Performance of PHMC and HMC algorithms in $n_f = 4$ LQCD with twisted Wilson quarks

ETM Collaboration

Albert Deuzeman, Siebren Reker
Centre for Theoretical Physics, University of Groningen, Nijenborgh 4, 9747 AG Groningen, the Netherlands
E-mail: a.deuzeman@rug.nl, s.f.reker@rug.nl

Vicent Giménez
Dep. de Física Teórica and IFIC (Univ. de Valencia-CSIC), Dr. Moliner 50, E-46100 Burjassot, Spain
E-mail: vicente.gimenez@uv.es

Gregorio Herdoiza, Karl Jansen[*]
DESY, Zeuthen, Platanenallee 6, D-15738 Zeuthen, Germany
E-mail: gregorio.herdoiza@desy.de, karl.jansen@desy.de

David Palao[†]
Dep. de Física Teórica and IFIC (Univ. de Valencia-CSIC), Dr. Moliner 50, E-46100 Burjassot, Spain and INFN–"Tor Vergata" c/o Dipartimento di Fisica, Universita di Roma "Tor Vergata", Via della Ricerca Scientifica 1, I-00133 Roma, Italy
E-mail: david.palao@roma2.infn.it

Carsten Urbach
Institut für Elementarteilchenphysik, Fachbereich Physik, Humbolt Universität zu Berlin, D-12489, Berlin, Germany
E-mail: carsten.urbach@physik.hu-berlin.de

We discuss the performance and stability of both (P)HMC and HMC algorithms in the context of the production of gauge configurations with four dynamical flavours of twisted mass Wilson fermions. The (P)HMC algorithm is crucial in the $2+1+1$ setup (a degenerate light doublet and a mass split heavy doublet), whereas the HMC is the simplest choice for simulations dedicated to non-perturbative renormalization in a mass-independent scheme, where the two doublets are degenerate. We also present LEMON: a C library designed for the writing and reading of LIME format files using MPI's parallel I/O facilities.

The XXVII International Symposium on Lattice Field Theory - LAT2009
July 26-31 2009
Peking University, Beijing, China

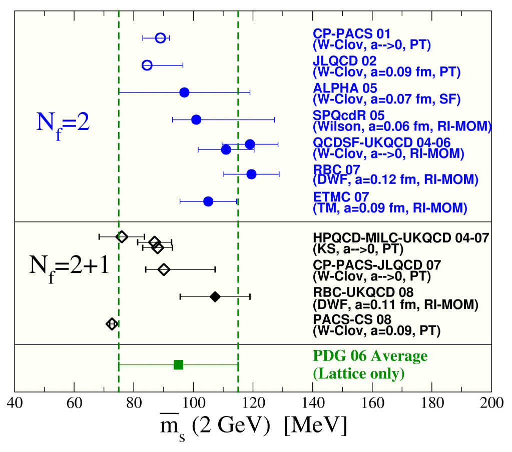
[*]Preprint number: DESY 09-197
[†]Speaker.

1. Introduction and motivation

Recently simulations of MtmLQCD with dynamical u, d (degenerate) as well as s and c (non-degenerate) quarks (so called $n_f = 2+1+1$ setup) have been performed, based on the lattice fermionic formulation of [1] and a suitable (Iwasaki) pure gauge action. While preliminary results appear to have good quality (see [2] and [3]), much in line with those obtained with only two dynamical flavours, and further simulations on fine lattices are in progress, a special effort is required to obtain non-perturbative renormalization constants; this necessity is clearly illustrated in figure 1. Indeed, since the scheme where it is convenient to compute the running of the renormalization constants is a "mass-independent" one[1] (in the sense of [5]), namely the RI'mom scheme (see e.g. refs. [6], [7] and [8]), the unquenched simulations with heavy c, "lightish" s and light u and d sea quarks, from which the non-renormalized and/or bare quantities are computed, can not be immediately used for the evaluation of the renormalization constants of the operators with non-vanishing anomalous dimension. Rather one has to perform, for each of the considered lattice resolutions, further dedicated simulations, now with $n_f = 4$ degenerate sea quarks at a few different, not too large mass values. On the resulting gauge configurations one can then compute the Landau gauge correlators that yield, after a suitable analysis (including a safe extrapolation to the zero-mass point), the desired renormalization constants in RI'mom scheme (see e.g. refs. [8] or [9]).

