PROCEEDINGS OF SCIENCE

Performance Optimization of Baryon-block Construction in the Stochastic LapH Method

Phuong Nguyen[a,b,*] **and Ben Hörz**[a]

[a] *Intel Deutschland GmbH, Dornacher Str. 1, 85622 Feldkirchen, Germany*
[b] *Technical University Munich, Boltzmannstr. 3, 85748 Garching, Germany*

E-mail: phuong.nguyen@tum.de, ben.hoerz@intel.com

Implementations of measurement kernels in high-level Lattice QCD frameworks enable rapid prototyping, but can leave hardware capabilities significantly underutilized. This is an acceptable tradeoff if the time spent in unoptimized routines is generally small. The computational cost of modern spectroscopy projects however can be comparable to or even exceed the cost of generating gauge configurations and computing solutions of the Dirac equation. One such key kernel in the stochastic LapH method is the computation of baryon blocks; we discuss several implementation strategies and achieve a 7.2x speedup over the current implementation on a system with Intel® Xeon® Platinum 8358 processors, formerly Ice Lake.

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

PoS(LATTICE2022)343

[*] Speaker

1. Introduction

With the advent of modern spectroscopy methods for multi-hadron systems [1–3], ever more complicated physical systems are coming into reach. While a variety of two-meson systems including with several coupled channels have been investigated (see [4] for a plenary review at this conference), and more recently even studies of three-meson systems have started appearing [5–12] (also subject of a topical review this year [13]), the situation for systems with baryons is comparably less advanced. Hadron interactions involving baryons are however among the fundamental building blocks for an understanding of nuclear physics rooted in QCD (for a review, see for instance [14]). Examples include two-nucleon interactions, which – when determined at sufficiently light pion mass – can serve as a validation system to establish the reliability of these kinds of lattice QCD calculations [15–18]; three-nucleon interactions, which are difficult to access phenomenologically and hence present a great opportunity for lattice QCD to provide useful data; a variety of meson-baryon systems, for instance nucleon-pion scattering in the isospin $I = 3/2$ channel featuring the $\Delta(1232)$ resonance [19–21].

Even though modern spectroscopy methods present no conceptual difficulties generalizing to systems involving baryons, there are a few challenges in practice. Baryonic systems typically suffer from a worse signal-to-noise ratio than purely mesonic systems, requiring larger amounts of statistics to obtain meaningful results. In addition to necessitating more correlation-function samples, every individual sample tends to be more computationally expensive compared to the mesonic sector due to the increased number of quark fields. At the correlator-construction level, algorithms eliminating redundant computations have been devised to alleviate the proliferation of Wick contractions [5, 22–24].

This work is concerned with improving the efficiency of the computation of baryon functions in the stochastic LapH method [2]. In the stochastic LapH framework, baryon blocks are rank-three tensors in dilution indices, carrying additional labels identifying the baryon operator (flavor, spin, hadron momentum) as well as the three noises used for the stochastic estimate of the quark propagators in the LapH subspace. Correlators are then computed through tensor contractions over dilution indices of those baryon functions as governed by Wick's theorem. This beneficial property of the stochastic LapH method – the evaluation of complicated multi-hadron correlation functions is reduced to tensor contractions involving blocks representing the constituent hadrons – enables the re-use of baryon blocks for a wide variety of physical systems. The optimizations presented in this work are hence immediately applicable to a breadth of calculations involving baryons.

This contribution is organized as follows: section 2 defines the baryon-block kernel and discusses its computational characteristics, section 3 contrasts several implementation strategies, and benchmark results are presented in section 4.

