Parameters of the openQCD main programs

Martin Lüscher March 2013; last revised February 2022

1. Introduction

The openQCD main programs have many adjustable parameters. Most parameter values are passed to the programs at run time through a human-readable parameter file. The only parameters that must be specified at compilation time are the lattice sizes and the MPI process grid (see `include/global.h` and `main/README.global`). Each program also has a few command-line options.

Some of the parameters are of a fairly trivial kind and are omitted in this note (see, instead, the `README` files that come with each program). Details about the simulation algorithm and the exact normalization conventions employed can be found in the documentation files in the `doc` directory.

2. Preliminaries

The input parameter files are divided into sections such as

```
[Solver 3]
solver CGNE
nmx 256
istop 0
res 1.0e-10
```

In this case, the section describes the solver number 3 for the Dirac equation. The solver program is `CGNE`, the conjugate-gradient (CG) algorithm for the normal Dirac equation, `nmx` specifies the maximal number of CG iterations that may be performed,

`res` the desired maximal relative residue of the calculated solutions and `istop` the stopping criterion to be used (see subsect. 5.1).

Sections start with a title in square brackets `[...]` and the lines within a section can be ordered arbitrarily. The section quoted above, for example, could equivalently be written as

```
[Solver 3]
res 1.0e-10
solver CGNE   # Should try a better solver
istop 0
nmx 256
```

Any text appearing to the right of the number sign `#` is considered to be a comment. A section is delimited by its headline (the text in square brackets) and the headline of the next section. Blank lines are ignored and sections can appear in any order in the input file.

Once the program has read the parameter file, the data are entered into a parameter data base and can, from there, be retrieved by the subprograms that depend on some of the parameter values. The data base is administered by a set of modules in the directory `modules/flags`. In case the exact meaning of a particular parameter is unclear, it may be helpful to read the explanations on the top of these program files.

3. Actions

The lattice theory is defined by the lattice sizes, the total action and the boundary conditions. A description of the latter and, in the case of the simulation programs, of all parts of the action must be included in the parameter file.

3.1 Action parameters and boundary conditions

The basic parameters of the theory are the inverse gauge coupling `beta`, the coefficient `c0` of the plaquette term in the gauge action [1], the hopping parameters `kappa` of the sea quarks as well as the variant `isw` of the Sheikholeslami–Wohlert term (0: traditional, 1: exponential) and its coefficient `csw` [2]. In the parameter files, their values are specified in a section of the form

```
[Lattice parameters]
beta 5.3
c0 1.66666667
kappa 0.13635 0.13620 0.13550
isw 0
csw 1.9078
```

There are three numbers on the line with tag `kappa` in this example, which could be the values of the light, strange and charm quark hopping parameters. In principle, any number of values can be listed here. Hopping parameter values are referred to by their position in the list, the first value having position 0, the second position 1, and so on.

The present version of the openQCD programs supports four types of boundary conditions in time [1,2]. Periodic boundary conditions (anti-periodic for the quark fields) are selected by including the section

```
[Boundary conditions]
type 3
theta 0.0 0.0 0.0
```

in the parameter file. The integer value on the line with tag `type` specifies the type of boundary condition (0: open, 1: Schrödinger functional (SF), 2: open-SF, 3: periodic), while the three values on the line with tag `theta` are the angles $\theta_1, \theta_2, \theta_3$ specifying the phase-periodic boundary conditions to be used for the quark fields in the space directions [2].

In the case of open boundary conditions [3], the section

```
[Boundary conditions]
type 0
cG 1.10
cF 0.95
theta 0.0 0.0 0.0
```

has two further lines, where the values of the boundary improvement coefficients `cG` and `cF` are specified. These lines are also required when SF boundary conditions are chosen. In addition, the parameter section

```
[Boundary conditions]
type 1
phi 1.57 0.0
phi' 2.75 -1.89
cG 1.10
```

```
cF 0.95
theta 0.5 1.0 -0.5
```

must include the angles `phi` and `phi'` that define the boundary values of the gauge field at time 0 and T [1] (the two values on these lines are the values of the angles ϕ_1, ϕ_2 at time 0 and ϕ'_1, ϕ'_2 at time T, respectively).