However, before embarking on the computation of renormalization constant from configurations with four dynamical flavours, it is necessary a careful analysis from the algorithmic point of view.

Figure 1: Effect on the strange quark mass of using a non-perturbative method to compute the renormalization constants. Empty/filled symbols correspond to perturbative/non-perturbative computations of the renormalization constants respectively; taken from [4]



[1]this fact guarantees the equivalence between tmQCD and QCD at finite lattice spacing for Wilson fermions.

Table 1: Results of the performance comparison test between HMC and (P)HMC with $n_f = 4$ dynamical flavours. The times per trajectory refer to the same machine using the same number of processes.

Algorithm	$<P>$	τ_{int}	P_{acc}	$<t/s>$	η/s
HMC	0.58415(12)	180(30)	86%	65	$\sim$ 13600
(P)HMC	0.58415(20)	160(20)	88%	76	$\sim$ 13800

2. Performance of (P)HMC vs HMC with $n_f = 4$ at maximal twist

The ETMC simulations with four non-degenerate dynamical flavours have been carried out using the so-called (P)HMC algorithm: plain HMC on the first degenerate doublet and PHMC on the second non-degenerate one (see [10] for a detailed description). Even restricting ourselves to PHMC and HMC, when simulating $n_f = 4$ degenerate flavours there are more options: pure HMC, pure PHMC or (P)HMC. In this section we compare HMC with (P)HMC in terms of performance.

2.1 Test setup

We try to mimic, as much as possible, the setup used in the production of $n_f = 2+1+1$ ETMC gauge configurations (see [2] and [3]). We simulated 4 degenerate twisted mass Wilson fermions near maximal twist with $a\mu = 0.0085$ and $\kappa = 0.160943 \sim \kappa_c$. The gauge action chosen is Iwasaki with $\beta = 1.95$ and the lattice size is $L = T/2 = 16a$. We will consider MC histories of 8000 trajectories of length 1 in both cases.

For the (P)HMC simulation we consider the following additional parameters: the degree of the polynomial is 1789; the precision required is 9×10^{-6} and the approximation interval is $[\lambda_{\min}/\lambda_{\max}, 1] = \left[8.7 \times 10^{-6}, 1\right]$.

2.2 Results

Table 1 and figures 2(a) and 2(b) summarize the performance comparison between (P)HMC and HMC with four dynamical degenerate flavours of twisted mass Wilson Fermions.

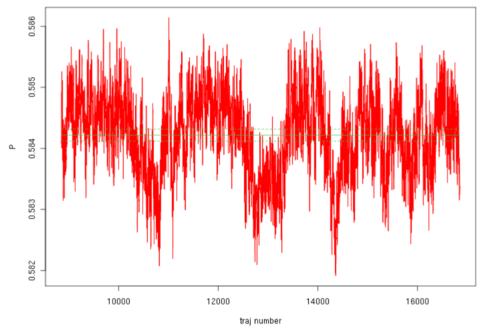
One possibility to compare the performance is to use the combination

$$\eta = \frac{\tau_{\text{int}}(P)\,t}{P_{\text{acc}}} \tag{2.1}$$

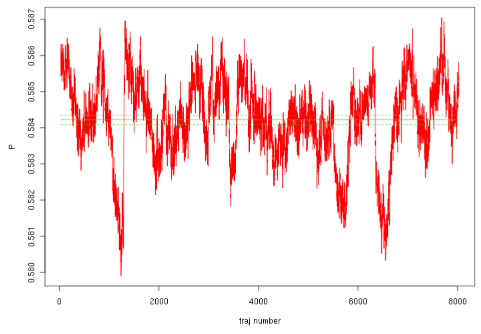
where t is the time per trajectory; this quantity is an estimate of the necessary time to obtain two independent gauge configurations, but can be useful only if we run our simulations on the same machine using the same number of CPUs. From the results we notice that for this particular set of parameters the advantage of using HMC over (P)HMC is very small. Still we consider that using HMC is the best choice: apart from performing slightly better, the HMC is simpler to tune and more robust.

