complex numbers per fine site.

The revised approach is to apply the KD operator in a form that takes the decomposition $A = (B+m)^{-1} D_{\text{stag}}$ literally. First, we apply the HISQ operator, taking full advantage of the optimizations already present in the QUDA. Next, we apply the local KD operator in a separate kernel. It is a point of future optimization to fuse these operations.

In contrast to the initial brute-force implementation, the optimized operator requires loading the existing fat and long 3×3 links in four directions per fine site on the V-sized lattice, along with the same explicit $(B+m)^{-1}$ matrix per $\frac{V}{2^d}$ sites. This corresponds to $8 \times 3^2 + 48^2/16 = 180$ complex numbers per fine site: a 72% reduction. An extra benefit is the fine links do not require additional storage because they can be reused from the outer HISQ operator.

Level	Property	Value
0	Solver	MG-preconditioned GCR
0	Operator	Full (non-Schur) HISQ operator
0	Degrees of freedom	$N_c = 3$
0	Solver convergence	10^{-10}
0	Pre-, post-smoother	none, CA-GCR(8)
1	Solver	MG-preconditioned GCR
1	Operator	KD-preconditioned (truncated) HISQ
1	Degrees of freedom	$N_c = 3$
1	Solver convergence	0.25 **or** 8 iterations
1	Pre-, post-smoother	none, CA-GCR(8)
$1 \rightarrow 2$	Setup solver	CGNE
$1 \rightarrow 2$	Setup tolerance	10^{-6} **or** 2000 iterations
$1 \rightarrow 2$	Aggregate size	4 x 6 x 6 x 6

Level	Property	Value
2	Solver	MG-preconditioned GCR
2	Operator	Schur-preconditioned
2	Degrees of freedom	$N_c = 64$, $N_s = 2$
2	Solver tolerance	0.25 **or** 8 iterations
2	Pre-, post-smoother	none, CA-GCR(8)
$2 \rightarrow 3$	Setup solver	CGNE
$2 \rightarrow 3$	Setup tolerance	10^{-6} **or** 2000 iterations
$2 \rightarrow 3$	Aggregate size	3 x 2 x 2 x 3
3	Solver	SVD-deflated CA-GCR(16)
3	Operator	Schur-preconditioned
3	Degrees of freedom	$N_c = 96$, $N_s = 2$
3	Singular vectors	1024
3	Deflation acceleration	Yes, 400-degree Chebyshev polynomial

Table 1: Key parameters for the MG preconditioner.

One benefit to this formulation is we have the flexibility to preserve or drop the Naik contribution. When we preserve the Naik term, the KD-preconditioned operator is an exact left preconditioning of the full HISQ operator. When we truncate the Naik contribution, it becomes a less effective preconditioner on paper, however it shows comparable, if not better, performance during runs at scale due to a factor-of-two reduction in memory traffic during the stencil application and a factor-of-three reduction in data communicated.

In the following subsection we present results from solving the HISQ linear system with an MG preconditioner. We only present results for the optimized implementation of the full HISQ operator and the truncated operator because the memory overheads of the brute force operator lead to at least a doubling of the number of required nodes.

3.2 Performance comparison of Multigrid vs Conjugate Gradient

While the algebraic description above suggests a successful MG algorithm, it does not prove the viability of the technique in production workflows. The one true metric of success is an improvement in *time to solution* for propagator solves.

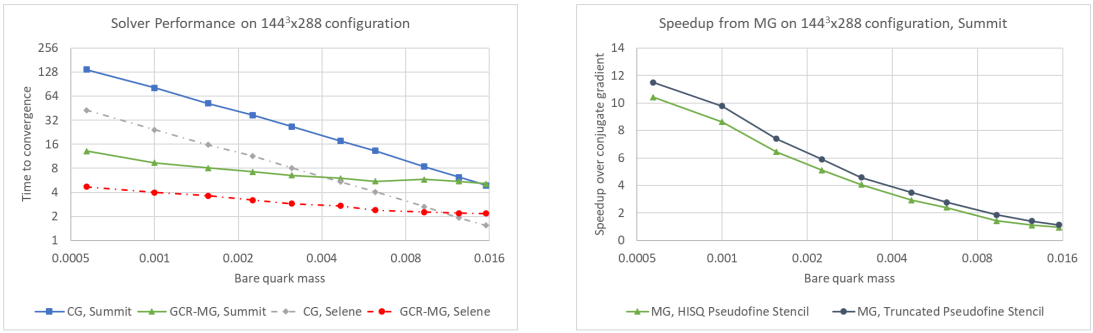
For our studies here, we consider a $144^3 \times 288$, 2+1+1 physical pion mass HISQ configuration shared by the MILC collaboration. The configuration is very fine, with a lattice spacing of 0.04 fm, and has a bare light and strange quark mass of 0.000569 and 0.01555, respectively.

