Lattice Quantum Chromodynamics

Day 1: Calculating simple gauge field observables

Day 2: Determination of the lattice scale $\sqrt{t_0}/a$

Oliver Witzel
Email: oliver.witzel@uni-siegen.de

September 27, 2022

Abstract

This 2-day project provides an introduction to the concept of lattice quantum chromodynamics (QCD). The aim of day 1 is to determine simple gauge field observables. On day 2, we implement another, slightly more complicated, gauge field observable and embark on determining the lattice scale $\sqrt{t_0}/a$. The project combines the different tasks building up a typical lattice calculation on high performance super computers:

- Implementing gauge field observables using the lattice QCD software `Qlua` (based on the scripting language `lua`)
- Performing the numerical determination of the observables (measurements) by writing and submitting job-scripts to a scheduler (`SLURM`)
- Parsing the output data and performing a statistical data analysis

This project requires programming skills and familiarity with a `linux` command shell. The numerical calculations will be performed on the university's high performance compute cluster (`OMNI`). The different tasks to be accomplished and to be documented in the report are highlighted using the keywords **Tasks** and **Question**.

Prior to starting this project

- An account on `OMNI` needs to be setup
- Files relevant for this project need to be transferred

1 Introduction

1.1 Quantum Chromodynamics

The strong interactions of quarks and gluons are described by the theory called quantum chromodynamics (QCD). QCD is a non-Abelian quantum field theory with SU(3) gauge group. Quark and gluons carry a color charge which can take three different values. One of the characteristic features of QCD is color confinement i.e. no isolated color charges exist in nature. When two color sources are separated, the strong force between them grows so strongly that a quark-antiquark pair can be generated and the two color sources split into two (or even more) color-less bound states. These color-less bound states are generally called hadrons. The two most common types are quark-antiquark pairs named mesons and three quark states referred to as baryons. While in total six different quark flavors (up, down, strange, charm, bottom, top) are known resulting in a large zoo of particles studied experimentally, the two lightest up and down flavors are the dominant constituents of ordinary matter. Predominantly matter is made up of protons and neutrons, three particle baryons with either two up and one down quark (proton) or one up and two down quarks (neutron). The most common mesonic state are the pions with the π^+ composed of an up and anti-down quark (or short $u\bar{d}$), its anti-particle is the π^- ($d\bar{u}$), and further the π^0, a mixture of $u\bar{u}$ and $d\bar{d}$ exists.

In addition to the color charge, quarks carry an electric charge and also have a spin which leads to global chiral symmetry. This chiral symmetry gets however spontaneously broken which results in hadron masses much larger than the masses of the constituent quarks and leads to the fact that the pseudoscalar meson (pions) are much lighter than other states of the spectrum. In 2008 Yoichiro Nambu was awarded the Nobel Prize in Physics for his work on spontaneous symmetry breaking in subatomic physics.

Another characteristic feature of QCD is asymptotic freedom: When the energy scale of interactions increases (corresponding to decreasing length scales), the strength of the interaction between quarks and gluons reduces. For the discovery of asymptotic freedom David Gross, Frank Wilczek, and David Politzer were awarded the Physics Nobel Prize in 1973. While for large energies perturbative descriptions of QCD work well, we need truly nonperturbative methods to theoretically study QCD at low energies. In the next Section we introduce the concept of lattice field theory which provides an ab-initio nonperturbative framework to study QCD, typically referred to as lattice QCD. The basic idea of lattice QCD is to perform numerical simulations based on the QCD Lagrangian and was introduced by Nobel

laureate Kenneth G. Wilson in 1974 [1].

The QCD Lagrangian is given by

$$\mathcal{L}_{\mathrm{QCD}} = \bar{\psi}_i(x)\left(i\left(\gamma^\mu D_\mu\right)_{ij} - m\delta_{ij}\right)\psi_j(x) - \frac{1}{4}G^a_{\mu\nu}(x)G^{\mu\nu}_a(x), \tag{1}$$

where the first term describes the fermion contribution and the second the contribution of the gauge field. The quark field $\psi(x)$ is a function of the space-time (x) and the spinor indices i, j denote the quark fields are given in the fundamental representation of the SU(3) gauge group. D_μ is the gauge covariant derivative, γ^μ are Dirac matrices, m the mass of the quark, and δ_{ij} is a Kronecker delta. The gauge invariant gluon field strength tensor $G^a_{\mu\nu}$ is defined by

$$G^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu, \tag{2}$$

with $A^a_\mu(x)$ the gluon field as a function of the space-time in the adjoint representation of the SU(3) gauge group carrying the color indices a, b, c. The gauge coupling is denoted by g and f^{abc} are structure constants of SU(3).

In general, field theoretical quantities of interest are Green's functions. A Green's function is the vacuum expectation value of a time ordered product of field operators $\phi(\vec{x}, t)$

$$\langle 0|\phi(\vec{x}, t_1)\phi(\vec{x}, t_2)\ldots\phi(\vec{x}, t_n)|0\rangle \qquad \text{with} \quad t_1 > t_2 > \ldots > t_n. \tag{3}$$

Such Green's functions can be expressed in terms of functional integrals and calculated using the concept of Feynman path integrals i.e. we perform an integral over all possible paths from the initial state to the final state

$$\begin{aligned}&\langle 0|\phi(\vec{x}, t_1)\phi(\vec{x}, t_2)\ldots\phi(\vec{x}, t_n)|0\rangle \\ &\qquad = \frac{1}{Z}\int\left[\prod_{\vec{x},t} d\Phi(\vec{x}, t)\right]\Phi(\vec{x}, t_1)\Phi(\vec{x}, t_2)\ldots\Phi(\vec{x}, t_n)e^{iS},\end{aligned} \tag{4}$$

where S is the action of the system and the expression is normalized by

$$Z = \int\left[\prod_{\vec{x},t} d\Phi(\vec{x}, t)\right]e^{iS}. \tag{5}$$

Since the exponents in Eqs. (4) and (5) are imaginary, the integrand oscillates and convergence is not guaranteed. That can be remedied by performing a simultaneous Wick rotation of all times from Minkowski to Euclidean time: $t = -i\tau$.