3. Stability

There are clear numerical evidences pointing out that the Monte Carlo simulations of LQCD become less stable when the number of active sea flavours increases. For example, we have ob-



(a)



(b)

Figure 2: MC histories of HMC (a) and (P)HMC (b)

served that when going from $n_f = 2$ (two light degenerate flavours) to $n_f = 2+1+1$ (two light, the strange and the charm), the simulations become more unstable around the critical point. A similar effect has been noticed when going from $n_f = 2+1+1$ to $n_f = 4$ (four light degenerate flavours).

This fact must be related to the presence of a Singleton-Sharpe phase transition ([11], [12]). In order to have a better quantitative understanding of this phenomenon we have performed a numerical study in the context of twisted mass LQCD near maximal twist with four degenerate light quarks. The goal is to measure the slope of m_{PCAC} when the hopping parameter changes and the twisted mass remains constant.

3.1 Test setup

The lattice volume and the gauge action are the same as in section 2. We have performed three sets of simulations: the first with 4 light degenerate flavours (set S1); the second with two light degenerate flavours (S2); and the third corresponds to simulations with 2 heavier degenerate flavours (S3). We measured the slope of $< am_{\mathrm{PCAC}} >$ with $1/2\kappa$ around the critical region and far from it. We know that, when $m_0 \equiv 1/2\kappa - 1/2\kappa_c \gg 0$, this slope is $\mathscr{O}(1)$ and independent of m_0. With this idea in mind we are able to identify the region where the phase transition occurs, and the region where we are far from the critical point.

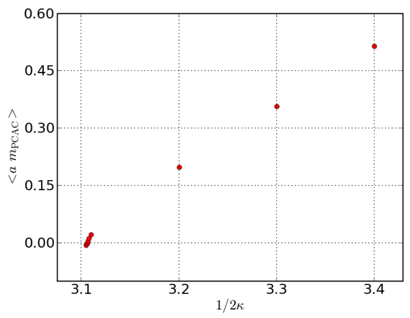
3.2 Results

Table 2 and figures 3(a) to 3(d) summarize the results of this test. In table 2 we used the notation s_x to denote our estimation of the slope, on region x, of $< am_{\mathrm{PCAC}} >$ when m_0 changes. As can be seen the slope around the critical point clearly increases when the number of active flavours increase; we see from the last column of the table that, in the present test, the ratio of slopes increase roughly linearly with the number of active flavours.

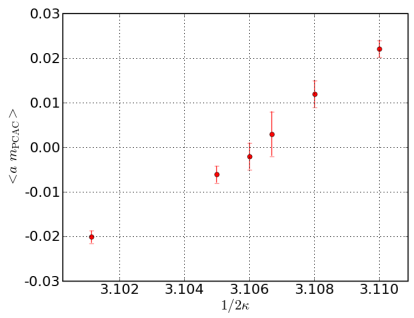
From this test we learn an important lesson. As can be seen from table 2, and figures 3(b) and 2(a) (which corresponds to the simulation of set S1 with the smallest measured value of am_{PCAC} and the highest statistics), in this setup it is difficult to measure am_{PCAC} with high precison when we are close to the critical value, hence the tuning at maximal twist is a difficult task. Of course, the situation will only worsen if we go to bigger volumes.