2. Baryon blocks in the stochastic LapH method

The stochastic LapH method [2] is a stochastic variant of distillation [1] which avoids the V^2 scaling with the spatial simulation volume V by stochastically estimating the quark propagator projected into the LapH (or distillation) subspace. A useful quantity for the computation of multi-hadron correlation functions involving baryons is the (single-site) baryon function defined per time slice of the simulation volume,

$$B^{(\vec{p},\Lambda,\mu,\eta)}_{d_1 d_2 d_3} = c^{(\vec{p},\Lambda,\mu)}_{\alpha\beta\gamma} \sum_{\vec{x}} \mathrm{e}^{-\mathrm{i}\,\vec{p}\vec{x}} \epsilon_{abc} q^{(\eta_1,d_1)}_{\alpha a\vec{x}} q'^{(\eta_2,d_2)}_{\beta b\vec{x}} q''^{(\eta_3,d_3)}_{\gamma c\vec{x}}, \tag{1}$$

with color indices a, b, c, and the summation runs over the sites of a three-dimensional time slice of the lattice. The relevant combinations of spin indices α, β, γ are selected according to the group-theoretical projection coefficients $c^{(\vec{p},\Lambda,\mu)}_{\alpha\beta\gamma}$ for a given hadron momentum $\vec{p}$, irrep Λ and irrep row μ [25]. The quark fields q, q', q'', which have been projected into the LapH subspace, are obtained by repeatedly solving the Dirac equation with diluted stochastic sources identified by a noise label η and dilution index $d_1 = 1, \ldots, N_{\mathrm{dil}}$. Crucially, those solutions of the Dirac equation for a given noise need only be computed once, and can then be cheaply stored on disk and used for various multi-hadron projects by reconstructing the smeared quark fields from their coefficients Q in the basis of eigenvectors of the three-dimensional gauge-covariant Laplacian $\phi^{(l)}$, $l = 1, \ldots, N_{\mathrm{ev}}$, which is used to define the LapH subspace,

$$q^{(\eta,d)}_{\alpha a\vec{x}} = \sum_{l=1}^{N_{\mathrm{ev}}} Q^{(\eta,d)}_{\alpha l} \phi^{(l)}_{a\vec{x}}, \tag{2}$$

and similarly for q' and q'', which differ only in their coefficients Q, but with the same eigenvectors ϕ.

Both the group-theoretical projection involving the spin indices and the bookkeeping of noise indices in (1) are handled by the calling application, leaving

$$B^{(\vec{p})}_{d_1 d_2 d_3} = \sum_{\vec{x}} \mathrm{e}^{-\mathrm{i}\,\vec{p}\vec{x}} \epsilon_{abc} Q^{(1)}_{d_1 l_1} Q^{(2)}_{d_2 l_2} Q^{(3)}_{d_3 l_3} \phi^{(l_1)}_{a\vec{x}} \phi^{(l_2)}_{b\vec{x}} \phi^{(l_3)}_{c\vec{x}}, \tag{3}$$

where summation over the eigenvector indices l_1, l_2 and l_3 is implied, as the computational kernel, which is called many times for different noise and spin combinations, i.e. different quark field coefficients, but with the same momentum set and Laplacian eigenvectors.

Depending on the sizes of N_{ev} and N_{dil}, as well as how many times the kernel (3) is called per time slice, one of the following two different approaches is preferable.

For moderate values of N_{ev}, (3) can be efficiently computed using a two-step procedure: During the setup stage, the mode-triplets

$$T^{\vec{p}}_{l_1 l_2 l_3} = \sum_{\vec{x}} \mathrm{e}^{-\mathrm{i}\,\vec{p}\vec{x}} \epsilon_{abc} \phi^{(l_1)}_{a\vec{x}} \phi^{(l_2)}_{b\vec{x}} \phi^{(l_3)}_{c\vec{x}}, \tag{4}$$

which are spin-, noise- and flavor-blind, are computed and kept in memory. All lattice-sized objects in (3) have then been consumed, and the baryon function for a given set of quark-field coefficients can be computed by tensor-contracting them onto the precomputed mode-triplet. Those tensor contractions can be performed with high performance, so the majority of the runtime tends to be associated with the initial setup phase, which needs to be amortized over many kernel invocations. The major drawback of this mode-triplet approach is the need to keep one N_{ev}^3-sized object per momentum in memory[1].

[1] Based on the symmetries of (4), only $\binom{N_{\mathrm{ev}}}{3}$ elements of a mode-triplet are independent. Exploting that symmetry with a sparse storage scheme however complicates the subsequent tensor contractions of quark-field coefficients onto the mode-triplet.