We distribute the calculation over 864 GPUs. We perform a $6 \times 3 \times 6 \times 8$-way partitioning of the global volume, leading to a per-GPU local volume of $24 \times 48 \times 24 \times 36$. For the Summit supercomputer, with 6 NVIDIA V100 GPUs per node, this corresponds to 144 nodes, and for the Selene supercomputer, with 8 NVIDIA A100 GPUs per node, we use 108 nodes.

Many parameters go into tuning an MG solve between the near-null space generation, coarsest-level SVD deflation, and the parameters of the MG solve itself. We summarize the parameters of our HISQ workflow in Table 1. We note that for the smoother and for the coarsest-level solve we have developed a communication-avoiding s-step version of GCR, denoted CA-GCR, an extension of the idea developed in [10] for generalized minimum residual (GMRES).

In Fig. 2, we show the solver performance of our staggered MG algorithm compared with the typical Schur-preconditioned CG solve for the HISQ stencil. In all cases we performed the solve to a relative tolerance of 10^{-10} on the full (non-Schur) operator. In the case of the single-parity CG solve, this requires a solve to the relative tolerance $m \times 10^{-10}$.

On the left, we show the performance of CG and MG-preconditioned GCR on both the Summit and Selene supercomputers. In this case, we are using the full HISQ stencil in the KD-preconditioned



(a) Comparison of timings for MG and CG runs

(b) Ratio of timings from MG to CG on Summit

Figure 2: On the left, the time to solution for conjugate gradient and MG-preconditioned GCR on Summit (solid lines) and Selene (dashed lines). The MG-preconditioned solve uses the full HISQ stencil in the pseudo-fine operator. On the right, the relative performance of MG-preconditioned solves—with the full HISQ stencil and the truncated stencil in the pseudo-fine operator—to a standalone CG solve on Summit.

operator, that is, we do not drop the Naik term. While the cost of the CG solve scales with the inverse of the mass, corresponding to the expected scaling of the condition number, we see that the MG-preconditioned solve shows only a slight dependence on the mass. This is an indication of successful removal of critical slowing down.

On the right, we show the relative performance of our MG algorithm against a CG solve on Summit. Here, we present two curves: the performance boost for the full HISQ stencil in the KD-preconditioned operator (green triangles), and the performance boost for the *truncated* stencil in the KD-preconditioned operator (black circles). For the smallest mass, $m = 0.000569$, the performance gain is more than 10x for both the full and the truncated operator. Perhaps even more impressively, there is rough performance parity at the strange quark mass, $m = 0.01555$.

There is an important note to go along with this data. We generate the near-null vectors and singular vectors on the coarsest level *once* using the operators formed from the smallest mass, $am = 0.000569$. We save these and reuse them for each subsequent solve. This approach is an emulation of a propagator workflow, where the costs of re-generating near-null vectors and singular vectors, along with re-formulating the coarse operator, could become end-to-end prohibitive.

We note that these numbers do not include the setup costs as that is a point of active optimization.

3.3 Multigrid performance on a MILC workflow on Summit

We see in section 3.2 that the staggered MG algorithm in four dimensions offers a very significant benefit over CG for light masses. However, it is important to understand how useful this algorithm in actual lattice gauge theory workflows with multiple masses and greater complexity. We consider a workflow that involves computing propagators for ten different quark masses, ranging from the physical light quark to the strange quark. Typically this workflow is run using *multi-shift CG*, where the propagator for each mass shift is solved simultaneously with the overhead of only one matrix-vector application per iteration.