Table 2: Results of the stability test. s_x is our estimate of $\frac{\partial am_{\rm PCAC}}{\partial m_0}$ on region x

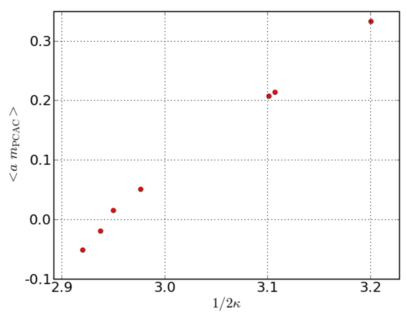
Set	$n_f^{\rm sea}$	$a\mu^{\rm sea}$	$s_{m_0 \sim 0}$	$s_{m_0 \gg 0}$	$s_{m_0 \sim 0}/s_{m_0 \gg 0}$
S1	4	0.0085	6.0(1.2)	1.584(9)	3.8(7)
S2	2	0.0085	2.21(11)	1.275(9)	1.73(7)
S3	2	0.24	1.402(7)	1.385(7)	1.01227(6)



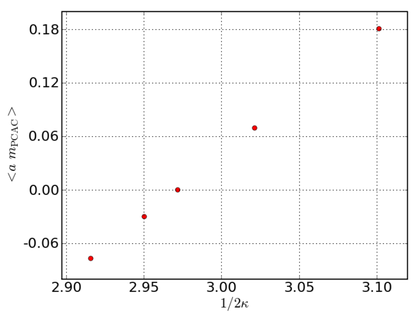
(a) S1 set



(b) S1: detail



(c) S2 set



(d) S3 set

Figure 3: $< am_{\rm PCAC} >$ versus the Wilson mass. (a) and (b) refer both to the S1 set of simulations

4. Conclusions

Our main motivation for the present work has been to understand the difficulties asociated with the production of gauge configurations with four degenerate light flavours of dynamical twisted mass Wilson quarks. The information obtained from the two tests described will be crucial to decide the strategy to follow in the computation of the renormalization constants in a massless scheme as RI'mom.

A. Parallel I/O: LEMON

Parallel I/O was one of the crucial missing components of the codes currently in use by the lattice community. By Amdahl's law, the maximum theoretical speed up factor of any code is given by the reciprocal of the fraction of the code that is inherently serial. With the increasing parallelism

Table 3: Comparison of file sizes and serial writing times of the tmLQCD code on the Babel BG/P at IDRIS

$L = T/2$	24	32	48	64	80
Conf. size (Gb)	0.382	1.21	6.12	19.3	47.2
serial write time @ Babel (s)	8.8(2)	27.8(2)	141.6(7)	450(2)	1105(7)

of current lattice simulations, the serial parts of the code represent a significant fraction of the total execution time of the code. In other words, those processes taking place in addition to the parallel calculations start to take long enough to be a noticeable part of the time we find acceptable for performing a calculation.

The most important serial part of our codes by far is I/O. To keep parallelization efficiency up, we need to find a way to reduce the time it takes to write, and to a lesser degree read, files to and from disk. This need is clearly illustrated in table 3.

In order to attack this problem, two complementary approaches are available. First, we need to have access to fast I/O systems and attempt to utilize the bandwidth they provide. We have limited control over the hardware provided, but modern supercomputers tend to be equipped with parallelized I/O systems like GPFS that can in principle provide excellent performance if used appropriately. The latter dovetails with the second approach, where we would try to implement parallel processing of I/O requests, instead of channeling all data to a single node that carries responsibility for writing to file. Not only does this mean that less computational power is lost while other nodes attempt to send their share of the data to the privileged I/O node, but for modern systems different nodes tend to have access to diferent hardware pipelines for writing. From this point of view, parallel I/O actually increases the hardware performance in a way that scales with the size of the allocated number of nodes, breaking the barrier of Amdahl's law. For this reason, the LEMON parallel I/O library was introduced.

LEMON is a parallel alternative to LIME (or C-LIME). As its web site [13] explains: *The LIME software package consists of a C-language API for creating, reading, writing, and manipulating LIME files and a small set of utilities for examining, packing and unpacking LIME files*.

