PROCEEDINGS OF SCIENCE

Automated tuning for HMC mass ratios

J. Torsiello,[a,b] **G. T. Fleming,**[c] **S. Foreman,**[d] **X.-Y. Jin**[a] **and J.C. Osborn**[a,*]

[a] *Computational Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL, USA*

[b] *Physics Department, Temple University, 1925 N. 12th St., Philadelphia, PA, USA*

[c] *Theoretical Physics Division, Fermilab, Batavia, IL 60510, USA*

[d] *Leadership Computing Facility, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL, USA*

E-mail: osborn@anl.gov

We extended previous work on tuning HMC parameters using gradient information to include Hasenbusch mass ratios. The inclusion of mass ratios adds many more parameters that need to be tuned, and also allows for lots of variations in the choice of integrator pattern. We investigate the effectiveness of automatically tuning a large number of HMC parameters and compare the optimally tuned versions over a range of integrator variants.

The 41st International Symposium on Lattice Field Theory (LATTICE2024)
28 July - 3 August 2024
Liverpool, UK

*Speaker

PoS(LATTICE2024)052

While Hybrid Monte Carlo (HMC) [1] has been very successful in sampling gauge fields from lattice gauge theory actions, efficient use of HMC can require tuning several simulation parameters [2, 3]. Previously, we investigated using machine learning (ML) methods, in particular gradient-based optimization, to automatically tune these parameters during sampling [4]. We found that the gradient-based tuning was very effective in tuning the higher-order molecular dynamics (MD) parameters used in HMC for a simple staggered fermion simulation. We now extend this work to include tuning Hasenbusch mass ratios [5] used as a preconditioner for the MD evolution.

1. Mass Preconditioning

The staggered fermion determinant (with 4 continuum flavors) can be written in terms of a pseudofermion (ϕ) action as

$$\left|D^\dagger D + m^2\right| = \int d\phi \; e^{\phi^\dagger [D^\dagger D + m^2]^{-1} \phi} \; , \tag{1}$$

where $D = D_{oe}$ is the staggered Dirac matrix from even to odd sites (which depends on the gauge field) and m is the fermion mass. With mass preconditioning, the determinant can be written as

$$\left|D^\dagger D + m^2\right| = \frac{\left|D^\dagger D + m^2\right|}{\left|D^\dagger D + h_1^2\right|} \frac{\left|D^\dagger D + h_1^2\right|}{\left|D^\dagger D + h_2^2\right|} \cdots \frac{\left|D^\dagger D + h_{n-1}^2\right|}{\left|D^\dagger D + h_n^2\right|} \left|D^\dagger D + h_n^2\right| \; , \tag{2}$$

with a set of n tunable parameters, h_i. This gives a pseudofermion action of

$$\int d\phi_0 e^{\phi_0^\dagger \frac{D^\dagger D + h_1^2}{D^\dagger D + m^2} \phi_0} \int d\phi_1 e^{\phi_1^\dagger \frac{D^\dagger D + h_2^2}{D^\dagger D + h_1^2} \phi_1} \cdots \int d\phi_{n-1} e^{\phi_{n-1}^\dagger \frac{D^\dagger D + h_n^2}{D^\dagger D + h_{n-1}^2} \phi_{n-1}} \int d\phi_n e^{\phi_n^\dagger \frac{1}{D^\dagger D + h_n^2} \phi_n} \; . \tag{3}$$

A careful choice of the h_i's can significantly reduce the fluctuations in the fermion force and allow for a larger step size in the MD integrator. Here we investigate using the gradient of the full HMC trajectory with respect to the h_i's to automatically tune these parameters towards their optimal settings.

2. Evaluating HMC efficiency

As was done previosly [4], we model a set of N HMC updates as a random walk due to the momentum refresh, and define the effective MD integration time as

$$T_{\text{eff}} \equiv \tau \sqrt{\langle P_a \rangle N} \tag{4}$$

where τ is the MD integration time for a single update and $\langle P_a \rangle$ is the average acceptance probability. This expression can then be solved for N to get the effective number of update steps, $N(T_{\text{eff}})$, needed to go an effective MD time of T_{eff}. The cost to evolve for an effective MD time T_{eff} is then $N(T_{\text{eff}})C$ where C is the cost per update.