1.2 Lattice Field Theory

After rotating the QCD Lagrangian (Eq. (1)) to Euclidean time, the path integral formalism provides a framework to numerically calculate QCD quantities like the masses of hadrons. However, before attempting any numerical calculation, we need to render the problem finite by discretizing 4-dimensional space-time using a hypercubic grid and confining the volume to a box of finite extent. Hence we yield a lattice where neighboring points are separated by the lattice spacing a and the coordinates x_μ of the lattice points are given by integer multiples of a:

$$x_\mu = a n_\mu \qquad \text{with} \quad n_\mu \in \mathbb{Z} \quad \text{and} \quad \mu = 0, 1, 2, 3. \tag{6}$$

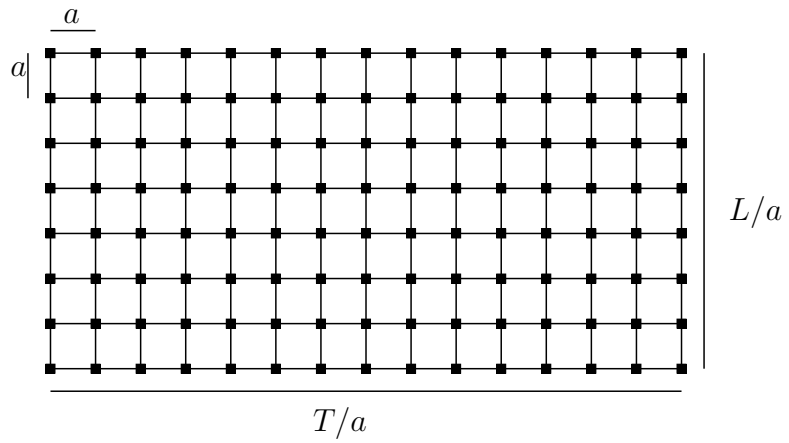

Figure 1: Sketch of a 2-dimensional lattice with $L/a \times T/a$ sites and lattice spacing a.

Typically we simulate a box with equal extent in the spatial directions, denoted by L, and a temporal direction labeled T. Considering a, L, and T to carry units of length [fm], the total number of lattice sites is given by

$$V = (L/a)^3 \times T/a. \tag{7}$$

A simple sketch (in easier to draw 2-dimensions) is shown in Fig. 1. Further we need to specify what we do at the boundaries of the lattice and choose to connect them periodically i.e. the 2-dimensional Grid in Fig. 1 becomes a "doughnut" and in four dimensions we work with a (hyper-)torus.

Next we need a prescription to discretize the fermion fields ψ and the gauge field A^a_μ. The fermion fields[1] ψ live on the lattice sites x_μ (black

[1]Here we are going to only calculate observables of the gauge fields and will therefore skip any further details on discretizing or simulating fermion fields.

squares in Fig. 1) and the gauge field is assigned to the links connecting two neighboring lattice sites (black lines). We introduce the link variable of the gauge field U and use the notation

$$U(x + a\hat{\mu}, x) \equiv U_{x,\mu} \tag{8}$$

to denote a link pointing from lattice site x in $\hat{\mu}$ direction. This setup defines the general framework of lattice field theory (LFT). For QCD the link variables are elements of SU(3) and defined by

$$U_{x,\mu} \equiv \exp\left\{ig_0 a A^c_\mu(x) T_c\right\}, \tag{9}$$

where $A^c_\mu(x)$ is the discretized Lie algebra valued gauge filed, T_c the generators of the gauge group, and g_0 the bare coupling. The smallest closed path on the lattice is called a plaquette which is the smallest 1×1 Wilson loop. The product of its four gauge links defines the plaquette variable $P_{x,\mu\nu}$. (The detailed definition is presented in Sec. 2.1.1.) By performing a sum over all plaquettes of the gauge field U, we arrive at the definition of Wilson's plaquette gauge action [1]

$$S_W = -\sum_{\substack{\text{all}\\ \text{plaquettes}}} \frac{2}{g_0^2} \mathrm{Re}\left(\mathrm{Tr}\left(1 - P_{x,\mu\nu}\right)\right). \tag{10}$$

When taking the naive continuum limit ($a \to 0$), we recover the Yang-Mills action

$$S_W = \frac{1}{4g_0^2} \sum_x a^4 F^c_{\mu\nu} F^c_{\mu\nu} + \mathcal{O}(a^6). \tag{11}$$

The integral over all gauge field configurations on the lattice corresponds to an integral over all link variables $U_{x,\mu}$ i.e. we obtain the expectation value for any observable A by calculating

$$\langle A \rangle = \frac{1}{Z} \int \left[\prod_{x,\mu} dU_{x,\mu}\right] A e^{-S_W}, \tag{12}$$

where the integration $dU_{x,\mu}$ for a given link x, μ is to be understood as the invariant integration over the group manifold with normalization

$$\int dU = 1. \tag{13}$$

Now we have introduced all ingredients to attempt a first quantum chromodynamics calculation on the lattice. What remains is to calculate expectation values. On a finite lattice calculating expectation values requires to evaluate finite dimensional integrals which calls for numerical methods. However, performing a simple numerical quadrature is not feasible: a typical lattice with volume $32^3 \times 64$ has about 2.1 million lattice sites and 8.4 million link variables which amounts for the SU(3) gauge group to 67 million real variables.

To tackle such a problem we need Monte Carlo methods with importance sampling i.e. for a given lattice action S we generate a set of points x_i with a probability

$$P(x_i) \sim \exp\{-S(x_i)\}. \tag{14}$$

This allows us to better sample the important regions of the integral and significantly improve the overall accuracy. For a lattice gauge theory we choose configurations $U^{(i)} = \left\{U^{(i)}_{x,\mu}\right\}$ to sample the integral. Calculating the expectation value

$$\langle 0|A|0\rangle = \frac{1}{Z}\int \mathcal{DU}\ A(U)\ e^{-S(U)} \tag{15}$$

is then numerically approximated by the average

$$\bar{A} \equiv \frac{1}{n}\sum_{i=1}^{n} A(U^{(i)}), \tag{16}$$

where the index i runs over a sequence (ensemble) of gauge field configurations $U^{(i)}$. The gauge field configurations are generated using Monte Carlo simulations creating a Markov chain i.e. a sequence of configurations $U^{(1)} \to U^{(2)} \to U^{(3)} \to \ldots$ according to appropriate probabilities. For this project these configurations have already been generated and two sets of suitable configurations are provided as binary files ready to use. Calculating expectation values with Monte Carlo techniques further implies to perform a statistical analysis i.e. our results carry statistical uncertainties

$$\langle A\rangle = \bar{A} \pm \delta_{\bar{A}}, \tag{17}$$

where $\langle\cdot\rangle$ indicates the average over the sequence of configurations and $\delta_{\bar{A}}$ is the statistical error of the average derived from the variance $\sigma^2_{\bar{A}}$

$$\delta_{\bar{A}} = \sqrt{\frac{\sigma^2_{\bar{A}}}{n}}. \tag{18}$$

Moreover there are systematic effects which need to be accounted for and we need to correct for effects due to the finite volume, and remove the discretization by taking the continuum limit. However, details on that are beyond the scope of this project.