For large number of eigenvectors N_{ev}, a more economical approach is to first reconstruct the quark fields from the coefficients as per (2), and subsequently perform the reduction (3) over sets of lattice-sized objects. The quark-field reconstruction can be efficiently implemented using matrix-matrix multiplication and reduces the complexity of subsequent lattice-sized reductions to N_{dil}^3 (rather than N_{ev}^3 for the mode-triplet approach), which however must be performed for every kernel invocation.

In view of the requirements of baryon calculations in large volumes – such as the E250 ensemble [26] generated by the CLS effort [27, 28], where employing the mode-triplet approach is not feasible[2] – the goal of this work is to provide an efficient implementation of (3), which utilizes the great compute capabilities of modern hardware by exploiting the N_{dil}^3 compute complexity with only linear-in-N_{dil} memory traffic.

3. Implementation details

The quark-field reconstruction can be performed efficiently using matrix-matrix multiplication, for which highly optimized implementations are available for all hardware architectures. Hence, in the following section, we focus on optimizing the baryon-block calculation given the reconstructed quark fields q_1, q_2, q_3.

The optimized algorithm is shown in Algorithm 1. Typically, the number of requested hadron momenta N_{mom} is much smaller than the number of allowed momenta (e.g. $33 \ll 64^3$); therefore using a fast Fourier transform is not beneficial. Hence, the phase factor $e^{-\mathrm{i}\vec{p}\vec{x}}$ for the momentum projection in (3) can be precomputed and re-used for several kernel invocations.

Cache blocking techniques are employed in conjunction with an appropriate data layout to optimize data locality. Blocking is implemented both in the spatial indices x and the three dilution indices d_1, d_2, d_3. The blocking in x allows the kernel to exploit the available inherent input reuse. For example, each block of q_1 input can be kept in the cache and re-used for different *diq* calculations with different q_2 since the input size is small enough to stay in the cache (ll. 8-14). The blocking in d_1, d_2, d_3 enables the kernel to keep the intermediate data (*diq*, *singlet*, *tmpBuf*) in cache and use it for subsequent calculations (ll. 16-22). A suitable data memory layout is (in row-major convention) $N_{\mathrm{dil}} \times N_{\mathrm{BlockX}} \times N_{\mathrm{color}} \times N_{\mathrm{BsizeX}}$ for the input q_1, q_2, q_3, ensuring that the data is accessed contiguously in x for each each color component. Furthermore, this data layout stores the BlockX-sized chunks for the three colors of q_1, q_2, q_3 adjacently, enhancing spatial locality in the calculations of *diq* and *singlet*.

The small matrix-matrix multiplication in l. 23 to compute *tmpBuf(:)*$+= singlet \times phase(:)$, utilizes the Intel® Math Kernel Library (Intel® MKL) with just-in-time (JIT) code generation for small matrices with $m = \mathrm{BsizeD}_1 \times \mathrm{BsizeD}_2 \times \mathrm{BsizeD}_3$, $n = N_{\mathrm{mom}}$ and $k = \mathrm{BsizeX}$. These matrices – $singlet$ and *tmpBuf* – should be sufficiently small to remain in the cache. General-purpose GEMM implementations are typically optimized targeting larger matrix sizes. Thus, for the small matrix-matrix multiplication required here, the Intel® Math Kernel Library (Intel® MKL) with JIT

[2] First results in the mesonic sector presented in [29] used the analogous mode-doublet approach for meson construction, which is still affordable due to its slightly weaker N_{ev}^2 scaling. For the $N_{\mathrm{ev}} = 1536$ employed in that work, the mode triplet on the other hand occupies 1.8 TB of memory already for the moderate number of momenta $N_{\mathrm{mom}} = 33$, clearly making this approach impractical.