During the last few years the LIME format has become widely used by several Lattice groups. However the LIME software is only able to do serial I/O, and as explained before, this starts to be a bottleneck. Fortunately, LIME is at its core nothing but a well defined format for writing files, one that can be independently implemented. LEMON does exactly that: it implements the LIME file format on top of the MPI-2 I/O routines. Its basic API has been kept similar to that of LIME, to lower the learning curve and making porting of code easier. A LEMON reader or writer object requires a MPI file handle and an associated Cartesian communicator and will then take care of calling the appropriate MPI routines. Like LIME does, it provides a system to read and write appropriately documented and padded files comprised of a series LIME records and messages.

Much of LIME is dedicated to serial writing of meta information and this is supported by LEMON. Unless specific functions are called, the provided information is written once as if a serial I/O call had been performed. Since LEMON needs to be aware of the parallel file system it interacts with, all calls should be made collectively. The library will take care of assigning the task to a single node. There is no real performance advantage to LEMON over LIME here, but meta data written in this way is usually no larger than a kilobyte or so.

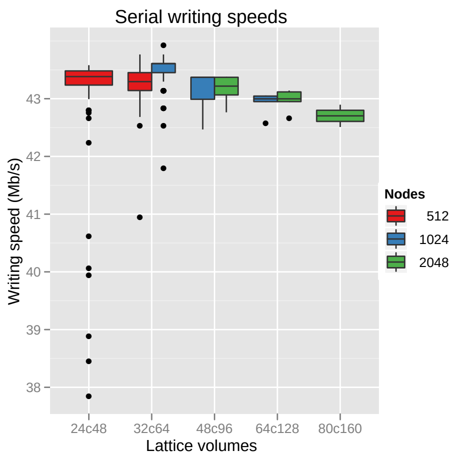
LEMON starts to shine when a lattice worth of that data needs to be written, when each node has direct access to part of data arranged in a hypercube. Using the Cartesian communicator and MPI-2 functionality for serialization, specialized functions in the library will take care of writing the data at each coordinate in the lattice directly to the appropriate position in the file (according to the SCIDAC binary data scheme) in a parallel manner. Compared to the procedure with LIME, where serialization needs to be done explicitly and tends to be coordinated by a single writing node (exactly the reason why parallel writing is complicated to implement on top of the C standard library), the performance gains are truly tremendous for large lattices. As a fortunate side effect, the code becomes easier to read and maintain since serialization can be off-loaded to lower level routines.

There may be situations where one wants to go further than this. When the computational component is fast, I/O may be relatively costly even in parallel. In such cases additional performance may be had from using non-blocking I/O. These routines will start I/O in the background and return immediately, without waiting for completion of the processes. Depending on the hardware, this will have computation mask I/O to a certain extent, reducing the total execution time by up to a factor of two. LEMON provides routines to perform parallel I/O in this fashion as well, handling the completion of requests within the reader or writer structure. The cost for this performance gain is measured in memory. Generally, one will want to provide buffer space for a complete additional local lattice volume of the quantity involved.An appropriate choice for the internal data structure or some clever construction may eliminate this necessity, but one could incur a performance hit again. There will usually be space for such a buffer on massively parallel systems, because we don't tend to be constrained by memory there. In those few cases that we are, I/O should not really be a bottleneck to begin with. If non-blocking I/O is desired, the requirements are a little higher. We need to make sure the MPI functions have exclusive access to the buffers involved, so if we are to perform other operations on the data we may need to make additional copies. Care has to be taken throughout the code to leave these designated areas alone until the I/O operations are finished as well. Because of this added complexity, it is probably best to reserve the non-blocking functionality for those situations where the pay off is sufficiently large.