In addition to the plaquette (the 1×1 Wilson loop) we will later discuss the planar 2×1 rectangle as well as the so called clover operator formed by the four plaquettes around the lattice site x in the μ-ν plane. Rectangle and clover operator can e.g. be used to reduce discretization errors in the definition of the gauge actions. With appropriately determined coefficients, this leads e.g. to the Symanzik [2, 3] or the Wilson-clover [4] gauge actions. These actions have different discretization artifacts. Values for bare parameters, like the gauge coupling g_0 in Eq. (9), depend on the chosen gauge action. All simulations are performed at a finite value of the lattice spacing a and the actual value of a needs to be numerically determined. To obtain physically meaningful results, simulations at different values of the bare gauge coupling are combined to subsequently take the continuum limit i.e. performing the extrapolation $a \to 0$. By taking the continuum limit, discretization artifacts are removed and universal results independent of the used discretization (gauge action) are obtained.

Plaquette, rectangle and clover operators are the key quantities to determine properties of the gauge field. They can be related to the energy density and allow to determine the lattice spacing or study nonperturbatively the renormalization group (RG) β function. Later we also introduce the Polyakov loop [5] which forms a closed loop of all links in one direction. Polyakov loops are frequently used as an order parameter to study the deconfinement transition of QCD simulations at finite temperature.

Further introductions to lattice field theory and lattice QCD can e.g. be found in

- Münster, "Lattice quantum field theory", (2010), Scholarpedia, 5(12):8613
- Weisz and Majumdar, "Lattice gauge theories", (2012), Scholarpedia, 7(4):8615
- Gattringer and Lang, "Quantum Chromodynamics on the Lattice", Springer (2010)
- Knechtli, Günther, Peardon, "Lattice Quantum Chromodynamics", Springer (2017)
- DeGrand and DeTar, "Lattice methods for quantum Chromodynamics", World Scientific (2006)

- Montvay and Münster, "Quantum Fields on a Lattice", Cambridge University Press (1994)

1.3 Workflow of a LFT calculation

A typical lattice calculation proceeds in several steps to account for the fact that the numerical costs and the required computational resources can be rather different.

1. Generate gauge field configurations
2. Determine observables (perform measurements)
   - Implement observables to be measured
   - Test and validate code
   - Run measurements for an ensemble of gauge field configurations
3. Perform a statistical data analysis
4. Combine data from simulations at different choices for the simulation parameters to obtain a result in the continuum and at physical parameter values free of discretization artifacts

Generating the gauge field configurations typically dominates the costs but the same gauge field configurations can be used to measure many different observables. Hence these gauge field configurations are saved as binary files and then subsequently read-in to carry out different measurements. This is exactly what we will do for this project and hence we can skip further details on how to generate these gauge field configurations. The example below refers to configurations [6] generated using the Symanzik gauge action with two massless flavors in the fundamental representation. The fermions were simulated using stout-smeared Möbius domain-wall fermions ($L_s = 12$, $M_5 = 1.0$, $\varrho = 0.1$, $N_{\text{stout}} = 3$) and the gauge coupling is set to be $\beta \equiv 6/g_0^2 = 4.60$. We are thus looking at a system which has QCD-like properties. The format of the binary gauge field is called `NERSC` and before the binary part 26 `ASCII` lines are stored containing some information about the gauge field. Not all predefined entries are filled. We can print the entire header information using the `head` command with the option `-26`

```
head -26 ckpoint_lat.500
```

```
BEGIN_HEADER
HDR_VERSION =
DATATYPE = 4D_SU3_GAUGE_3x3
```

```
STORAGE_FORMAT =
DIMENSION_1 = 32
DIMENSION_2 = 32
DIMENSION_3 = 32
DIMENSION_4 = 64
LINK_TRACE = -3.571047985e-05
PLAQUETTE  = 0.6460529819
BOUNDARY_1 = PERIODIC
BOUNDARY_2 = PERIODIC
BOUNDARY_3 = PERIODIC
BOUNDARY_4 = PERIODIC
CHECKSUM =   8d462b98
SCIDAC_CHECKSUMA =          0
SCIDAC_CHECKSUMB =          0
ENSEMBLE_ID = UKQCD
ENSEMBLE_LABEL = DWF
SEQUENCE_NUMBER = 1
CREATOR = witzel
CREATOR_HARDWARE = lq1wn056.fnal.gov-x86_64-Linux
    -3.10.0-957.21.3.el7.x86_64
CREATION_DATE =
ARCHIVE_DATE =
FLOATING_POINT = IEEE64BIG
END_HEADER
```

The focus of this project is on steps 2) and 3) and we will separately give the details for the actually provided gauge field configurations. Specifically we are interested in determining observables of the gauge field with the details given in the Section 2.1. On the technical side, we utilize a lattice QCD software package named `Qlua`[2] [7, 8]. This package provides an interface based on the scripting language `lua` to highly optimized routines written in `c/c++` which are parallelized using the `message passing interface (mpi)`. Luckily `Qlua` hides almost all of these technicalities and we can solely focus on the physics problem we like to solve.[3] At the same time we do take advantage of running a parallel program. **Thus we must always execute the program on a compute node of the `OMNI` cluster.** After implementing our measurement in a `Qlua` script, we can either submit a batch script (job script) to the resource manager (scheduler) of the cluster (`SLURM`) or we can request from the scheduler an interactive node. The latter may be more convenient for debugging/validating but is not suitable for running many measurements.

Next we discuss the specific example of calculating the average trace link

[2] https://usqcd.lns.mit.edu/w/index.php/QLUA

[3] **Important:** There is one pitfall to be aware of: array indices in `c` start with 0, whereas in `lua` they start with 1. This can get tricky when calling a `c` function from `lua`.

of a gauge field configuration. Key aspects of `lua`'s syntax are summarized in Appendix A and Appendix B lists the most relevant `SLURM` commands.