Algorithm 1 Cache blocking algorithm with pseudo code

Input: $q_1, q_2, q_3, phase$
Output: $baryon$

```
 1: ! $BlockD_i \leftarrow N_{D_i} \,/\, BsizeD_i \quad (i = 1, 2, 3)$
 2: ! $BlockX \leftarrow N_X \,/\, BsizeX$
 3: function BaryonConstruct($q_1, q_2, q_3, phase$)
 4:     for $BlockD_1$ do in parallel
 5:         for $BlockD_2$ do in parallel
 6:             $tmpBuf \leftarrow 0.$
 7:             for each $BlockX$ do
 8:                 for $d_1 \leftarrow 1$ to $BsizeD_1$ do
 9:                     for $d_2 \leftarrow 1$ to $BsizeD_2$ do
10:                         for $x \leftarrow 1$ to $BsizeX$ do
11:                             $diq(d_1, d_2, :, x) \leftarrow q_1(\tilde{d}_1, :, x) \times q_2(\tilde{d}_2, :, x)$
12:                         end for
13:                     end for
14:                 end for
15:                 for each $BlockD_3$ do
16:                     for $d_3 \leftarrow 1$ to $BsizeD_3$ do
17:                         for each $diq_i$ in $diq$ do
18:                             for $x \leftarrow 1$ to $BsizeX$ do
19:                                 $singlet(d_1, d_2, d_3, x) \leftarrow diq_i \times q_3(\tilde{d}_3, :, x)$
20:                             end for
21:                         end for
22:                     end for
23:                     $tmpBuf(:) \leftarrow tmpBuf(:) + singlet \times phase(:)$     ▷ Intel® MKL JIT GEMM
24:                 end for
25:             end for
26:             $baryon \leftarrow tmpBuf$
27:         end for parallel
28:     end for parallel
29:     return $baryon$
30: end function
```

(lines 4–16) Cache Blocking in N_{D_1}, N_{D_2}, N_{D_3}, N_X

compilation is used to generate target microarchitecture code for the kernel which is optimized for small-sized complex-valued matrix multiplication problems. As the matrix sizes are fixed, the kernel can be produced once and then called many times, amortizing the JIT compilation overhead.

Parallelization for multiple threads is achieved by distributing the work in the loops over dilution-index blocks, (ll. 4-5) . In practice, the loops over BlockD$_1$ and BlockD$_2$ are collapsed into a joint iteration space and parallelized for multiple threads with OpenMP. In this approach, there are no data dependencies since computations of each thread are fully independent. Furthermore, as each block works on an identical amount of data, the load is expected to be well-balanced between threads. Lastly, parallelization is implemented at an outer level, encompassing plenty of work per loop trip to amortize the OpenMP runtime overhead for instance for thread scheduling.

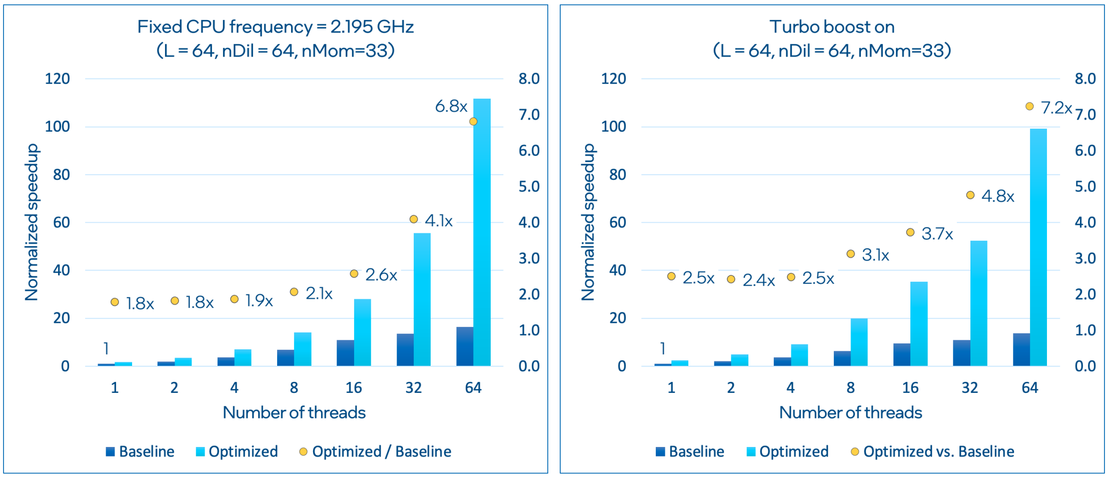

Figure 1: Performance of the previous and optimized kernel normalized to the single-thread performance of the previous implementation. The optimized kernel outperforms the baseline by up to 6.8x for the test with fixed frequency (left) and by up to 7.2x for the test when turbo boost is enabled (right).