If open-SF boundary conditions are chosen, the boundary values of the gauge field at time T and thus the angles ϕ'_1, ϕ'_2 must be specified as before. The parameter section then looks like

```
[Boundary conditions]
type 2
phi' 2.75 -1.89
cG 1.10
cG' 1.08
cF 0.95
cF' 0.93
theta 0.5 1.0 -0.5
```

Note that the values of the boundary improvement coefficients at time 0 (`cG`,`cF`) and time T (`cG'`,`cF'`) need not be the same.

3.2 Gauge action

Currently the supported gauge actions include the Wilson plaquette action, the tree-level improved Symanzik action and, more generally, any linear combination of these [1]. Since the parameters of the gauge action are already specified in the `Lattice parameters` and `Boundary conditions` sections, it suffices to include the lines

```
[Action 0]
action ACG
```

in the parameter file to instruct the program that the total action of the theory includes the gauge action. This section is the first example of an action section. It has a single line containing the tag `ACG`, which specifies that the action number 0 is the gauge action.

3.3 Twisted-mass pseudo-fermion actions

Pseudo-fermion actions

$$S_{\rm pf} = (\phi, (D^\dagger D + \mu_0^2)^{-1}\phi) \tag{3.1}$$

with a single twisted-mass parameter μ_0 are described by an action section

```
[Action 1]
action ACF_TM1
ipf 0
im0 0
imu 0
isp 3
```

The number in the headline is an index that serves as a tag for the different actions. It can be chosen arbitrarily in the range `0,1,...,31`, but must be unambiguous, i.e. there may be at most one action section in the parameter file with a given index.

The action (3.1) depends on the pseudo-fermion field ϕ, the bare quark mass m_0 in the Dirac operator and the twisted mass μ_0. Pseudo-fermion fields are distinguished by an index `ipf=0,1,...,npf-1`, where `npf` is the total number of these fields. The bare sea-quark masses m_0 are referred to by an index `im0=0,1,...` that labels the corresponding sea-quark hopping parameters $\kappa = (8 + 2m_0)^{-1}$ (subsect. 3.1), while twisted masses are specified by giving their index `imu` in an array of values defined elsewhere (see sect. 4). The last parameter in the section, `isp`, is the index of the solver for the Dirac equation, which is to be used by the program that computes the action (3.1) (see sect. 5 for the list of the available solvers).

Hasenbusch pseudo-fermion actions

$$S_{\rm pf} = (\phi, (D^\dagger D + \mu_1^2)(D^\dagger D + \mu_0^2)^{-1}\phi) \tag{3.2}$$

with two twisted-mass parameters μ_0 and μ_1 are described by an action section

```
[Action 2]
action ACF_TM2
ipf 1
im0 0
imu 0 1
isp 1 0
```

As explained in the notes [4], these actions arise when the light-quark determinant is split into several factors. The two masses correspond to the two entries on the lines with tag `imu` and `isp`. Note that one needs to specify two solvers, the first for the Dirac equation with twisted mass μ_0 and the second for the equation with twisted mass μ_1 (which must be solved when ϕ is generated at the beginning of the molecular-dynamics trajectories).

3.4 Even-odd preconditioned pseudo-fermion actions

When even-odd preconditioning is used, the pseudo-fermion action (3.1) gets replaced by

$$S_{\mathrm{pf}} = (\phi, (\hat{D}^\dagger \hat{D} + \mu_0^2)^{-1} \phi) - 2 \ln \det D_{\mathrm{oo}}, \tag{3.3}$$

where $\hat{D}$ is the even-odd preconditioned Dirac operator and D_{oo} the odd-odd part of the Dirac operator [2]. The factor $\det D_{\mathrm{oo}}$ is referred to as the "small determinant" and it is understood that the pseudo-fermion field ϕ vanishes on the odd sites of the lattice.

Apart from the action line, the parameter sections

```
[Action 3]
action ACF_TM1_EO_SDET
ipf 2
im0 0
imu 0
isp 3
```

describing the action (3.3) look the same as the ones describing the actions without even-odd preconditioning. The same comment applies to the sections

```
[Action 4]
action ACF_TM2_EO
ipf 3
im0 0
imu 0 1
isp 1 0
```

that the describe the even-odd preconditioned version

$$S_{\mathrm{pf}} = (\phi, (\hat{D}^\dagger \hat{D} + \mu_1^2)(\hat{D}^\dagger \hat{D} + \mu_0^2)^{-1} \phi) \tag{3.4}$$

of the Hasenbusch pseudo-fermion action (there is no small-determinant contribution in this case).