1.4 Example: Calculation of the average link trace

1.4.1 `Qlua` script

```
1  package.path = package.path .. ";./?.qlua"
2  require "stdlib"
3  require "Utils"
4
5  -- main program
6  L = 32
7  T = 64
8  config="../Two/f2l32t64ls12b460m000_3stout_pppa/ckpoint_lat.500"
9
10 -- Start timing
11 tic = os.time()
12
13 -- initialize lattice
14 lattice, volume, rand = InitLattice(L,L,L,T)
15
16 -- load gauge field in NERSC format
17 U, GaugeInfo = LoadGaugeField(config, lattice, volume)
18
19 -- calculate average link trace
20 ltr = 0
21 for mu = 1, #lattice do
22     ltr = ltr + U[mu]:trace():sum():real()
23 end
24 ltr = ltr / (U[1]:colors() * #lattice * volume) -- normalize
25 diff = (ltr - GaugeInfo.LINK_TRACE) / ltr -- calc. rel. diff.
26 printf("link trace: %d %15.11f %15.7e\n", volume, ltr, diff)
27
28 -- Finish timing
29 toc = os.time()
30 printf("Qlua finished. Total time: %f sec\n", toc-tic)
```

Meas.qlua

- As in line 5, comments start with two hyphens (`--`) and instruction do not need to be closed with semicolon (`;`) as e.g. in `C`. You may however use a semicolon to put two or more instruction in one line. Strings are enclosed using double quotes (`"`). The hash-operator (`#`) returns the number of elements of an array which in `lua` has an index running from one 1 to `#array`

- Line 1 specifies where `Qlua` searches for additional files like the file `Utils.qlua` (included in line 3) which provides the routines to setup the lattice or load a gauge field configuration. The listing of the file `Utils.qlua` is shown in Appendix C

- Line 6, 7 specify spatial and temporal extent of the lattice and the values must match the size of the gauge field specified by the filename in line 8 and referred to by the variable `config`

- Line 14 calls the routine to initialize the lattice i.e. information is generated how to access e.g. the next neighbors at any lattice site x

- Line 17 loads the gauge field into the variable `U` and the information from the header to `GaugeInfo`. Technically the gauge field U is an array with an index running over all lattice sites x of the 4-dimensional lattice volume. At each lattice site four link variables are stored corresponding to the four space-time directions which we typically access by an index μ or ν. Each link variable is an element of $SU(3)$ i.e. a complex 3×3 matrix. The index for the site x is "hidden" and operations act on the entire gauge field `U`. Since overloading allows to redefine the + operator for SU(3) color matrices, the link variables can straight forwardly be added without explicitly running over the elements of each SU(3) matrix. Similarly − and $\star$ operators are redefined for SU(3) color matrices

- The loop in lines 21-23 runs over the four space-time dimensions and performs the parallel computation of calculating the trace of each link variable `U[mu]`, summing over the space-time coordinate x, and taking the real part. By taking the trace of `U[mu]`, we remove the color index and instead of a matrix we have a complex number for each lattice site. Next the `:sum()` operation performs a reduction over the full 4-dimensional lattice and we obtain a single complex number. Finally `:real()` returns only the real part of that complex number which is of physical relevance

- The resulting sum is normalized in line 24 where `U[1]:colors()` provides the number of colors, `#lattice` the dimensions of the lattice and `volume` the number of lattice sites

- In line 25 we calculate the relative difference to the value saved as part of the header information of the gauge field file accessed via the variable `GaugeInfo`

- The special `printf` command in line 26 and 30 ensures that the output is only written by one `mpi` process and the usual syntax for formatted output can be used

1.4.2 SLURM job script

```
1  #!/bin/bash
2  #SBATCH -N 1
3  #SBATCH -p debug
4  #SBATCH --time 00:10:00
5  #SBATCH --tasks-per-node 64
6  #SBATCH --exclusive
7  Script=./Meas.qlua       # name of script to be executed
8
9  ###############################################################################
10 # redirect output and force immediate writing
11 exec > ${SLURM_SUBMIT_DIR}/${SLURM_JOB_NAME}-${SLURM_JOB_ID}.omni
       .out 2>&1
12
13 ###############################################################################
14 # start taking the time
15 echo "# time-begin "`date`
16 TotalTic=`date +%s`
17
18 ###############################################################################
19 #  report information on cluster
20 echo "------------------------------------------------------------"
21 echo "Running Qlua on OMNI"
22 echo "SLURM job runnning on: `hostname`"
23 echo "in directory:          `pwd`"
24 echo "SLURM jobid:           ${SLURM_JOB_ID}"
25 echo "SLURM #nodes:          ${SLURM_NNODES}"
26 echo "SLURM #tasks/node:     ${SLURM_TASKS_PER_NODE}"
27 echo "Nodefile: ${SLURM_JOB_NODELIST}"
28 echo "------------------------------------------------------------"
29
30 ###############################################################################
31 # load environment and set variables for running Qlua
32 umask 007
33 module load intel/19.1.3.100008_cm9.0_f654bdadee
34 module load impi/2020.4
35 MPIrun="srun --mpi=pmi2 "
36 Qlua="/home/ow907254/Software/QLUA/omni-20201002-intel19.1.3
       _impi2020.4/qlua/bin/qlua "
37
38 ###############################################################################
39 # print script and command to be executed to the log file
40 echo "============================================================"
```

```
41 echo "Script: ${Script}"
42 echo "=============================================================="
43 cat ${Script}
44 echo "=============================================================="
45 echo "${MPIrun} -n ${SLURM_NTASKS}  ${Qlua}  ${Script}"
46 echo "=============================================================="
47 echo ""
48
49 ##########################################################################
50 # run Qlua in parallel executing the script
51 ${MPIrun} -n ${SLURM_NTASKS}  ${Qlua}  ${Script}
52
53 ##########################################################################
54 # report the time
55 TotalToc=`date +%s`
56 echo " "
57 echo "# time-finish "`date`
58 TotalTime=$(( $TotalToc - $TotalTic ))
59 TotalHours=`echo "$TotalTime / 3600" | bc -l`
60 echo "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
61 echo "Total time  $TotalTime [sec] = $TotalHours [h]"
62 echo "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
```

Test.job.sh

- **Only lines 3, 4, and 7 may need adjustment!**
- Line 1 specifies that this script uses `bash` syntax i.e. comments start with #. To assign a value to a variable, use the equal sign (=) without any white space before or after "="; only when referring to a value of the variable, prefix $
- Lines 2-6 starting with `#SBATCH` specify the resources we request from the scheduler. Specifically we ask for one full node for our exclusive usage with the intention to spawn 64 `mpi` tasks. Line 3 names the partition (queue) to use, in this case `debug`. `debug` has the shortest (wall-)time limit and smallest number of nodes. `debug` is solely intended for testing/debugging with short turnaround times. In Line 4 we say our job will finish within 10 minutes. If it runs longer, `SLURM` will terminate it
- For production measurements changing line 3 to "-p short" and line 4 to (up to 2h) "--time 2:00:00" may be advised
- Line 7 names the `Qlua` script we want to run