4. Performance results

The implementation of Algorithm 1 is evaluated on a test system with two Intel® Xeon® Platinum 8358 processors @ 2.60 GHz for a moderately large problem size of $L = 64$ with $N_{\text{dil}} = 64$ dilution indices per quark field and number of requested momenta $n_{\text{mom}} = 33$. Figure 1 shows the performance of the optimized kernel compared to the previous implementation using 1 to 64 cores.

The block sizes are tunable parameters which generally depend on the target architecture and should be tuned experimentally. For our test node, the block sizes yielding the best performance are BsizeX = 32, $\text{BsizeD}_1 = 4$, $\text{BsizeD}_2 = 8$, $\text{BsizeD}_3 = 16$. This can be understood as aligning the memory footprint of each single intermediate object with the available cache hierarchy. For instance, the *diq* array stores $\text{BsizeD}_1 \times \text{BsizeD}_2 \times N_{\text{colors}} \times \text{BsizeX} = 3072$ complex double-precision values, occupying 48 kB of memory, thus fitting perfectly into the L1 cache. Similarly, the *singlet* has a size of 256 kB which fits into the L2 cache.

When run on a single core, the optimized kernel is 1.8x and 2.5x faster than the baseline in tests with and without turbo boost, respectively. With in-depth profiling, the superior performance of the optimized implementation can be traced back to better use of the memory system, as expected. The optimized kernel operates at 1.6x and 3x higher arithmetic intensity in Read and Write, respectively. In addition, the memory access patterns are also significantly improved such that fewer cycles are spent on load and store operations in the optimized kernel in comparison to the baseline (42% reduction in loads and 70% reduction in stores). The optimized kernel is indicated to be L1-bound instead of DRAM-bound. As a result, with a performance of 37.7 DP GFlops/s, the kernel reaches 54% of the theoretical peak performance.

The improvement of the optimized kernel becomes particularly apparent in multithreaded runs, where it outperforms the baseline by 1.8x to 6.8x at fixed frequency and 2.5x to 7.2x for tests with turbo boost on (Figure 1). This boost in performance can be understood as being due to improved temporal locality. As long as threads progress at a comparable rate, adjacent threads working on a