3.5 Rational function actions

The RHMC algorithm for the strange and the charm quark is based on the Zolotarev rational approximation of the operator $(\hat{D}^\dagger \hat{D})^{-1/2}$ (see ref. [5] for detailed explanations). Zolotarev rational functions are defined by a section

```
[Rational 0]
degree 10
range 0.025 6.02
```

that specifies the degree (number of poles) of the function and the approximation range on the axis of eigenvalues of $(\hat{D}^\dagger\hat{D})^{1/2}$. The range should be sufficiently large to include, with probability practically equal to 1, the whole spectrum of the operator.

The poles and zeros of the Zolotarev rational functions are ordered such that the larger ones come first. For reasons of stability and performance, the rational function should be split into a product of factors, each including a range of poles and zeros [5]. The pseudo-fermion action associated with such a factor is described by a section

```
[Action 5]
action ACF_RAT
ipf 4
im0 1
irat 0 7 9
isp 1 2
```

The parameters `ipf`, `im0` and `isp` in this section have the same meaning as in the case of the Hasenbusch twisted-mass pseudo-fermion actions, while `irat` specifies the index of the rational function and the range of poles included in the factor. In particular, the first solver is used when the action is calculated and the second when the pseudo-fermion field is regenerated. Poles are counted from zero and pole ranges are inclusive, i.e. `irat 0 7 9` selects the poles number `7`, `8` and `9` from the rational function with index `0`.

Since even-odd preconditioning is used for the strange and the charm quark, the associated quark determinants contain the small determinant $\det D_{\mathrm{oo}}$. It is convenient to combine this factor with the rational factor that contains the largest poles. The corresponding section is

```
[Action 6]
action ACF_RAT_SDET
ipf 5
im0 1
irat 0 0 6
isp 4 4
```

For a given rational function and mass index `im0`, the action sections must be such that all poles and the small determinant are included once. In the example discussed

here, this is achieved with two factors, but one is free to split the rational functions in more factors. Note that for each factor a different pseudo-fermion field must be used.

4. Molecular-dynamics parameters

The simulation algorithms implemented in the openQCD package involve a numerical integration of the molecular-dynamics equations that derive from the chosen action. Apart from the action, this requires the integration scheme (the "integrator") to be specified as well as the solvers used for the computation of the various pseudo-fermion forces.

4.1 HMC parameters

Some basic parameters of the algorithm are collected in the section

```
[HMC parameters]
actions 0 1 2
npf 6
mu 0.015 0.2 1.0
nlv 2
tau 1.0
```

The numbers on the `action` line are the indices of the actions that are to be included in the molecular-dynamics Hamilton function. If some of these actions depend on twisted mass parameters, their values must be listed on the line with tag `mu`. As already mentioned, the action sections refer to these values by quoting their position `imu` (counting from 0) in the array. The parameter `npf` specifies the total number of pseudo-fermion fields that must be allocated.

The last two parameters, `nlv` and `tau`, define the number of integrator levels and the molecular-dynamics trajectory length (see ref. [4] for the normalization of the latter; the integration scheme is discussed in subsect. 4.4).

4.2 SMD parameters

The basic parameters of the SMD algorithm are similar to those of the HMC algorithm. In particular, the meaning of the first four lines in the section

```
[SMD parameters]
actions 0 1 2
npf 2
mu 0.01 1.0
nlv 3
gamma 0.3
eps 0.2
iacc 1
```

is unchanged. Then follow the "friction parameter" `gamma`, the simulation step size `eps` (which coincides with the molecular-dynamics trajectory length) and finally the parameter `iacc`, which specifies whether the algorithm should include the accept-reject step (`iacc=1`) or not (`iacc=0`). At the beginning of each update cycle, the momentum and pseudo-fermion fields are all refreshed using the same value of the friction parameter (i.e. as described in sect. 4.3.1 of ref. [6]).