- Subsequent lines do not need adjustment. All lines starting with `echo` simply print information to the log-file and the comments provide further explanations

- `sbatch Test.job.sh`

  submits the script and you may check the queue with

  `squeue -u $USER`

  (for further comments on `SLURM` commands see Appendix B)

1.4.3 Running interactively on a compute node

First we need to request an interactive node from the scheduler (single line)

```
srun --pty --exclusive -N 1 --time=00:30:00 --tasks-per-node 64
    -p debug /bin/bash
```

which specifies we want one full node of the debug partition (queue) for 30 min, using up to 64 `mpi` tasks, and running on it with the `BASH` shell. Once we submit this request, the terminal is blocked until the resource is available. Do check that the prompt states you are connected to a `hpc-node` and not still on `hpc-login`. When the interactive session has started, we need to load the software environment to run `Qlua`. For that execute

```
module load intel/19.1.3
module load impi/2020.4
```

and define as shortcut the alias (no space around the equal sign, one line)

```
alias Qlua="/home/ow907254/Software/QLUA/omni-20201002-intel19
    .1.3_impi2020.4/qlua/bin/qlua "
```

With this setup in place we can (repeatedly) execute `Qlua` and run our script

```
Qlua ./Meas.qlua
```

where we assume we are in the directory of the file `Meas.qlua`. The output will only be printed on the screen (`stdout`). If the program hangs or is not doing what you want, hit `CTRL-C` to terminate it. By just executing `Qlua` the code will be slow and only run on one processor. That option typically gives you the most useful error messages when seeking a bug which terminates the script. To run on more processors (up to 64 on a single node) prefix `mpirun -n` followed by the number of tasks (1, 2, 4, 8, 16, 32, or 64) and then run your script i.e.

```
mpirun -n 64 Qlua ./Meas.qlua
```

Each time you start a new interactive session, the specified modules must be loaded and the `Qlua` variable defined to use the shortcut.

2 Gauge field observables (day 2)

2.1 Implement gauge field observables

2.1.1 Plaquette (1×1 Wilson Loop)

The smallest closed path on the lattice is called a **plaquette** (see Fig. 2) which defines the plaquette variable $P_{x,\mu\nu}$ at a lattice site x in the μ-ν plane

$$P_{x,\mu\nu} \equiv U_{x,\mu} U_{(x+a\hat{\mu}),\nu} U^{\dagger}_{(x+\hat{\nu}),\mu} U^{\dagger}_{x,\nu}. \tag{19}$$

As indicated by the directions of μ and ν in the lower left corner of Fig. 2, the gauge links are oriented in the forward/upward direction. However, to close the loop at least one link needs to be included which runs in the opposite direction. In such cases we need the adjoint link indicated by †.

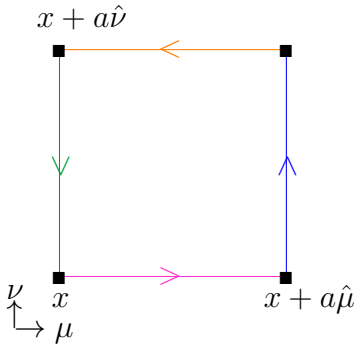

Figure 2: Sketch of plaquette at lattice site x in directions μ and ν. The colors of the link match the terms of the r.h.s. in Eq. (19).

Tasks: Calculate the average plaquette for a given gauge field configuration taking the trace over the color indices and summing over all plaquettes i.e. calculate

$$P = \sum_{\substack{\text{all} \\ \text{plaquettes}}} \mathrm{Tr}\{P_{x,\mu\nu}\}. \tag{20}$$

Only the real part has a physical meaning and the resulting value should be normalized such that the value of the average plaquette can take values from 0 to 1. Quote the following values:

- "world" (average of plaquettes in all four directions)
- "space-like" (plaquettes with space-like directions only)

- "time-like" (plaquettes along the temporal direction)

Question: Why is it mathematically irrelevant where you start calculating the plaquette or in which direction you go around the links?
Hints:

- To get the adjoint link $U^{\dagger}_{x,\mu}$, use the syntax

```
U[mu]:adjoin()
```

- Accessing the link from a next neighbor relative to the site x is done using the syntax

```
U[nu]:shift(mu, "from_forward")
```

  In this example, we grab the link for the direction ν at the site $x + a\hat{\mu}$ i.e. moving from x one step forward in the direction of μ. **When using μ for a shift, the index runs from 0 to 3, whereas when accessing the elements of `U[mu]` it runs from 1 to 4.** In the first case the index acts using a function written in `C`, while in the latter case the index operates on a field defined in `lua`

- To validate your code for the "world" plaquette of a given gauge field configuration, you may cross-check the value you calculated against the one recorded as part of the header information of the file you are reading

- When looping over all four space-time directions consider that each plaquette is a 2-dimensional quantity (potential issue of "double-counting")

- To sort out the normalization, you may calculate the average of a unit gauge field which you can create by using

```
require "gauge"  -- to be added at the top
U = {}
for i = 0, #lattice - 1 do
    U[i+1] = toSUn(lattice:ColorMatrix(complex(1,0)))
end
```

- To get average space-like and time-like plaquettes right, consider a small 4-dimensional lattice and count how many plaquettes in space-like and time-like directions exist. What do you expect to obtain for the sum of space-like and time-like plaquettes?

2.1.2 Rectangle (2×1 Wilson Loop)

The planar 2×1 and 1×2 rectangles (see Fig. 3) are the next larger and similarly simple Wilson Loops after the plaquette.

Tasks: Write down mathematical equations defining the 2×1 and 1×2 rectangle $R^{2\times1}_{x,\mu\nu}$ and $R^{1\times2}_{x,\mu\nu}$, respectively. Implement both rectangles choosing

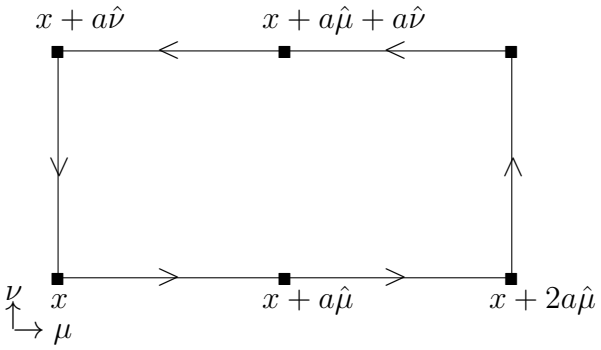

Figure 3: Sketch of the 2×1 rectangle in directions μ and ν.

a normalization consistent with the one for the plaquette. Calculate the average values $R^{2\times1} = \sum_{\substack{\text{all}\\\text{rectangles}}} \mathrm{Tr}\{R^{2\times1}_{x,\mu\nu}\}$ and $R^{1\times2} = \sum_{\substack{\text{all}\\\text{rectangles}}} \mathrm{Tr}\{R^{1\times2}_{x,\mu\nu}\}$ for a given gauge field configuration again quoting values for

- "world" (average of rectangles in all four directions)
- "space-like" (rectangles with space-like directions only)
- "time-like" (rectangles along the temporal direction)

Question: Without double counting, how many 2×1 and 1×2 rectangles exist? Does the answer depend on the number of space-time dimensions? How do their numbers compare to the number of plaquettes?

