EPJ Web of Conferences **175**, 14011 (2018) https://doi.org/10.1051/epjconf/201817514011
Lattice 2017

An update on the BQCD Hybrid Monte Carlo program

Taylor Ryan Haar[1], *Yoshifumi* Nakamura[2], and *Hinnerk* Stüben[3,⋆]

[1] *CSSM, Department of Physics, The University of Adelaide, Adelaide, SA, Australia 5005*
[2] *RIKEN Advanced Institute for Computational Science, Kobe, Hyogo 650-0047, Japan*
[3] *Universität Hamburg, Regionales Rechenzentrum, 20146 Hamburg, Germany*

Abstract. We present an update of BQCD, our Hybrid Monte Carlo program for simulating lattice QCD. BQCD is one of the main production codes of the QCDSF collaboration and is used by CSSM and in some Japanese finite temperature and finite density projects. Since the first publication of the code at Lattice 2010 the program has been extended in various ways. New features of the code include: dynamical QED, action modification in order to compute matrix elements by using Feynman-Hellman theory, more trace measurements (like $\mathrm{T}r(D^{-n})$ for κ, c_{SW} and chemical potential reweighting), a more flexible integration scheme, polynomial filtering, term-splitting for RHMC, and a portable implementation of performance critical parts employing SIMD.

1 Introduction

BQCD is a Hybrid Monte Carlo program for simulating lattice QCD with dynamical Wilson fermions. It was first published at Lattice 2010 [1] and has been used by several groups: the QCDSF-UKQCD collaboration [2–7], CSSM [6–9], Japanese finite density [10, 11] and finite temperature [12] projects, and the RQCD collaboration [13].

Here we report on extensions and optimizations that were made meanwhile and give an update on compute performance. The code and a manual can be downloaded from [14]. New features of the program are:

- *Actions:* hopping term with chemical potential, clover $O(a)$ improved Wilson action plus a CPT breaking term, QCD+QED, QCD+Axion. See section 2.
- *Algorithms:* polynomial filtering, a generalized multiscale integration scheme, truncated RHMC, Zolotarev approximation. See section 4.
- *Compute performance optimizations*: explicit vectorization with SIMD intrinsics, improvement of MPI communication. See section 5.

2 Actions

2.1 Gauge actions

Implemented are the Wilson gauge action and an improved gauge action, see [1].

⋆Speaker, e-mail: hinnerk.stueben@uni-hamburg.de

2.2 Fermion actions

At the time of [1] the program could simulate the standard Wilson fermion action $S_F^{\rm Wilson}$, $O(a)$ clover improved Wilson fermions, and the SLiNC fermion action [4]. The new version can also simulate:

- the hopping term with chemical potential μ [10, 11]

$$S_F = \sum_x \Big\{ \bar\psi(x)\psi(x) - \kappa \sum_i^3 \Big[\bar\psi(x) U_i^\dagger(x-\hat\imath)(1+\gamma_i)\psi(x-\hat\imath) + \bar\psi(x) U_i(x)(1-\gamma_i)\psi(x+\hat\imath) \Big] - \kappa \Big[\bar\psi(x) U_4^\dagger(x-\hat4)(1+\gamma_4) e^{-\mu}\psi(x-\hat4) + \bar\psi(x) U_4(x)(1-\gamma_4) e^{\mu}\psi(x+\hat4) \Big] \Big\} \quad (1)$$

- the clover $O(a)$ improved Wilson action plus a CPT breaking term with coefficient λ and a 4×4 matrix H [6, 7]

$$S_F = S_F^{\rm Wilson} - \frac{i}{2}\kappa\, c_{\rm SW} \sum_x \left[\bar\psi(x)\sigma_{\mu\nu}F_{\mu\nu}(x)\psi(x) + \kappa\,\lambda \bar\psi(x) H \psi(x) \right] \quad (2)$$

2.3 QCD+QED

The program can simulate QCD+QED [5] using the action

$$S = S_G + S_A + \sum_q S_F^q \,. \quad (3)$$

S_G is an SU(3) gauge action, S_A is the non-compact U(1) gauge action

$$S_A = \frac{\beta_{\rm QED}}{2} \sum_{x,\mu<\nu} \Big[A_\mu(x) + A_\nu(x+\hat\mu) - A_\mu(x+\hat\nu) - A_\nu(x) \Big]^2 \,, \quad (4)$$

and the fermion action for flavour q is

$$S_F^q = \sum_x \Big\{ \kappa_q \sum_\mu \Big[\overline{q}(x)(\gamma_\mu - 1) e^{-iQ_q A_\mu(x)} \tilde U_\mu(x) q(x+\hat\mu) - \overline{q}(x)(\gamma_\mu+1) e^{iQ_q A_\mu(x-\hat\mu)} \tilde U_\mu^\dagger(x-\hat\mu) q(x-\hat\mu) \Big] + \overline{q}(x)q(x) - \frac{1}{2}\kappa_q c_{\rm SW} \sum_{\mu,\nu} \overline{q}(x)\sigma_{\mu\nu}F_{\mu\nu}(x)q(x) \Big\} \,, \quad (5)$$

where $Q_u = +2/3$, $Q_d = Q_s = -1/3$ and $\tilde U_\mu$ is a singly iterated stout link.