4.3 Forces

The forces that derive from the actions `ACG,...,ACF_RAT_SDET` discussed in sect. 3 are distinguished by the symbols `FRG,...,FRF_RAT_SDET`. The corresponding sections are

```
[Force 0]
force FRG

[Force 1]
force FRF_TM1
isp 6
ncr 4

[Force 2]
force FRF_TM2
isp 7
ncr 0

[Force 3]
force FRF_TM1_EO_SDET
isp 6
ncr 4
```

```
[Force 4]
force FRF_TM2_EO
isp 7
ncr 0

[Force 5]
force FRF_RAT
isp 8

[Force 6]
force FRF_RAT_SDET
isp 9
```

In each case, the index in the headline must match the index of the associated action. The force tags (such as `FRF_TM2_EO`) could be inferred from the corresponding action sections, but are required in order to improve the readability of the input parameter files.

Most parameters of the forces are inherited from those of the associated actions. The solver parameter sets selected by the indices `isp` may however be different. In the case of the forces `FRF_TM2` and `FRF_TM2_EO`, the specified solver is used for the solution of the Dirac equation with twisted mass μ_0. No solver is needed here for the second twisted mass μ_1.

The parameter `ncr` controls the chronological propagation of the solutions of the Dirac equation along the molecular-dynamics trajectories. If `ncr` is set to a positive value, the simulation program attempts to reduce the number of solver iterations required for the computation of the specified force by extrapolating the previous `ncr` solutions in molecular-dynamics time. The feature is switched off if `ncr` is set to zero.

4.4 Integration scheme

The numerical integration of the molecular-dynamics equations makes use of a hierarchical integrator that can be specified on the input parameter file. Currently three elementary integration schemes are supported, the leapfrog, the 2nd order Omelyan–Mryglod–Folk (OMF) and a 4th order OMF integrator. They are distinguished by the symbols `LPFR`, `OMF2` and `OMF4`.

Hierarchical integrators have several levels with increasing integration step sizes. They are described by the total integration time `tau`, the number `nlv` of levels, the elementary integrators used at each level and the forces integrated there. The values of these parameters are copied from the HMC and SMD parameter sets (see

subsects. 4.1,4.2). Each level is then described by a section like

```
[Level 2]
integrator LPFR
nstep 12
forces 2 4 5
```

In this example, the simulation program is instructed to use the leapfrog integrator at the third level (levels are counted from 0). The elementary leapfrog integration step is to be applied `12` times and the forces integrated at this level are the ones with index `2,4` and `5`. If the level is the top level, the integration step size for these forces is thus `tau/12`.

An example of a complete integrator description is provided by the following three sections:

```
[Level 0]
integrator OMF4
nstep 1
forces 0

[Level 1]
integrator OMF4
nstep 1
forces 1 2 3

[Level 2]
integrator OMF2
lambda 0.16667
nstep 6
forces 4
```

There are three levels in this case, with average step sizes equal to `tau/300`, `tau/60` and `tau/12` at level `0`, `1` and `2`, respectively (note that the `OMF2` and `OMF4` integrators update the gauge field 2 and 5 times per application). The `OMF2` integrator depends on a parameter `lambda`, whose value must be given at each level where this integrator is used. See `modules/update/README.mdint` for further explanations.

5. Solver parameters

All solvers referred to in the action and force sections must be described in the input parameter file. The solvers are labeled by an index `isp` in the range `0,1,...,63` in much the same way as the actions and the forces. There is no one-to-one correspondence between solver programs and solver sections in the parameter file. One may, for example, have two sections

```
[Solver 3]
solver CGNE
nmx 256
istop 1
res 1.0e-10

[Solver 4]
solver CGNE
nmx 256
istop 1
res 1.0e-8
```

where the only difference is the value of the desired residue `res`. By setting `isp=3` or `isp=4` in an action or force section, the numerical accuracy of the action and force computations can then be individually controlled.