The cost per update would ideally come from measurements of the time for every update operation (gauge update, force evaluations, etc.). For simplicity here, we assume the Dirac matrix solver is dominant and only include that. The time for each solve is proportional to the number

of iterations (N_{CG}), which for staggered fermions using conjugate gradient (CG), is approximately proportional to the inverse of the mass. We have verified this approximation to be correct to within 15% accuracy for all the solves used in our study. We then model the relative cost of all the solves in the HMC update (relative to a single solve at the dynamical mass) as

$$C_S = \sum_s \frac{N_{CG}(m_s)}{N_{CG}(m)} \approx \sum_s \frac{m}{m_s} \tag{5}$$

where m_s is the mass in each solve, s, that occurs in the full HMC update (which may repeat the dynamical or Hasenbusch masses some number of times).

Using C_S as the cost per update and setting $T_{\text{eff}} = 1$ we get for the cost function

$$\text{Cost} = N(T_{\text{eff}} = 1)C_S = \frac{C_S}{\tau^2 \langle P_a \rangle} , \tag{6}$$

Given this cost function, we now want to tune the parameters to minimize the cost. Due to the average acceptance, $\langle P_a \rangle$, appearing in the denominator, it is more convenient to work with the inverse of the cost, so we instead minimize the loss function

$$\text{Loss} = -\frac{1}{\text{Cost}} = -\frac{\langle P_a \rangle \tau^2}{C_S} . \tag{7}$$

The loss function is minimized with the Adam optimizer [6] using the gradient of the loss function, which includes taking the gradient of the acceptance probability, $P_a = \min[1, \exp(-\Delta H)]$, with ΔH the action difference between the end and start of the trajectory.

The gradient is taken with respect to the MD integration parameters and the Hasenbusch masses, h_i. Much of this calculation proceeds similar to before [4]. One notable new piece here is the need to take the gradient of the pseudofermion field, since the h_i's are used in the refresh. For example, the pseudofermion ϕ_k in the action term

$$\int d\phi_k e^{\phi_k^\dagger \frac{D^\dagger D + h_{k+1}^2}{D^\dagger D + h_k^2} \phi_k} \tag{8}$$

is generated from

$$\phi_k = P_e (D_0 + h_{k+1})^{-1} (D_0 + h_k) \eta_k \tag{9}$$

where η_k is a Gaussian random field, D_0 is the staggered Dirac operator using the gauge field from the start of the trajectory, and P_e is a projection onto even sites. The gradient of this expression with respect to the Hasenbusch masses must be included everywhere the pseudofermions enter in the update steps.

3. Integrators

Here we used four different MD integrator patterns. Two of the integrators, ABABA and ABACABA, were tested previously [4]. Here A stands for an update of the gauge field, B for an update of the momentum using both the gauge and fermion forces, and C is a force-gradient update

(more details on that below). We introduce two more integrators, in order to reduce the number of force evaluations from the pseudofermion actions. The new G5F2 integrator can be written in a similar notation as AGABAGABAGA where G stands for a momentum update using only the gauge force. This integrator has 5 gauge force evaluations and 2 fermion forces. We also consider a variation of this, G5F2,3 which distinguishes the light mass fermion force (L), from the heavy one (H), and can be written as AHALAHALAHA, where each of the fermion forces (H,L) also includes the gauge force (though each with their own tunable coefficient).

The force-gradient update, denoted as C in our notation, uses a variation of the Hessian-free force-gradient update step introduced in [7],

$$U_{\rm temp} = e^{aF(U)}U \tag{10}$$
$$p' = p + bF(U_{\rm temp}) \,, \tag{11}$$

where F is a force and a, b are parameters which are typically set analytically to cancel errors at a given order. In our case we combine the gauge and fermion force updates together, but allow each to have their own tunable coefficients

$$U_{\rm temp} = e^{a_g F_g(U) + a_f F_f(U)}U \tag{12}$$
$$p' = p + b_g F_g(U_{\rm temp}) + b_f F_f(U_{\rm temp}) \,, \tag{13}$$

where g (f) denotes the gauge (fermion) coefficient and force term. This form is very flexible and allows the gauge and fermion force updates to use different time scales within a single force-gradient evaluation.