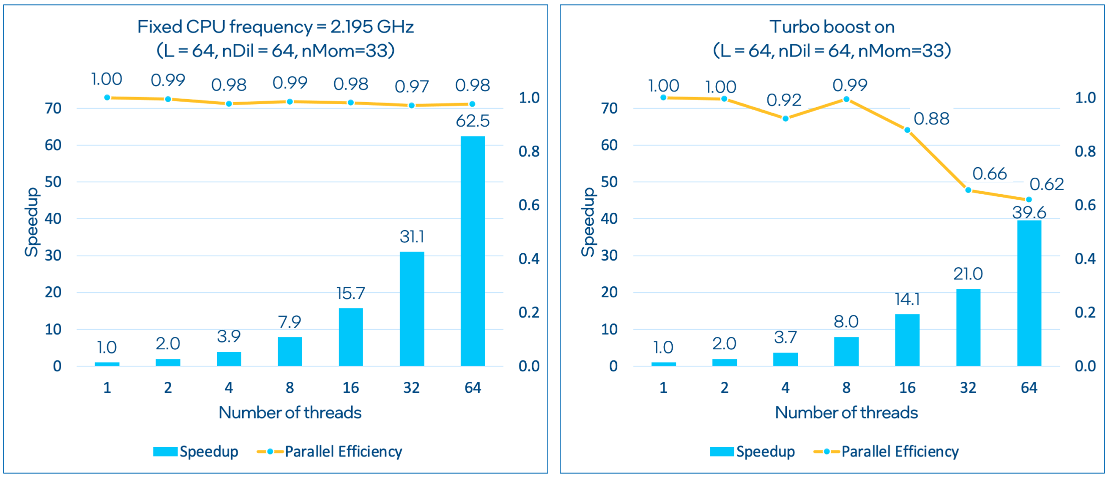

Figure 2: Strong-scaling behavior of the optimized kernel. *Left:* With fixed frequency, the kernel achieves almost perfect parallel efficiency for a 62.5x speedup with 64 threads. *Right:* With turbo enabled, the kernel scales well up to eight cores. Beyond eight cores, the variable frequency gets throttled and the decrease in parallel efficiency matches the decrease in frequency, implying that the optimized kernel is compute-bound.

different BlockD2 but sharing the same BlockD1 access the same input data, which may be served from the cache hierarchy.

Figure 2 shows the strong-scaling behavior of the optimized kernel at fixed clock frequency, achieving almost perfect parallel efficiency at a speedup of 62.5x when using 64 cores. In addition, the parallel efficiency is above 0.98 for all tests, implying that the kernel scales almost perfectly within a node.

For production runs with turbo boost enabled the clock frequency can increase up to the boost frequency (3.6 GHz on our test system). While performance in absolute terms is slightly better than at fixed frequency, the strong-scaling parallel efficiency deteriorates, showing a 39.6x speedup with 64 cores (Figure 2). The loss in parallel efficiency is due to frequency throttling. While for one thread the average frequency is 3.285 GHz, it drops to 2.219 GHz when running with 32 threads. The frequency ratio 0.67 matches the parallel efficiency, indicating that the optimized kernel is compute-bound and indeed limited by the frequency throttling.

The scalability of the optimized kernel with respect to the problem size, of importance in view of ever-increasing simulation volumes, is shown in Figure 3. As is evident from (3), the computational cost scales as $O(L^3)$ and $O(N_{\text{dil}}^3)$ with the spatial volume and number of dilution indices, respectively. The optimized kernel scales as expected to within a few percent as a function of the problem size both with the spatial volume as well as the number of dilution indices.

5. Summary

We have presented an optimized implementation of the kernel computing baryon blocks in the stochastic LapH method, achieving an up to 7.2x speedup over the previos implementation. Exploiting the high arithmetic intensity of (3) by blocking in dilution and spatial indices, and performing the momentum projection with a JIT-compiled microkernel provided by the Intel®

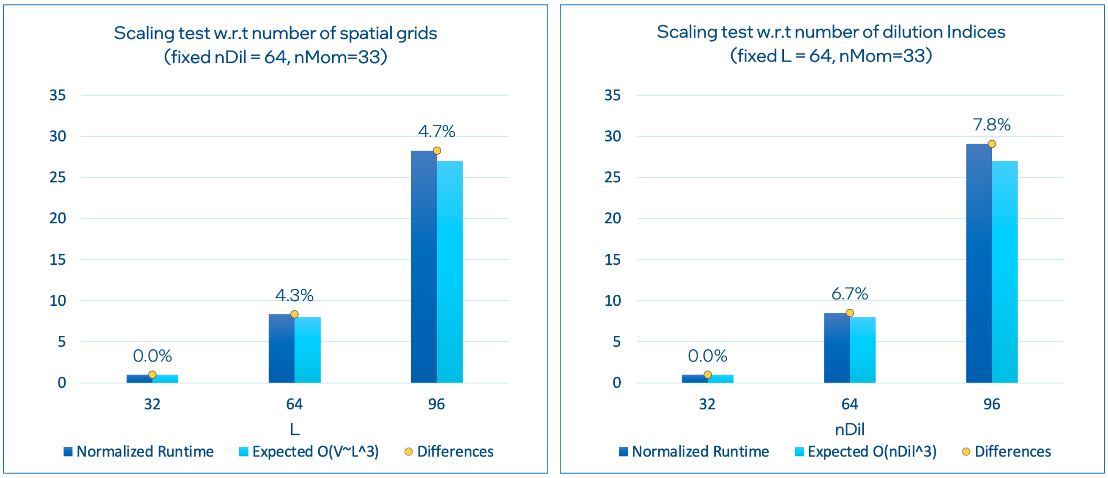

Figure 3: Scaling performance of the optimized kernel with respect to the problem size. The kernel scales as expected to within a few percent both with the spatial volume and the number of dilution indices.