2.4 QCD+Axion

The program can simulate QCD+Axion [15] using the action

$$S = S_G + S_a + \sum_q S_F^q \,. \quad (6)$$

S_a is scalar action for the axion field ϕ_a

$$S_a = \kappa_a \sum_x \sum_\mu \Big(\phi_a(x) - \phi_a(x+\mu) \Big) \phi_a(x) \,, \quad (7)$$

and the fermion action for flavour q in the case of Wilson fermions is

$$S_F^q = \sum_x \bar{q}(x)\Big[1 + (\kappa_q \lambda_q + f_{\rm inv}\phi_a)\gamma_5\Big]q(x) \\ - \kappa_q \sum_{x,\mu}\Big[\bar{q}(x)(1-\gamma_\mu)U_\mu(x)q(x+a\hat{\mu}) + \bar{q}(x-a\hat{\mu})(1+\gamma_\mu)U_\mu^\dagger(x-a\hat{\mu})q(x)\Big], \tag{8}$$

where

$$\kappa_q = \frac{1}{2am_q + 8}, \quad \lambda_q = i2am_q\frac{\theta}{N_f}, \quad f_{\rm inv} = i2\kappa_q m_q \frac{\sqrt{\kappa_a}}{f_a N_f}. \tag{9}$$

3 Measurements

The following quantities can be measured with BQCD: plaquettes (quadratic and rectangular), topological charge (cooling method), Polyakov loop, Wilson flow, traces of the fermion matrix ($\mathrm{Tr}\,(M^{-1})$, $\mathrm{Tr}\,(\gamma_5 M^{-1})$, $\mathrm{Tr}\,(M^\dagger M)^{-1}$), quark determinant with chemical potential, smallest and largest eigenvalue of the Dirac matrix, meson and baryon propagators.

4 Algorithms

In addition to nested integrators for multiple time scales, a generalized integration scheme [8] has been implemented. This is where separate integration schemes for each action term are superimposed onto a single time step evolution. This allows the integration step-sizes for each action term to be completely independent of the others.

Rational Hybrid Monte Carlo (RHMC) is implemented with rational approximations from the Remez algorithm. In the case of approximating $(W^\dagger W)^{-1/2}$, an alternative rational function is available, namely the Zolotarev optimal rational approximation (see e.g. [16] for an explanation).

Alongside Hasenbusch filtering, there are two new filtering methods, one of which applies exclusively to RHMC:

- *Polynomial filtering* applies to both RHMC and standard HMC fermions, and is the application of a polynomial filter $P(W^\dagger W)$ to split the fermion action into several terms:

$$S_{PF} = \overline{\psi}_1 P(W^\dagger W)\psi_1 + \overline{\psi}_2 P(W^\dagger W)^{-1} W^\dagger W \psi_2. \tag{10}$$

- *Term-splitting for RHMC* splits the sum in the rational approximation $R(W^\dagger W)$ for RHMC into several terms, giving action

$$S_{tRHMC} = \overline{\psi}_1 R_{1,t}(W^\dagger W)\psi_1 + \overline{\psi}_2 R_{t+1,N}(W^\dagger W)\psi_2 \tag{11}$$

where

$$R_{i,j}(K) = c_n^{\delta_{i1}} \sum_{k=i}^{j} \frac{W^\dagger W + a_k}{W^\dagger W + b_k}, \tag{12}$$

a_k, b_k are ordered decreasing.

BQCD has a wide range of iterative solvers: *cg*, BiCGstab, GMRES, GCRODR, multishift *cg*, block multishift *cg*.

5 Optimization of compute performance

5.1 SIMD vectorization and MPI

In addition to parallelization with MPI and OpenMP a third level of parallel implementation was introduced for solvers: SIMD vectorization with SIMD intrinsic functions. The SIMD implementation is generic, i.e. it works for any size of SIMD vectors. In order to achieve this, the data layout of arrays had to be changed. All arrays (for gauge, spin-colour and clover fields) now have SIMD vectors as the smallest structure. In Fortran notation the gauge field is defined in the following way

```
old:  complex(8) :: u(3, 3, volume/2)
new:  real(8) :: u(SIMDsize, re:im, 3, 3, volume/2 / SIMDsize)
```

and the new layout of the spin-color field is

```
old:  complex(8) :: a(4, 3, volume/2)
new:  real(8) :: a(SIMDsize, re:im, 2, 3, volume/2 / SIMDsize, 2)
```

where the 4 spin components of the spin-colour field are split into 2 + 2 components which optimizes MPI communication in t-direction. The clover arrays, for which a packed format is used, were changed accordingly.