2.1.3 Testing gauge invariance

Plaquette, rectangle, clover operator, and Polyakov loops are all gauge invariant quantities — in contrast to e.g. the average link trace used as introductory example in Sec. 1.4. Hence we can perform certain symmetry transformations on the gauge field without changing the values of such gauge invariant quantities.

A simple (trivial) transformation is to shift the origin of the lattice by a 4-vector. This can be done using the function `translat` where the shift

is specified by the variable `xvec`. To test translational invariance, add the following two lines

```
xvec = {13,9,4,37}
U = translat(U,xvec)
```

right after loading the gauge field and before doing any calculation.
Questions: Can you confirm that this does not change the results? Next apply a non-trivial gauge transformation by multiplying an SU(3) matrix to the gauge field U using the function `gauge_transform_U`

```
local g = lattice:ColorMatrix(1.0)
g = g:proj(1e-8,150)
U = gauge_transform_U(g, U)
```

and check how that changes the values of the observables.

2.1.4 Polyakov Loop

Exploiting the periodic boundary conditions of the gauge field, we can define the Wilson loop with the largest extent in one direction. In the temporal direction this quantity is also referred to as the thermal Wilson line. Considering this largest loop in all four space-time directions, we generally refer to it as Polyakov loop i.e. we are interested in the quantity

$$\mathrm{Pl}_\mu = \mathrm{Tr}\left\{\prod_{k=0}^{L_\mu-1} U_\mu\right\} \qquad \text{with} \quad L_\mu = (L, L, L, T). \tag{21}$$

Tasks: Implement and print both the real and the imaginary part of the average Polyakov loop for all four space-time directions. For simulations in the confined phase the expectation value $\langle \mathrm{Pl}_\mu \rangle$ is expected to be zero, whereas it is nonzero for simulations in the deconfined phase.

2.2 Run measurements

Only after validating the code with the supervisor, you are ready to start a larger scale lattice calculation using all of the provided gauge field configurations. Please do not call the `translat` or `gauge_transform_U` function for the so called production run.

Modify the `SLURM` job script and the `Qlua` measurement script to loop over several configurations within one job. Take care of organizing the output such that you can "easily" and robustly extract the measurement values from log-files written to `stdout`.
Tasks: Please include a description of how you organized the output and post-processing in your report.

Hint:
Powerful and maybe useful command line tools are `grep` and `awk`.

2.3 Statistical data analysis

Perform a statistical data analysis for the plaquette and rectangle observables.
Tasks:

- Create history plots of the quantities vs. the Monte Carlo time (configuration number)
- Show a histogram for these observables and obtain mean values with their statistical errors
- Try to estimate effects of autocorrelation by binning subsequent measurements
- A statistical analysis for the Polyakov loops is likely not suitable. Here however it is interesting to make a scatter plot showing the real part on the horizontal and the imaginary part on the vertical axis overlaying the four different directions using different colors/symbols. Can you decide whether the provided configurations are in the confined or deconfined phase of QCD?
- Please make sure to report results for both data sets provided

3 Gradient flow scale setting (day 2)

Before we move on and use the previously implmented gauge field observables to determine the gradient flow (GF) scale $\sqrt{t_0}/a$, we first discuss implementing another observable, the clover operator $C^{\mu\nu}(x)$. Subsequently, we discuss the gradient flow and describe how to determine the lattice scale from it.

3.1 Implement the clover operator

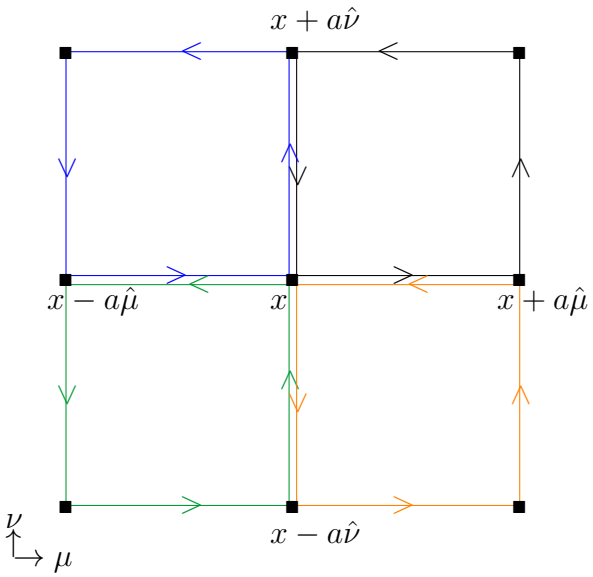

Figure 4: Sketch of clover operator for lattice site x in the μ-ν plane.

Figure 4 shows a sketch for the Sheikholeslami-Wohlert or clover operator at lattice site x which is given by the sum of the four different plaquettes at lattice point x in the μ-ν plane. The anti-Hermitian and traceless part of the average clover operator $C^{\mu\nu}(x)$ can be used to obtain a lattice definition of the field strength tensor $G^{\mu\nu}(x)$. Calculating $G^{\mu\nu}(x)$ on the entire lattice,[4]

[4]Note, that calculating $G^{\mu\nu}(x)$ for all x in the μ-ν plane requires to take all loops with the same orientation.

lets us define the energy density E

$$E = \frac{1}{4} G^a_{\mu\nu} G^a_{\mu\nu}, \tag{22}$$

which is proportional to the gaugecoupling g^2.
Tasks:

- First write down the corresponding mathematical expression to determine the clover operator $C^{\mu\nu}(x)$ in terms of the link variables $U_{x,\mu}$. How many "hops" to neighbors of x do you need to get all links? (Fewer hops is better.)
- The anti-Hermitian and traceless part of $C^{\mu\nu}(x)$ defines the lattice field strength tensor $G^{\mu\nu}(x)$ which we want to use to calculate the energy density E. Detail how you plan to obtain E from $C^{\mu\nu}(x)$.
- Implement the above described energy density E to determine the average for a given gauge field configuration. (Just determining the "world" energy density is sufficient.)