5.1 Stopping criterion

The iterative algorithms used to solve the Dirac equation

$$D\psi(x) = \eta(x) \tag{5.1}$$

are stopped if the residue of the current approximate solution satisfies

$$\|\eta - D\psi\| \leq \texttt{res} \times \|\eta\|, \tag{5.2}$$

where `res` is the tolerance specified in the solver section. The norm $\|\cdot\|$ may be the usual L_2 norm $\|\cdot\|_2$ or the uniform norm

$$\|\psi\|_\infty = \max_x \|\psi(x)\|_2. \tag{5.3}$$

Each solver parameter set includes the parameter `istop`, which selects the norm to be $\|\cdot\|_2$ if `istop=0` and $\|\cdot\|_\infty$ if `istop=1`.

5.2 Conjugate gradient solvers

There are two conjugate gradient solvers, the ordinary CG algorithm for the normal Dirac equations,

$$(D^\dagger D + \mu^2)\psi = \eta \quad \text{and} \quad (\hat{D}^\dagger \hat{D} + \mu^2)\psi = \eta, \tag{5.4}$$

and the mult-shift CG algorithm for the even-odd preconditioned simultaneous equations [5,7]

$$(\hat{D}^\dagger \hat{D} + \mu_k^2)\psi_k = \eta \qquad k = 0, 1, \ldots, n-1. \tag{5.5}$$

Examples of parameter sections describing solvers based on the former have already appeared in this note. The parameter sections

```
[Solver 3]
solver MSCG
nmx 256
istop 0
res 1.0e-10
```

for the multi-shift solver look practically the same. Note that this solver can only be used for rational function actions and forces.

5.3 SAP preconditioned solvers

There are currently two solvers that make use of the Schwarz Alternating Procedure (SAP) as preconditioner for the GCR algorithm. When these solvers are used, the block size `bs` of the SAP block grid needs to be specified in a separate section

```
[SAP]
bs 4 6 6 4
```

The ordinary SAP preconditioned GCR algorithm is then described by a parameter section

```
[Solver 7]
solver SAP_GCR
nkv 16
isolv 1
nmr 4
ncy 5
nmx 24
```

```
istop 1
res 1.0e-8
```

where `nkv` is the maximal number of Krylov vectors that may be generated before the GCR algorithm is restarted, while `nmx` is the maximal total number of generated Krylov vectors and `res` the desired relative residue of the calculated solutions. All other parameters in the section describe the particular SAP preconditioner to be used, namely `ncy` and `nmr` specify the number of SAP cycles and block solver iterations to be applied and the flag `isolv` indicates whether the block solver is even-odd preconditioned (`isolv=1`) or not (`isolv!=1`).

The other solver that makes use of the SAP is the deflated SAP preconditioned GCR solver [8]. This solver is described by a section

```
[Solver 8]
solver DFL_SAP_GCR
nkv 16
isolv 1
nmr 4
ncy 5
nmx 24
istop 1
res 1.0e-8
```

that coincides with the section for the ordinary SAP preconditioned solver except for the solver symbol.

5.4 Deflation parameters

If the `DFL_SAP_GCR` solver is used, the parameters related to the deflation subspace must be specified in a few further sections. The current version of openQCD supports one-level deflation, as in the previous versions, with a slightly improved solver for the little Dirac equation.

The number `Ns` of global deflation modes and the deflation block sizes `bs` are set by including the section

```
[Deflation subspace]
Ns 28
bs 4 4 4 4
```

For the generation of the deflation subspace, an inverse iteration algorithm is applied to a set of `Ns` random quark fields using an approximate inverse of the Dirac operator $D+i\mu\gamma_5 1_{\rm e}$, where $1_{\rm e}$ is 1 on the even sites of the lattice and 0 elsewhere. The hopping

parameter kappa on which the operator depends and the twisted mass mu must be specified in the section

```
[Deflation subspace generation]
kappa 0.1392888
mu 0.0
ninv 11
nmr 4
ncy 4
```

along with the number ninv of inverse iteration steps to be applied and the numbers ncy and nmr of SAP cycles and block solver iterations to be used in this process.