At the single core level the SIMD code is about 2 times faster than the corresponding Fortran code. With this speed-up computations are increasingly dominated by communication and improvement of MPI communication becomes important. Hence, the following MPI optimizations where made:

- The overhead introduced by the reduction to two-component spinors was minimized. Previously the projection was done for the whole local volume, now it is only done for boundary sites, and there is no projection in the t-direction needed any more.
- All MPI 'buffers' are consecutive in memory and aligned to SIMD vector boundaries.
- Communication can overlap with computation. This is implemented with MPI plus OpenMP, where the *master thread* communicates while the other threads compute.

In tables 1 and 2 performance figures are listed for machines and lattices that are currently used in production. The optimized code runs between 1.3 and 1.7 times faster.

Table 1. Double precision performance of the *cg*-solver of BQCD on a Cray XC40 (24 cores per node).

	$48^3 \times 96$ lattice				$64^3 \times 96$ lattice			
	Fortran		SIMD		Fortran		SIMD	
#cores	per core Mflop/s	overall Tflop/s	per core Mflop/s	overall Tflop/s	per core Mflop/s	overall Tflop/s	per core Mflop/s	overall Tflop/s
1536	930	1.4	1557	2.4	785	1.2	1028	1.6
3072	949	2.9	1516	4.7	795	2.4	1231	3.8
6144	1222	7.5	1558	9.6	876	5.4	1419	8.7
9216	1253	11.5	1775	16.4	–	–	–	–
12288	1274	15.7	1678	20.6	927	11.4	1572	19.3

5.2 QUDA

BQCD can run on GPUs by employing the QUDA library [17]. QUDA has a BQCD interface to its *cg* and multishift *cg* solvers.

Table 2. Double precision performance of the *cg*-solver of BQCD on an IBM BlueGene/Q (8192 cores, a *midplane*, is the smallest partition that has a fully wired torus network and with our SIMD implementation the largest possible partition for the $48^3 \times 96$ lattice, where the lattice volume per core is 48×3^3).

	$48^3 \times 96$ lattice				$64^3 \times 96$ lattice			
	Fortran		SIMD		Fortran		SIMD	
#cores	per core Mflop/s	overall Tflop/s	per core Mflop/s	overall Tflop/s	per core Mflop/s	overall Tflop/s	per core Mflop/s	overall Tflop/s
8192	539	4.4	912	7.5	525	4.3	752	6.2
16384	–	–	–	–	596	9.8	783	12.8
32768	–	–	–	–	503	16.5	771	25.3

Acknowledgements

We would like to thank Gerrit Schierholz, Roger Horsley, Waseem Kamleh, Paul Rakow and James Zanotti for support, stimulating discussions and bug reports. The computations were performed on a Cray XC40 of the North-German Supercomputing Alliance (HLRN) and the IBM BlueGene/Q at Jülich Supercomputer Centre (JSC).

References

[1] Y. Nakamura, H. Stüben, PoS **LATTICE2010**, 040 (2010), 1011.0199
[2] H. Stüben (UKQCD, QCDSF), Nucl. Phys. Proc. Suppl. **94**, 273 (2001), hep-lat/0011045
[3] M. Göckeler et al. (QCDSF), PoS **LAT2007**, 041 (2007), 0712.3525
[4] N. Cundy et al., Phys. Rev. **D79**, 094507 (2009), 0901.3302
[5] R. Horsley et al., JHEP **04**, 093 (2016), 1509.00799
[6] A.J. Chambers et al. (QCDSF/UKQCD, CSSM), Phys. Rev. **D90**, 014510 (2014), 1405.3019
[7] A.J. Chambers et al., Phys. Rev. **D92**, 114517 (2015), 1508.06856
[8] T. Haar, W. Kamleh, J. Zanotti, Y. Nakamura, Comput. Phys. Commun. **215**, 113 (2017), 1609.02652
[9] S. Hollitt, P. Jackson, R. Young, J. Zanotti, PoS **INPC2016**, 272 (2017)
[10] X.Y. Jin, Y. Kuramashi, Y. Nakamura, S. Takeda, A. Ukawa, Phys. Rev. **D88**, 094508 (2013), 1307.7205
[11] X.Y. Jin, Y. Kuramashi, Y. Nakamura, S. Takeda, A. Ukawa, Phys. Rev. **D92**, 114511 (2015), 1504.00113
[12] X.Y. Jin, Y. Kuramashi, Y. Nakamura, S. Takeda, A. Ukawa, Phys. Rev. **D96**, 034523 (2017), 1706.01178
[13] G.S. Bali, S. Collins, A. Cox, A. Schäfer, Phys. Rev. **D96**, 074501 (2017), 1706.01247
[14] https://www.rrz.uni-hamburg.de/bqcd
[15] G. Schierholz, Y. Nakamura, *Dynamical QCD+Axion simulation: First results*, in *Proceedings, 35th International Symposium on Lattice Field Theory (Lattice2017): Granada, Spain*, to appear in EPJ Web Conf.
[16] T.W. Chiu, T.H. Hsieh, C.H. Huang, T.R. Huang, Phys. Rev. **D66**, 114502 (2002), hep-lat/0206007
[17] M.A. Clark, R. Babich, K. Barros, R.C. Brower, C. Rebbi, Comput. Phys. Commun. **181**, 1517 (2010), 0911.3191