3.2 Determine the lattice spacing a

For simulations in the confining phase of QCD we can determine the lattice spacing a. One option is to use a gradient flow method [9, 10]

$$\frac{d}{dt} B_\mu(x,t) = -\frac{\partial S_{YM}(B)}{\partial B_\mu(x,t)} \qquad \text{with} \quad B_\mu(x,t)\Big|_{t=0} = A_\mu(x). \tag{23}$$

In Eq. (23) we introduced a new parameter, the flow time t which has mass dimensions [-2]. Further we define the Lie Algebra valued field $B_\mu(x,t)$ which compared to the gluon field A_μ has an additional dependence on the flow time t. By numerically integrating Eq. (23) for some specific Yang-Mills gauge action S_{YM} using the "unflown" gluon field A_μ as initial value, each gauge field configuration $U^{(i)}$ is expanded into a sequence of gauge fields $V_t^{(i)}$ along the flow time t. The gradient flow acts thereby as a smoothing procedure gradually removing UV fluctuations. By measuring a quantity related to the energy density E, we can then study its flow time dependence and deduce a lattice scale [11]. Practically we calculate for each flow time t the expectation value $\langle E(t) \rangle$ over the sequence of gauge field configuration. By forming the dimensionless product $t^2 \langle E(t) \rangle$ we obtain a quantity which is proportional to the gauge coupling

$$g^2_{GF}(t; L, \beta) = \frac{128\pi^2}{3(N_c^2 - 1)} \langle t^2 E(t) \rangle. \tag{24}$$

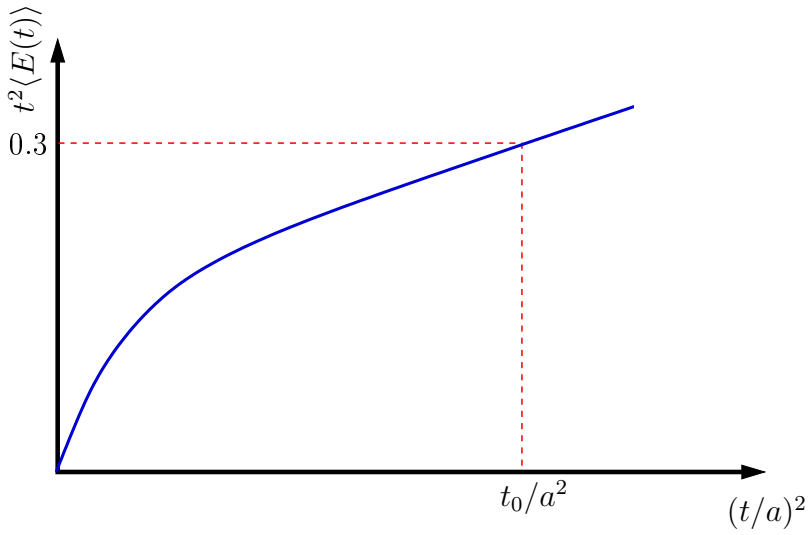

Figure 5: Sketch showing the determination of the Wilson flow scale $\sqrt{t_0}/a$ using $t^2\langle E(t)\rangle$.

The constants in Eq. (24) are chosen to match for $N_c = 3$ the perturbative 1-loop result in the $\overline{\text{MS}}$ scheme [11]. The flow time t for which $t^2\langle E(t)\rangle$ reaches a specific value finally defines our lattice scale.

In practice we use $S_{YM} = S_W$, the Wilson gauge action, and measure the energy density using the plaquette or the clover operator. The Wilson flow scale $\sqrt{t_0}/a$ is defined by [11]

$$\left\{t^2\langle E(t)\rangle\right\}_{t=t_0} = 0.3, \tag{25}$$

and we show a sketch of the t_0 determination using $t^2\langle E(t)\rangle$ data in Fig. 5. While we draw a continuous line for $t^2\langle E(t)\rangle$ as a function of the flow time t, the flow time t is necessarily a discrete variable incremented in steps of ε. Choosing $0.01 \leq \varepsilon \leq 0.04$ is appropriate for this project. $\sqrt{t_0}/a$ is a lattice scale i.e. it is given in lattice units a. We can convert it to a physical length scale [fm] by using its continuum limit value determined e.g. in [12]

$$\sqrt{t_0} = 0.1528 \text{ fm}. \tag{26}$$

To use the Wilson flow implementation in `Qlua` add the following definitions to the top

```
require "gauge"
require "qcdlib/gradient-flow"
  -- Flow parameters
```

```
eps = 0.04
f_steps = L*L/64/eps;
-- Action parameters for Wilson
act_params = { plaq1 = 3.0 }
-- build force and action functions from action parameters
g_force = qcdLib.GaugeLoops.flow_force(act_params)
```

and then run a loop

```
V = U
for i = 1, f_steps do
   V = qcdLib.GradientFlow.flow(g_force, V, eps)
   monitor_flow(i, eps, i * eps, V)
end
```

where you need to provide the function `monitor_flow`. The indicated arguments name the integer step i, the step-size ε, the flow time $t = i \cdot \varepsilon$, as well as the "flown" gauge field V.

Question: Why is it important to calculate for a given μ-ν plane all plaquettes of the clover operator using the same orientation?

Tasks:

- Validate you determination of the energy density using the clover operator with your supervisor without using gradient flow.
- Implement the function `monitor_flow` to perform the measurements of the gauge field observables on the gauge field V and print the result together with the integer step number i as well as the flow time $t = i \cdot \varepsilon$.
- Test your implementation on one gauge field configuration and check that the walltime-limit is sufficient to complete the measurements.
- Run your gradient flow measurements for all gauge field configurations.
- The analysis proceeds as before by calculating the mean of $t^2\langle E(t)\rangle$ and its error to create a plot showing $t^2\langle E(t)\rangle$ vs. t/a^2 (standard statistical error analysis).
- By finding the value of $t^2\langle E(t)\rangle = 0.3$, you can read off t_0/a^2 and deduce the Wilson flow scale $\sqrt{t_0}/a$ (see sketch in Fig. 23). Explain how you obtain the statistical error on $\sqrt{t_0}/a$?
- The determination of $\langle E(t)\rangle$ can be done using different operators to estimate the energy density. Compare how the results differ when using the plaquette compared to the clover operator.
- Use the provided reference value (Eq. (23)) to convert the lattice scale into physical units [fm].