In the most general case it is challenging to formulate a preconditioned multi-shift CG. This makes applying the MG preconditioner described here non-trivial. The simplest, and effective, solution is to apply an MG preconditioner to the smallest subset of masses, where there is the

greatest benefit [1]. We reuse the coarse operators from the lightest mass for other masses in this subset. The remaining masses, equivalently shifts, are still solved in what is now a much less expensive multishift CG. For this workflow, we split the ten masses 3:7 between MG and multi-shift, an empirical decision that will be better explored in the future.

Table 2 gives a comparison of normalized timings for MG and CG on the Summit supercomputer, where we have used 144 nodes as described above for MG, and 72 nodes for CG, owing to the reduced memory overheads. The propagator time corresponds to the time it takes to solve for three right hand sides for an MG solve, and six for CG solves—three for the even system, three for the odd. We see that, when normalized by node-hours, the cost of MG setup is amortized after roughly three propagator calculations.

Run type	Summit Nodes	MG Setup	One Propagator
Units	—	Node-hours	Node-hours
Multishift CG	72	—	9.87×10^4
MG+CG (full op)	144	2.28×10^5	3.34×10^4
MG+CG (truncated op)	144	1.89×10^5	3.37×10^4

Table 2: Performance comparison of mixed MG and multishift CG **vs** just multishift CG for a propagator workflow with 10 masses ranging from the light to the strange quark. The break-even point for the full pseudo-fine operator is ≈ 3.5 propagators, while for the truncated pseudo-fine operator it is ≈ 2.9 propagators.

4. Conclusion

4.1 Summary

In this work we have described a successful implementation of a multi-grid solver for the HISQ operator in four dimensions. More importantly, we have demonstrated the success of the algorithm on a very fine $144^3 \times 288$ physical pion mass HISQ configuration, showing the near-elimination of critical slowing down across a range of masses from the physical light up to the strange quark mass. Using the QUDA library for lattice calculations on GPUs, we have demonstrated a 10x speedup in solver time to a fixed tolerance at the light quark mass for a solve distributed over 144 nodes of the Summit supercomputer, and shown we can break even in solve time at the strange quark mass.

Due to the performance portability work that went into QUDA, we have also successfully run this algorithm on a smaller configuration on the Crusher supercomputing testbed at ORNL.

4.2 Future directions

A key challenge of the current algorithm is the high overhead of generating several near-null vectors for the HISQ operator due to the large number of low modes inherent to the staggered formulation. We are currently exploring multiple avenues to improve the setup time, including but not limited to: multi-right hand side solver methods, Chebyshev filter approaches to generating near-null vectors [6], and block-TRLM methods for singular vector generation.

Although this algorithm has a clear use case in the measurement of correlation functions in lattice gauge theories, there is potential for using it for gauge generation. One challenge in using it in HMC is that we must evolve the near-null vectors as the gauge field evolves. This has been successfully applied in the Wilson-clover case [21] and the Shamir domain wall case [6], however the relatively higher setup costs for HISQ MG and the presence of multi-shift solves in RHMC make this less trivial. These studies will be a follow-up to the improved setup investigations above.

Acknowledgments

This work was supported in part by the U.S. Department of Energy (DOE) under Award No. DE-SC0015845 and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. For computation, we used the Summit and Crusher supercomputers at Oak Ridge National Laboratory.

References

[1] C. Alexandrou, S. Bacchio, and J. Finkenrath, *Multigrid Approach in Shifted Linear Systems for the Non-Degenerated Twisted Mass Operator*, Computer Physics Communications, 236 (2019), pp. 51–64.

[2] O. Axelsson, *A Generalized Conjugate Gradient, Least Square Method*, Numerische Mathematik, 51 (1987), pp. 209–227.

[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] T. Blum, R. S. Van de Water, D. Holmgren, R. Brower, S. Catterall, et al., *Working Group Report: Lattice Field Theory*, (2013).

[5] G. T. Bodwin and E. V. Kovács, *Equivalence of Dirac-Kähler and Staggered Lattice Fermions in Two Dimensions*, Phys. Rev. D, 38 (1988), pp. 1206–1219.

[6] P. Boyle and A. Yamaguchi, *Comparison of Domain Wall Fermion Multigrid Methods*, arXiv e-prints, (2021), p. arXiv:2103.05034.

[7] J. Brannick, R. C. Brower, M. A. Clark, J. C. Osborn, and C. Rebbi, *Adaptive Multigrid Algorithm for Lattice QCD*, Phys.Rev.Lett., 100 (2008), p. 041601.