Math Kernel Library (Intel® MKL), we achieve good single-core performance on a test system with Intel® Xeon® Platinum 8358 processors. Parallelizing over blocks of dilution indices using multithreading, we also observe good scalability all the way to the maximum number of cores per socket[3].

This optimized implementation has been upstreamed into the `chroma_laph` measurement suite and is ready for use in production runs on large lattice volumes.

Acknowledgments

We acknowledge useful conversations with Colin Morningstar as well as our Intel colleagues Christoph Bauinger and Carsten Uphoff. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries.

References

[1] Hadron Spectrum collaboration, *A Novel quark-field creation operator construction for hadronic physics in lattice QCD*, *Phys. Rev. D* **80** (2009) 054506 [0905.2160].

[2] C. Morningstar, J. Bulava, J. Foley, K.J. Juge, D. Lenkner, M. Peardon et al., *Improved stochastic estimation of quark propagation with Laplacian Heaviside smearing in lattice QCD*, *Phys. Rev. D* **83** (2011) 114505 [1104.3870].

[3] W. Detmold, D.J. Murphy, A.V. Pochinsky, M.J. Savage, P.E. Shanahan and M.L. Wagman, *Sparsening algorithm for multihadron lattice QCD correlation functions*, *Phys. Rev. D* **104** (2021) 034502 [1908.07050].

[3] Depending on memory requirements, this parallelization can straightforwardly be supplemented by a standard domain decomposition over MPI ranks in order to enable scaling beyond one socket or node at the expense of an N_{dil}^3-sized MPI reduction at the end of kernel execution

[4] L. Liu, *Hadron Spectroscopy and Interactions from Lattice QCD*, *PoS* **LATTICE2022** (2022) 448.

[5] B. Hörz and A. Hanlon, *Two- and three-pion finite-volume spectra at maximal isospin from lattice QCD*, *Phys. Rev. Lett.* **123** (2019) 142002 [1905.04277].

[6] T.D. Blanton, F. Romero-López and S.R. Sharpe, *$I = 3$ Three-Pion Scattering Amplitude from Lattice QCD*, *Phys. Rev. Lett.* **124** (2020) 032001 [1909.02973].

[7] M. Mai, M. Döring, C. Culver and A. Alexandru, *Three-body unitarity versus finite-volume $\pi^+\pi^+\pi^+$ spectrum from lattice QCD*, *Phys. Rev. D* **101** (2020) 054510 [1909.05749].

[8] C. Culver, M. Mai, R. Brett, A. Alexandru and M. Döring, *Three pion spectrum in the $I = 3$ channel from lattice QCD*, *Phys. Rev. D* **101** (2020) 114507 [1911.09047].

[9] M. Fischer, B. Kostrzewa, L. Liu, F. Romero-López, M. Ueding and C. Urbach, *Scattering of two and three physical pions at maximal isospin from lattice QCD*, *Eur. Phys. J. C* **81** (2021) 436 [2008.03035].

[10] Hadron Spectrum collaboration, *Energy-Dependent $\pi^+\pi^+\pi^+$ Scattering Amplitude from QCD*, *Phys. Rev. Lett.* **126** (2021) 012001 [2009.04931].

[11] A. Alexandru, R. Brett, C. Culver, M. Döring, D. Guo, F.X. Lee et al., *Finite-volume energy spectrum of the $K^-K^-K^-$ system*, *Phys. Rev. D* **102** (2020) 114523 [2009.12358].

[12] T.D. Blanton, A.D. Hanlon, B. Hörz, C. Morningstar, F. Romero-López and S.R. Sharpe, *Interactions of two and three mesons including higher partial waves from lattice QCD*, *JHEP* **10** (2021) 023 [2106.05590].