In each iteration of the DFL_SAP_GCR solver, the deflation projection requires the little Dirac equation to be solved with low precision. The solution is obtained using a preconditioned FGCR solver with a single-sweep (i.e. non-restarted) GCR algorithm as preconditioner. This algorithm has five parameters that must be specified in a parameter section of the form

```
[Deflation projection]
nkv 24
nmx 72
res 0.01
nmx_gcr 16
res_gcr 0.01
```

Here nkv, nmx and res are, respectively, the number of Krylov vectors generated by the FGCR algorithm before it is restarted, the maximal total number of vectors that may be generated and the desired relative residue of the calculated solution. In each step of the FGCR algorithm, the GCR preconditioner may generate at most nmx_gcr Krylov vectors and the iteration is stopped before this if the relative residue of the current approximate solution of the little Dirac equation reaches a value less than or equal to res_gcr. Setting the latter to res is, in the present context, often a good choice.

Along the molecular-dynamics trajectories, the deflation subspace looses its efficiency and must be refreshed from time to time. This feature is controlled by the parameter section

```
[Deflation update scheme]
dtau 0.091
nsm 1
```

where `dtau` sets the interval in molecular-dynamics time after which the subspace is updated and `nsm` the number of (inverse iteration) smoothing steps to be applied in this process. The subspace updates can be turned off by setting `nsm=0`. Clearly, the section is not needed in measurement programs, where the deflated solver may be used for quark propagator calculations, for example.

6. Reweighting factors

In general, the ensembles of field configurations generated in openQCD simulations must be reweighted. The rational approximation used for the strange and the charm quark determinants, for example, requires reweighting in order to correct for the approximation error. In the case of the light quark doublet, reweighting is required if a twisted mass term was added as infrared regulator [9]. For a given ensemble of field configurations, the program `ms1` computes stochastic estimates of the reweighting factors specified in a parameter file.

The openQCD package currently does not support reweighting in the bare quark masses [11,12], but a module providing this functionality can be downloaded from `http://www-ai.math.uniwuppertal.de/~leder/mrw/`.

6.1 Light quark reweighting factors

In ref. [9], two kinds of twisted-mass regularizations of the light quark determinant were proposed which amount to replacing

$$\det\{D^\dagger D\} \to \det\{D^\dagger D + \mu^2\} \tag{6.1}$$

and

$$\det\{D^\dagger D\} \to \det\{(D^\dagger D + \mu^2)^2 (D^\dagger D + 2\mu^2)^{-1}\} \tag{6.2}$$

respectively. The twisted mass parameter $\mu > 0$ provides the desired infrared regularization and is usually set to a value on the order of light quark mass.

The associated reweighting factors,

$$W_1 = \det\{D^\dagger D (D^\dagger D + \mu^2)^{-1}\}, \tag{6.3}$$

$$W_2 = \det\{D^\dagger D (D^\dagger D + 2\mu^2)(D^\dagger D + \mu^2)^{-2}\}, \tag{6.4}$$

can be estimated stochastically using Gaussian random quark fields. With a reasonably small number of random fields, a straightforward estimation may however turn out to be rather noisy, especially so on lattices where the lowest eigenvalues of the lattice Dirac operator fluctuate to values much smaller than the regularization mass μ. In these cases, the reweighting factors are better broken up into several factors according to [10–12]

$$W_1 = \prod_{l=1}^{n} \det\{R_1(\mu_{l-1}, \mu_l)\}, \qquad 0 = \mu_0 < \mu_1 < \ldots < \mu_n = \mu, \tag{6.5}$$

$$W_2 = \prod_{l=1}^{n} \det\{R_2(\mu_{l-1}, \mu_l)\}, \tag{6.6}$$

where

$$R_1(\mu_i, \mu_j) = \frac{D^\dagger D + \mu_i^2}{D^\dagger D + \mu_j^2}, \tag{6.7}$$

$$R_2(\mu_i, \mu_j) = \frac{(D^\dagger D + \mu_i^2)^2 (D^\dagger D + 2\mu_j^2)}{(D^\dagger D + 2\mu_i^2)(D^\dagger D + \mu_j^2)^2}. \tag{6.8}$$

The factors $\det\{R_a(\mu_{l-1}, \mu_l)\}$ are then estimated stochastically by choosing N random quark fields $\eta_1, \ldots, \eta_N$ with normal distribution and by calculating the estimators

$$\frac{1}{N} \sum_{k=1}^{N} \exp\{-(\eta_k, [R_a(\mu_l, \mu_{l-1}) - 1]\eta_k)\}. \tag{6.9}$$