[8] R. C. Brower, M. A. Clark, D. Howarth, and E. S. Weinberg, *Multigrid for Chiral Lattice Fermions: Domain wall*, Phys. Rev. D, 102 (2020), p. 094517.

[9] R. C. Brower, M. A. Clark, A. Strelchenko, and E. Weinberg, *Multigrid Algorithm for Staggered Lattice Fermions*, Phys. Rev. D, 97 (2018), p. 114513.

[10] A. T. Chronopoulos and S. K. Kim, *s-Step Orthomin and GMRES Implemented on Parallel Computers*, arXiv:2001.04886, High Energy Physics - Lattice, (2020).

[11] M. Clark, A. Strelchenko, A. Vaquero, M. Wagner, and E. Weinberg, *Pushing memory bandwidth limitations through efficient implementations of block-krylov space solvers on gpus*, Computer Physics Communications, 233 (2018), pp. 29–40.

[12] M. A. Clark, B. Joó, A. Strelchenko, M. Cheng, A. Gambhir, and R. C. Brower, *Accelerating lattice qcd multigrid on gpus using fine-grained parallelization*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '16, IEEE Press, 2016.

[13] M. A. Clark, C. Jung, and C. Lehner, *Multi-Grid Lanczos*, in 35th International Symposium on Lattice Field Theory (Lattice 2017) Granada, Spain, June 18-24, 2017, 2017.

[14] M. A. Clark and A. D. Kennedy, *Accelerating Dynamical-Fermion Computations Using the Rational Hybrid Monte Carlo Algorithm with Multiple Pseudofermion Fields*, Physical Review Letters, 98 (2007).

[15] T. A. DeGrand and P. Rossi, *Conditioning Techniques for Dynamical Fermions*, Comput. Phys. Commun., 60 (1990), pp. 211–214.

[16] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, *Hybrid Monte Carlo*, Physics Letters B, 195 (1987), pp. 216–222.

[17] E. Follana, Q. Mason, C. Davies, K. Hornbostel, G. P. Lepage, J. Shigemitsu, H. Trottier, and K. Wong, *Highly Improved Staggered Quarks on the Lattice, with Applications to Charm Physics*, Phys. Rev., D75 (2007), p. 054502.

[18] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *Adaptive Aggregation-Based Domain Decomposition Multigrid for the Lattice Wilson–Dirac Operator*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1581–A1608.

[19] J. Kogut and L. Susskind, *Hamiltonian Formulation of Wilson's Lattice Gauge Theories*, Phys. Rev. D, 11 (1975), pp. 395–408.

[20] C. Lanczos, *An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators*, Journal of research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[21] M. Lüscher, *Deflation Acceleration of Lattice QCD Simulations*, Journal of High Energy Physics, 2007 (2007), pp. 011–011.

[22] M. Lüscher, *Local coherence and deflation of the low quark modes in lattice qcd*, Journal of High Energy Physics, 2007 (2007), p. 081.

[23] A. A. Nikishin and A. Y. Yeremin, *Variable Block CG Algorithms for Solving Large Sparse Symmetric Positive Definite Linear Systems on Parallel Computers, I: General Iterative Scheme*, SIAM Journal on Matrix Analysis and Applications, 16 (1995), pp. 1135–1153.

[24] K. Orginos, D. Toussaint, and R. L. Sugar, *Variants of Fattening and Flavor Symmetry Restoration*, Phys. Rev., D60 (1999), p. 054503.

[25] J. C. Osborn, R. Babich, J. J. Brannick, R. C. Brower, M. A. Clark, S. D. Cohen, and C. Rebbi, *Multigrid solver for clover fermions*, arXiv:1011.2775, High Energy Physics - Lattice, (2010).

[26] D. Richtmann, N. Meyer, and T. Wettig, *MRHS Multigrid Solver for Wilson-Clover Fermions*, PoS, LATTICE2022 (2022).

[27] E. Romero, A. Stathopoulos, and K. Orginos, *Multigrid Deflation for Lattice QCD*, Journal of Computational Physics, 409 (2020), p. 109356.

[28] A. Yamaguchi and P. Boyle, *Hierarchically deflated conjugate residual*, PoS, LATTICE2016 (2016), p. 374.