[13] F. Romero-López, *Topical plenary on multi-particle interactions from lattice QCD*, *PoS* **LATTICE2022** (2022) 440.

[14] C. Drischler, W. Haxton, K. McElvain, E. Mereghetti, A. Nicholson, P. Vranas et al., *Towards grounding nuclear physics in QCD*, *Prog. Part. Nucl. Phys.* **121** (2021) 103888 [1910.07961].

[15] A. Francis, J.R. Green, P.M. Junnarkar, C. Miao, T.D. Rae and H. Wittig, *Lattice QCD study of the H dibaryon using hexaquark and two-baryon interpolators*, *Phys. Rev. D* **99** (2019) 074505 [1805.03966].

[16] J.R. Green, A.D. Hanlon, P.M. Junnarkar and H. Wittig, *Weakly bound H dibaryon from SU(3)-flavor-symmetric QCD*, 2103.01054.

[17] B. Hörz et al., *Two-nucleon S-wave interactions at the $SU(3)$ flavor-symmetric point with $m_{ud} \simeq m_s^{\rm phys}$: A first lattice QCD calculation with the stochastic Laplacian Heaviside method*, *Phys. Rev. C* **103** (2021) 014003 [2009.11825].

[18] S. Amarasinghe, R. Baghdadi, Z. Davoudi, W. Detmold, M. Illa, A. Parreno et al., *A variational study of two-nucleon systems with lattice QCD*, 2108.10835.

[19] C.W. Andersen, J. Bulava, B. Hörz and C. Morningstar, *Elastic* $I = 3/2p$*-wave nucleon-pion scattering amplitude and the* Δ*(1232) resonance from* N_f*=2+1 lattice QCD*, Phys. Rev. D **97** (2018) 014506 [1710.01557].

[20] G. Silvi et al., *P-wave nucleon-pion scattering amplitude in the* Δ*(1232) channel from lattice QCD*, Phys. Rev. D **103** (2021) 094508 [2101.00689].

[21] J. Bulava, A.D. Hanlon, B. Hörz, C. Morningstar, A. Nicholson, F. Romero-López et al., *Elastic nucleon-pion scattering at* $m_\pi = 200$ MeV *from lattice QCD*, 2208.03867.

[22] T. Doi and M.G. Endres, *Unified contraction algorithm for multi-baryon correlators on the lattice*, Comput. Phys. Commun. **184** (2013) 117 [1205.0585].

[23] W. Detmold and K. Orginos, *Nuclear correlation functions in lattice QCD*, Phys. Rev. D **87** (2013) 114512 [1207.1452].

[24] J. Günther, B.C. Toth and L. Varnhorst, *Recursive approach to determine correlation functions in multibaryon systems*, Phys. Rev. D **87** (2013) 094513 [1301.4895].

[25] C. Morningstar, J. Bulava, B. Fahy, J. Foley, Y.C. Jhang, K.J. Juge et al., *Extended hadron and two-hadron operators of definite momentum for spectrum calculations in lattice QCD*, Phys. Rev. D **88** (2013) 014511 [1303.6816].

[26] D. Mohler, S. Schaefer and J. Simeth, *CLS 2+1 flavor simulations at physical light- and strange-quark masses*, EPJ Web Conf. **175** (2018) 02010 [1712.04884].

[27] M. Bruno et al., *Simulation of QCD with* $N_f = 2 + 1$ *flavors of non-perturbatively improved Wilson fermions*, JHEP **02** (2015) 043 [1411.3982].

[28] M. Bruno, T. Korzec and S. Schaefer, *Setting the scale for the CLS* $2 + 1$ *flavor ensembles*, Phys. Rev. D **95** (2017) 074504 [1608.08900].

[29] S. Paul, A.D. Hanlon, B. Hörz, D. Mohler, C. Morningstar and H. Wittig, *I=1* π*-*π *scattering at the physical point*, PoS **LATTICE2021** (2022) 551 [2112.07385].