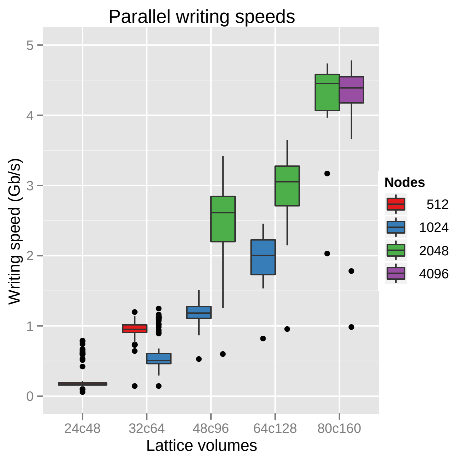
LEMON can be obtained using SVN from `svn://thep.housing.rug.nl/lemon`. It is also included in the tmLQCD code [14]. Since it is derived from LIME, it inherits its GPL license. Bugs should be reported at `a.deuzeman@rug.nl` and so should feature requests. The API is stable and no extensions are planned at this moment, but they will be introduced as need arises taking backwards compatibility into account as much as possible. Tests have been performed on scalar, shared memory and distributed memory systems. The code has been shown to work with several implementations of MPI and on both NFS and GPFS storage systems.

Acknowledgments

We thank to Roberto Frezzotti for useful comments. We also thank to BSC and IDRIS for giving us computer time to carry out this analysis. V.G. and D.P. thank to Ministerio de Ciencia e Innovación (Spain) for partial financial support under grant FPA2008-03373.



(a)



(b)

Figure 4: Serial (a) and parallel (b) writing speeds on Babel (at IDRIS) using the tmLQCD [14] I/O routines. The speeds listed here are not pure writing speed, but necessarily also include checksumming, communication and endianness conversion. Because of the setup used, it is impossible to separately measure these steps. On the y-axis, the writing speed is given, the boxes show the median and first quartile distribution of the measurements, the black dots indicate outliers. Lattice volumes are grouped on the x-axis, with different colours being used to indicate different partition sizes. The reason for the outliers is filesystem usage by other processes. The fact that more outliers show in the 24c48 job is that that job has the most statistics. Notice the different scales used in the serial case (a) and in th parallel case (b).

References

[1] R. Frezzotti and G. C. Rossi, Nucl. Phys. Proc. Suppl. **128** (2004) 193 [arXiv:hep-lat/0311008].

[2] R. Baron *et al.* [ETM Collaboration], PoS **LATTICE2008** (2008) 094 [arXiv:0810.3807 [hep-lat]].

[3] R. Baron *et al.* [ETM Collaboration], PoS **LATTICE2009** (2009) 104

[4] B. Blossier *et al.* [ETMC], JHEP **0804** (2008) 020 [arXiv:0709.4574 [hep-lat]].

[5] S. Weinberg, Phys. Rev. D **8** (1973) 3497.

[6] D. Becirevic, D. Meloni, A. Retico, V. Gimenez, L. Giusti, V. Lubicz and G. Martinelli, Nucl. Phys. B **618** (2001) 241 [arXiv:hep-lat/0002025].

[7] E. Franco and V. Lubicz, Nucl. Phys. B **531** (1998) 641 [arXiv:hep-ph/9803491].

[8] D. Becirevic, V. Gimenez, V. Lubicz, G. Martinelli, M. Papinutto and J. Reyes, JHEP **0408** (2004) 022 [arXiv:hep-lat/0401033].

[9] P. Dimopoulos, R. Frezzotti, G. Herdoiza, A. Vladikas, V. Lubicz, S. Simula and M. Papinutto, PoS **LAT2007** (2007) 241 [arXiv:0710.0975 [hep-lat]].

[10] T. Chiarappa, R. Frezzotti and C. Urbach, PoS **LAT2005** (2006) 103 [arXiv:hep-lat/0509154].

[11] S. R. Sharpe and R. L. . Singleton, Phys. Rev. D **58** (1998) 074501 [arXiv:hep-lat/9804028].

[12] S. R. Sharpe and J. M. S. Wu, Phys. Rev. D **70** (2004) 094029 [arXiv:hep-lat/0407025].

[13] `http://usqcd.jlab.org/usqcd-docs/LIME/`

[14] K. Jansen and C. Urbach, Comp. Phys. Comm. **180** (2009) 2717 [arXiv:0905.3331 [hep-lat]].