4. Numerical tests

We tested automated tuning of the HMC parameters using a plain staggered action (4 continuum flavors) with mass $m = 0.04$. The plaquette gauge action coupling was $\beta = 5.4$ on a $12^3 \times 24$ lattice. In all cases we started from a thermalized configuration and used 200 trajectories for tuning followed by 400 trajectories for measurements. The MD trajectory length, τ, and the improved MD integrator parameters were always included in the tuning.

In Figure 1 we show the cost versus Hasenbusch mass for HMC with $n_s = 1$ MD step and one Hasenbusch mass, for a variety of MD integrators. The blue points are for fixed Hasenbusch mass (hMass) while tuning MD parameters, and the red points include tuning hMass using different hMass values as a starting point (with the starting value shown at the beginning of the grey line).

We see that in all cases, tuning hMass reliably ended near the minimum of the cost function. In this case the best integrator was the force gradient (ABACABA).

In Figure 2 we show results for two Hasenbusch masses, again with $n_s = 1$ MD step and using the ABABA integrator. The left panel shows a 3D surface plot of the cost function with fixed h1 and h2 while tuning the MD parameters, and the right panel shows a contour plot of the same data. Runs including tuning of h1 and h2 are also shown with the initial values for h1 and h2 in blue and the final tuned values in red. In this case the minimum of the cost function is very shallow in the h2 direction. The gradient-based tuning was still able to find the minimum reasonably well.

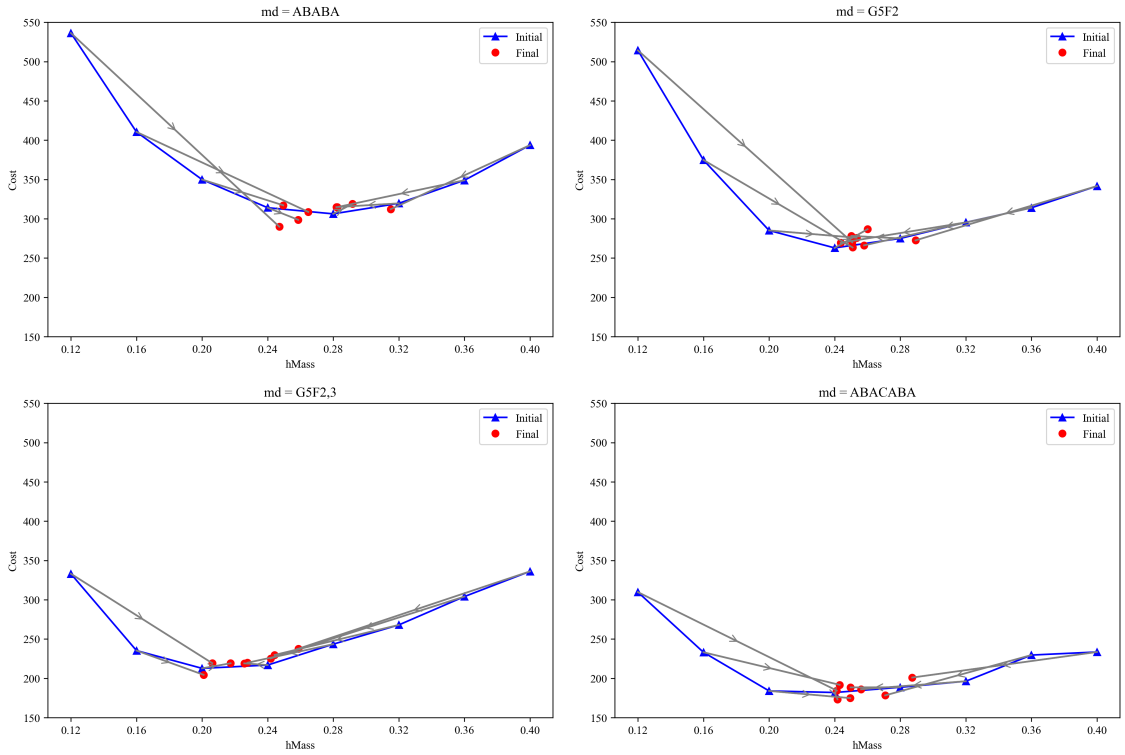

Figure 1: Cost function vs. Hasenbusch mass (hMass) for tuned HMC with $n_s = 1$ MD step and one Hasenbusch mass. Different MD integrators were used in each panel ABABA (upper left), G5F2 (upper right), G5F2,3 (lower left), ABACABA (lower right). The blue points are for fixed hMass while tuning MD parameters, and the red points include tuning hMass using different hMass values as a starting point (shown at the beginning of the grey line).