It is understood here that a fresh set of N random fields is generated for each factor. Note that the eigenvalues of

$$R_1(\mu_i, \mu_j) - 1 = \frac{\mu_i^2 - \mu_j^2}{D^\dagger D + \mu_j^2}, \tag{6.10}$$

$$R_2(\mu_i, \mu_j) - 1 = \frac{\mu_i^2 - \mu_j^2}{2\mu_i^2 - \mu_j^2} \left\{ \frac{\mu_j^2(\mu_i^2 - \mu_j^2)}{(D^\dagger D + \mu_j^2)^2} + \frac{2\mu_i^4}{(D^\dagger D + 2\mu_i^2)(D^\dagger D + \mu_j^2)} \right\}, \tag{6.11}$$

are all strictly positive if $\mu_i > \mu_j \geq 0$.

In the parameter files, the reweighting factors are described by sections of the form

```
[Reweighting factor 0]
rwfact RWTM1
im0 0
mu 0.0001 0.0005 0.001 0.003
isp 0
nsrc 24
```

As usual the index in the headline serves to distinguish different reweighting factors. The line with tag `rwfact` selects the reweighting factor type (`RWTM1` for W_1 and `RWTM2` for W_2), while the other parameters, `im0`, `mu`, `isp` and `nsrc`, are the index of the bare quark mass m_0, the list $\mu_1, \ldots, \mu_n$ of the twisted masses in the factorizations (6.5),(6.6), the index of the solver for the Dirac equation to be used and the number N of random source fields to be generated for each factor $\det\{R_a(\mu_{l-1}, \mu_l)\}$ and each gauge-field configuration. A different solver can optionally be chosen for each factor by replacing the single solver index on the line with tag `isp` by a list of as many indices as there are twisted masses.

If even-odd preconditioning is used, eqs. (6.1) and (6.2) get replaced by

$$\det\{D^\dagger D\} \to \det\{(D_{\rm oo})^2\}\det\{\hat{D}^\dagger\hat{D} + \mu^2\}, \tag{6.12}$$

$$\det\{D^\dagger D\} \to \det\{(D_{\rm oo})^2\}\det\{(\hat{D}^\dagger\hat{D} + \mu^2)^2(\hat{D}^\dagger\hat{D} + 2\mu^2)^{-1}\}. \tag{6.13}$$

The associated reweighting factors,

$$\hat{W}_1 = \det\{\hat{D}^\dagger\hat{D}(\hat{D}^\dagger\hat{D} + \mu^2)^{-1}\}, \tag{6.14}$$

$$\hat{W}_2 = \det\{\hat{D}^\dagger\hat{D}(\hat{D}^\dagger\hat{D} + 2\mu^2)(\hat{D}^\dagger\hat{D} + \mu^2)^{-2}\}, \tag{6.15}$$

can again be estimated stochastically following the lines above. The corresponding parameter sections in the input files are as before except that the symbols `RWTM1` and `RWTM2` must be replaced by `RWTM1_EO` and `RWTM2_EO`.

6.2 Strange and charm quark reweighting factors

The reweighting factors required for the heavier quarks are given by

$$W_R = \det\{\hat{D}R\}, \tag{6.16}$$

where R is the rational function used to approximate the operator $(\hat{D}^\dagger\hat{D})^{-1/2}$ (see ref. [5] for further details). In the input parameter files, these factors are described by sections like

```
[Reweighting factor 1]
rwfact RWRAT
im0 1
irp 0
np 6 4
isp 1 0
nsrc 1
```

As before, `im0` is the index of the bare mass m_0 of the quark considered and `nsrc` the number of random source fields to be used for the stochastic estimation of W_R. The other parameters select the rational function R and determine exactly how the reweighting factor is to be computed.

Rational functions are defined by separate parameter sections (see subsect. 3.5). The parameter `irp` must be set to the index of the section that describes the function R. As already mentioned, the poles of R are ordered such that the larger ones come first. The program that computes the reweighting factor expands R into partial fractions and logically divides the pole terms into groups R_k of terms,

$$R = A\{1 + R_0 + R_1 \ldots + R_{n-1}\}, \tag{6.17}$$

where R_0 contains the first few poles, R_1 the next few poles, and so on. The figures on the line with tag `np` specify the numbers of poles in these groups. In the example above, there are two parts, R_0 and R_1, where R_0 includes the poles $0, 1, \ldots, 5$ and R_1 the poles $6, \ldots, 9$. Any sequence of positive numbers can appear on the line with tag `np` subject to the constraint that their sum coincides with the number of poles of R.