A `lua`

- Comments

```
--single line comment
--[[
    multi-line comment
    ]]
```

- Variables
  Variables do not require a type definition and values are assigned using the equal sign (=). Similarly standard arithmetic operations `+`, `-`, `*`, `/` are defined. To concatenate strings use two dots (`..`)

```
local string = "part1" .. "part2"
```

  The scope of a variable can be explicitly limited by first declaring it with `local`. Also the standard relational operations are available: equal (==), not equal (~=), greater (>), greater equal (>=), less equal (<=), less (<) complemented by the logical operators `and`, `or`, `not`

- Loops

```
while <condition> do
end

for i = start,end,step do
    -- if step is omitted, step=1 is used
end

for k,v in pairs(tab) do
end

repeat
until <condition>
```

- Conditions

```
if <condition> then
   printf("yes")
elseif <condition> then
   printf("maybe")
else
   printf("no")
end
```

- Functions

```
function myFunction()
   return 1
end

function myFunction(a, b)
   return a + b
end
```

B SLURM

The ZIMT website https://cluster.uni-siegen.de/omni/usage/slurm/?lang=en has a detailed list of the various SLURM commands. Most relevant are properly:

- `sinfo` with option `-ls` reports a compact overview of the resources and the defined partitions (queues), whether the partition is available and what its maximal (wall-)time limit is as well as the number of nodes in the different states. `A:` allocated, `I:` idle, `O:` other (not available), and `T:` total. As can be seen in the last column, nodes can be assigned to more than one partition. `Qlua` is compiled for running on `hpc-nodes` only.

```
sinfo -ls
Sun Jul 11 11:03:15 2021
PARTITION AVAIL  TIMELIMIT  NODES(A/I/O/T) NODELIST
debug        up      1:00:00          0/5/0/5 hpc-node[001-005]
gpu          up 1-00:00:00         1/9/0/10 gpu-node[001-010]
htc          up 1-00:00:00        0/41/0/41 htc-node[001-041]
long         up 20-00:00:0        85/4/8/97 hpc-node[338-434]
medium       up 1-00:00:00   385/16/28/429 hpc-node[006-434]
short*       up     2:00:00   232/12/12/256 hpc-node[006-261]
smp          up 1-00:00:00          0/2/0/2 smp-node[001-002]
```

- `sbatch` submits a job-script. Options can be provided on the command line or using `#SBATCH` instructions as discussed for lines 1-6 of the run script in Sec. 1.4.2

- `squeue` reports all jobs the scheduler is currently managing. To get a list of only your jobs use

  ```
  squeue -u $USER
  ```

  where the environment variable `$USER` is by default equal to your username on `OMNI`

C Utils.qlua

```
function InitLattice(X,Y,Z,T)
  -- initializes lattice and random state
  local lat = qcd.lattice{X,Y,Z,T}
  local vol = 1
  local i
  for i = 0, #lat - 1 do
     vol = vol * lat[i]
  end
  -- Initialize the random state from the system source
  local r = { }
  local rand
  do
     r.x = os.random()
     local x = lat:Int(r.x)
     for i = 0, #lat - 1 do
        r[i+1] = os.random()
        x = x * lat[i] + lat:pcoord(i) * r[i+1]
     end
     r.f = os.random()
     printf("Initializing random state\n")
     rand = lat:RandomState(r.f, x)
  end
  return lat, vol, rand
end

-- ------------------------------------------------------------
function LoadGaugeField(fname, lat, volume)
   local U
   local info

   U, info = qcd.nersc.read_gauge(lat, fname,
                {unitarity=1.23e-7, FLOATING_POINT="IEEE64BIG"})

   printf("\n\nHEADER of %s\n", fname)
   local i
   local v
   for i, v in pairs(info) do
      if i == 'CHECKSUM' then
         v = string.format("%x", v)
         v = string.sub(v, 9, -1)
         printf("   %-20s   [number]  %s\n", tostring(i),  v)
      else
         printf("   %-20s   [%s]  %s\n", tostring(i), type(v),
          tostring(v))
      end
    end
```

```
    printf("\n")
    return U, info
end

--------------------------------------------------------------------------------
function translat(Utrans, xvec)
  local tmp = {}
  local n, dir, length, j
  printf("shift lattice by vector [")
  for n = 0, #Utrans-1 do
     dir = n
     length = xvec[dir+1]
     printf("%d ", length)
     if length > 0 then
       for mu = 1, #Utrans do
          for j = 1, length do
             tmp[mu] = Utrans[mu]:shift(dir,"from_forward")
             Utrans[mu] = tmp[mu]
           end
        end
      end
   end
   printf("]\n")
   return Utrans
end
```

Utils.qlua

References

[1] Kenneth G. Wilson. Confinement of Quarks. *Phys. Rev. D*, 10:2445–2459, 1974.

[2] M. Lüscher and P. Weisz. On-Shell Improved Lattice Gauge Theories. *Commun. Math. Phys.*, 97:59, 1985. [Erratum: Commun. Math. Phys.98,433(1985)].

[3] M. Lüscher and P. Weisz. Computation of the Action for On-Shell Improved Lattice Gauge Theories at Weak Coupling. *Phys. Lett.*, 158B:250–254, 1985.

[4] B. Sheikholeslami and R. Wohlert. Improved Continuum Limit Lattice Action for QCD with Wilson Fermions. *Nucl. Phys. B*, 259:572, 1985.

[5] Alexander M. Polyakov. Compact Gauge Fields and the Infrared Catastrophe. *Phys. Lett. B*, 59:82–84, 1975.

[6] Anna Hasenfratz and Oliver Witzel. Continuous renormalization group β function from lattice simulations. *Phys. Rev.*, D101(3):034514, 2020.

[7] Andrew Pochinsky et al. Qlua, 2008.

[8] Andrew Pochinsky. Writing efficient QCD code made simpler: QA(0). *PoS*, LATTICE2008:040, 2008.

[9] R. Narayanan and H. Neuberger. Infinite N phase transitions in continuum Wilson loop operators. *JHEP*, 0603:064, 2006.

[10] Martin Lüscher. Trivializing maps, the Wilson flow and the HMC algorithm. *Commun.Math.Phys.*, 293:899–919, 2010.

[11] Martin Lüscher. Properties and uses of the Wilson flow in lattice QCD. *JHEP*, 1008:071, 2010.

[12] A. Abdel-Rehim et al. First physics results at the physical pion mass from $N_f = 2$ Wilson twisted mass fermions at maximal twist. *Phys. Rev. D*, 95(9):094515, 2017.