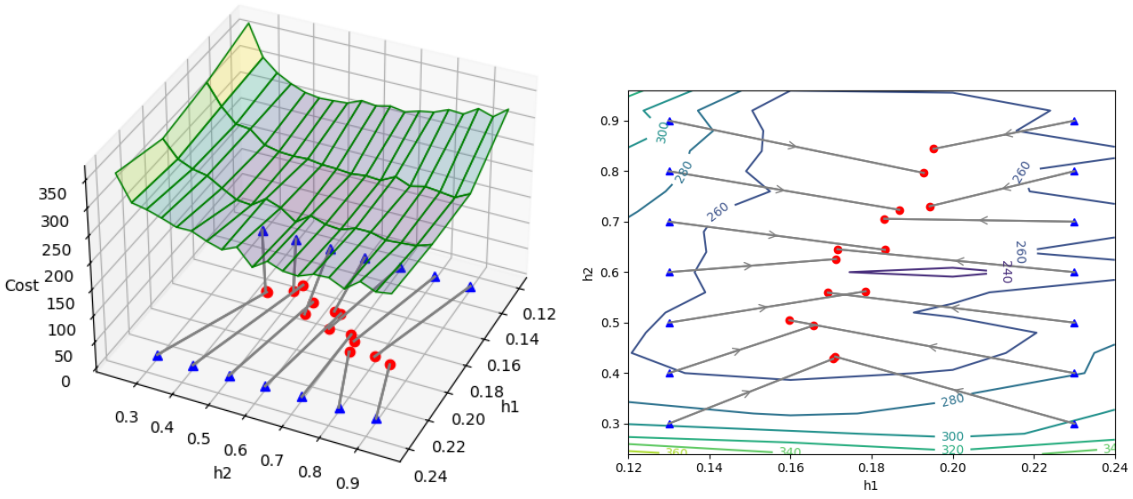

Figure 2: Cost vs. Hasenbusch masses (h1, h2) for HMC with $n_s = 1$ MD step, 2 Hasenbusch masses, and the ABABA integrator. The left panel shows a 3D surface plot of the cost function with fixed h1 and h2 while tuning the MD parameters, and the right panel shows a contour plot of the same data. Runs including tuning of h1 and h2 are also shown with the initial values for h1 and h2 in blue and the final tuned values in red.

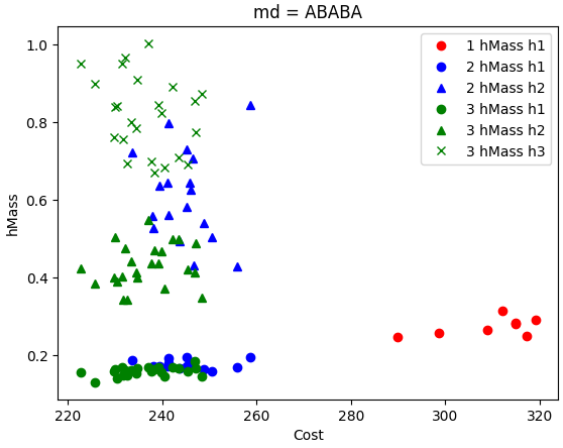

Figure 3: Tuned values for runs with 1, 2 & 3 Hasenbusch masses vs. cost with $n_s = 1$ step of ABABA integrator. For each case, results from several tuning runs are shown.

In Figure 3 we show tuned values from runs with 1, 2 & 3 Hasenbusch Masses using $n_s = 1$ step of the ABABA integrator. For each case, results from several tuning runs are shown. For one Hasenbusch mass (red points), we see that the mass is fairly well constrained, though there is about a 10% spread in the cost values due to statistical fluctuations. For two Hasenbusch masses (blue points), the cost is much smaller than for one mass. Here h_1 is relatively well determined, similar to the one mass case, but h_2 has a large spread. For three Hasenbusch masses (green points), the cost is only slightly lower than for two masses. The value for h_1 is similar to the two mass case. The values for h_2 and h_3 seem to split the range of h_2 values from the two mass case into two groups, with h_3 extending the range up a bit. For the ensemble studied here, there isn't much benefit to having more than two Hasenbusch masses. We would need to go to a larger, lighter mass ensemble to explore tuning with more masses.

In Figure 4 we show the dependence of the tuned values for two Hasenbusch masses on the number of MD steps using the ABABA integrator. As in the right panel of Figure 2, we show the contours of the cost function at fixed Hasenbusch masses while tuning the MD parameters, along with the ending mass values (in red) for runs including tuning of the masses with different starting points (in blue). Moving from $n_s = 1$ to $n_s = 2$ to $n_s = 4$ we see a trend of the cost minimum moving to larger values of h_1 and h_2. In these cases the tuning was able to reliably move towards the minimum, though for some starting points, it seems to have not converged to a minimum value yet. At $n_s = 8$ the minimum seems to be shallower and is influenced by large statistical fluctuations. Due to the shallow and noisy minimum, the tuning wasn't as effective here.

In Figure 5 we show the tuned cost vs. trajectory length for increasing number of MD steps for the ABABA and force gradient (ABACABA) integrators. The cost per update trajectory, C_S, is proportional to the number of MD steps, $n_s = \tau/\epsilon$. This gives for the cost

$$\text{Cost} = \frac{C_S}{\langle P_a \rangle \tau^2} \propto \frac{1}{\langle P_a \rangle \epsilon \tau} \tag{14}$$

If ϵ were held fixed, then the ideal scaling for the cost would be τ^{-1}. The acceptance will, however, decrease for increasing τ which will reduce the scaling, and also the optimized value of ϵ will vary

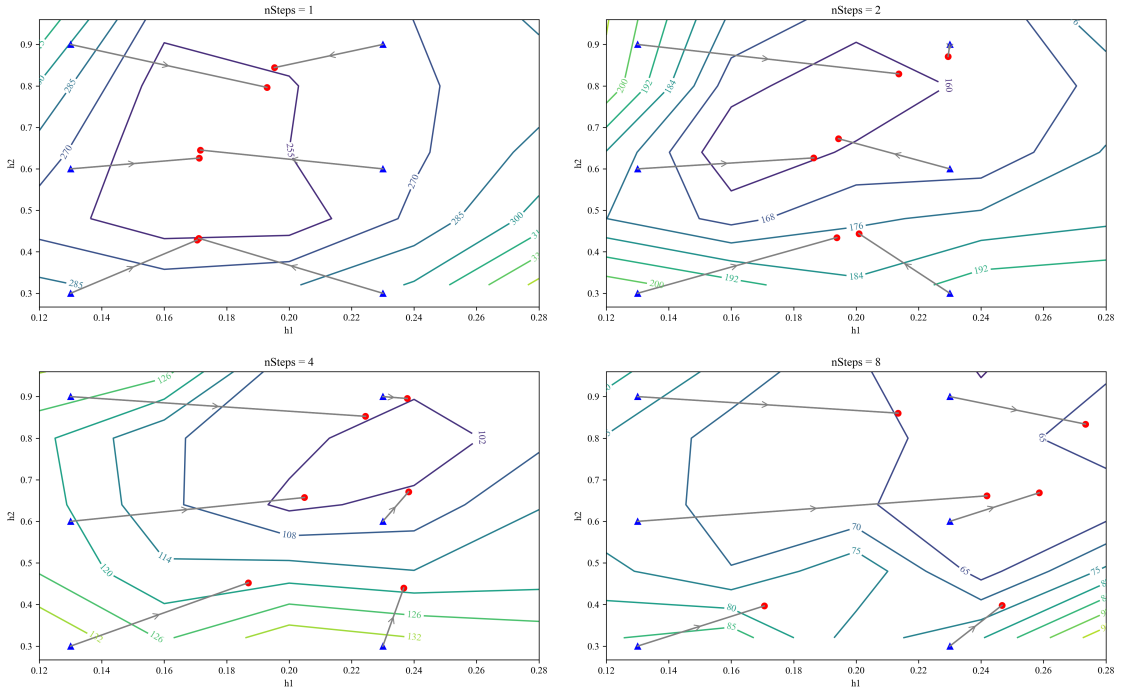

Figure 4: Dependence of the tuned values for two Hasenbusch masses on the number of MD steps (nSteps $\equiv n_s = 1, 2, 4, 8$) using the ABABA integrator. The $n_s = 1, 2, 4$ plots show a trend of the cost minimum to move to larger h_1 and h_2 as n_s is increased. The $n_s = 8$ plot seems to have a shallower minimum and is likely influenced by large statistical fluctuations.

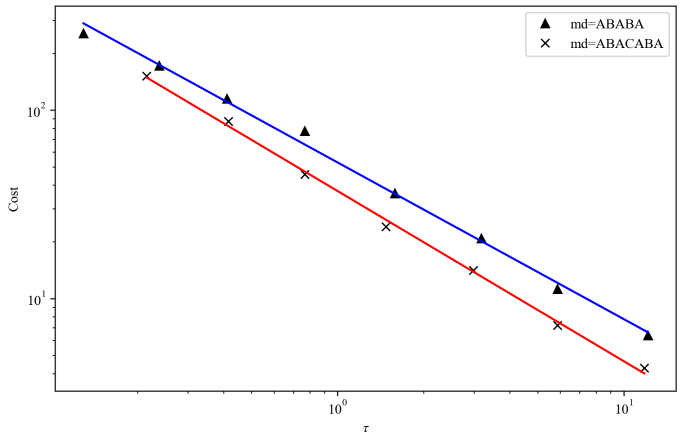

Figure 5: Tuned Cost vs. trajectory length for increasing number of steps. The blue line is a power-law fit to the ABABA integrator points with −0.83 as the fit value of the power. The red line is a power-law fit to the force-gradient (ABACABA) integrator points with −0.90 as the fit value.

some too. We find for the ABABA integrator, Cost $\propto \tau^{-0.83}$ while for ABACABA, Cost $\propto \tau^{-0.90}$.

Of course, the usefulness of increasing τ is decreased as it approaches the auto-correlation time for the observables of interest. This effect is not currently modeled in our cost function and we plan to consider that in future studies.

5. Summary

We found that one can effectively perform automated tuning of HMC parameters, including Hasenbusch mass ratios, using gradient-based optimization techniques. The automated tuning worked well for four-flavor staggered HMC with a variety of MD integrators and for up to three Hasenbusch masses, which was the largest feasible with the lattice size used. We found that the tuning also worked well as the number of MD steps was increased up to $n_s = 4$, though at $n_s = 8$ the minimum surface of the cost function seemed very noisy, and thus the autotuning didn't work as well. This requires further study to determine if it is simply a matter of requiring more statistics.

Overall, the autotuning method presented provides a convenient way to optimize HMC parameters. To continue this work, we are planning to extend the method to other types of HMC actions (improved gauge, quark smearing, RHMC) and test on production ensembles. We also plan to improve modeling of the cost estimate to include the full HMC costs and observable autocorrelation times.

Acknowledgments

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. This work also used the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, Office of High Energy Physics HEP User Facility. Fermilab is managed by Fermi Forward Discovery Group, LLC, acting under Contract No. 89243024CSC000002. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program. J.T. was supported by the DOE Computational Science Graduate Fellowship program.

References

[1] S. Duane, A. Kennedy, B.J. Pendleton and D. Roweth, *Hybrid Monte Carlo*, *Phys. Lett. B* **195** (1987) 216.

[2] T. Takaishi and P. de Forcrand, *Testing and tuning symplectic integrators for the hybrid Monte Carlo algorithm in lattice QCD*, *Phys. Rev. E* **73** (2006) 036706 [hep-lat/0505020].

[3] I.P. Omelyan, I.M. Mryglod and R. Folk, *Optimized Verlet-like algorithms for molecular dynamics simulations*, *Phys. Rev. E* **65** (2002) 056706 [cond-mat/0110438].

[4] J.C. Osborn, *Tuning HMC parameters with gradients*, *PoS* **LATTICE2023** (2024) 023 [2402.04976].

[5] M. Hasenbusch, *Speeding up the hybrid Monte Carlo algorithm for dynamical fermions*, *Phys. Lett. B* **519** (2001) 177 [hep-lat/0107019].

[6] D.P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, *Proc. 3rd International Conference for Learning Representations* 1412.6980.

[7] H. Yin and R. Mawhinney, *Improving DWF Simulations: Force Gradient Integrator and the Mobius Accelerated DWF Solver*, *PoS* **Lattice 2011** (2012) 051 [1111.5059].