For each part R_k, the solver for the (twisted mass) Dirac equation can be chosen separately. The indices of the corresponding solver parameter sets must be given on the line with tag `isp`, where it is understood that the first solver is to be used when the operator R_0 is applied to the source field, the second when R_1 is applied, and so on. Currently the supported solvers are `MSCG`, `SAP_GCR` and `DFL_SAP_GCR` (see sect. 5).

7. Configuration I/O

The openQCD simulation programs can store field configurations in various formats. Moreover, parallel I/O is possible, where several MPI processes write parts of the configurations to disk at the same time. The supported storage formats are *exported*, *block-exported* and *local*. See `main/README.io` for a description of how exactly the field configurations are written to disk in each case.

7.1 Configuration output

To write configurations in exported format, it suffices to provide the configuration directory name as in the section

```
[Configurations]
types e
cnfg_dir /data/qcd1/cnfg
```

The first line here specifies the storage type (`e`, `b` or `l` for exported, block-exported or local), while the second provides the configuration directory path. In this example, the path is an absolute one, but it can also be given relative to the directory from which the simulation program is launched.

In the case of block-exported storage, the lattice is divided into rectangular blocks of points (the "storage blocks") and the fields residing on each block are written to disk to separate files. These may be stored in several subdirectories of the configuration directory and may be written in parallel. A parameter section describing this form of configuration output is

```
[Configurations]
types b
block_dir /data/qcd1/blk
bs 32 24 24 24
nio_nodes 8
nio_streams 4
```

where the line with tag `bs` specifies the storage block size, `nio_nodes` is the number of subdirectories and `nio_streams` the number of parallel output streams. For local storage, the line with tag `bs` is not required, the storage identifier `b` gets replaced by `l` and the tag `block_dir` by `local_dir`.

In certain situations it may be desirable to write out the field configurations in more than one storage format. The parameter section

```
[Configurations]
types eb
cnfg_dir /data/qcd1/cnfg
block_dir /data/qcd1/blk
bs 32 24 24 24
nio_nodes 8
nio_streams 4
```

for example, implies that the configurations are stored in both `e` and `b` format in the specified directories.

The titles of configuration I/O sections may vary depending on the task performed by the program. Apart from the tags listed above, these section may contain further tags, whose meaning is described in the `README` that comes with the program.

7.2 Configuration input

To read configurations in exported format, a parameter section of the form

```
[Initial configuration]
type e
cnfg_dir /data/qcd1/cnfg
```

suffices. If the configurations are available in block-exported format, the appropriate parameter section is

```
[Initial configuration]
type b
block_dir /data/qcd1/blk
nio_nodes 8
nio_streams 4
```

Clearly, configurations to be read come with a definite storage format, which is why the tag on the first line is `type` rather than `types`. The storage block size does not appear in this case, since the information is included in the configuration files, but the number `nio_nodes` of top-level subdirectories of the configuration directory may not be omitted and must coincide with the one chosen when the configurations were written to disk (the last parameter, `nio_streams`, may vary).

As before, the parameter section looks the same for local storage, with `b` replaced by `l` and the tag `block_dir` by `local_dir`.

7.3 Constraints on the storage block size and the I/O parameters

When block-exported storage is used, the following conditions must be satisfied:

- The local lattices divide the storage blocks and the latter divide the global lattice.
- Both `nio_nodes` and `nio_streams` divide the number of storage blocks.

The same constraints apply for local storage, the storage blocks being equal to the local lattices in this case.

References

[1] *Gauge actions in openQCD simulations*, `doc/gauge_action.pdf`
[2] *Implementation of the lattice Dirac operator*, `doc/dirac.pdf`
[3] M. Lüscher, Stefan Schaefer, *Lattice QCD without topology barriers*, JHEP 1107 (2011) 036
[4] *Molecular-dynamics quark forces*, `doc/forces.pdf`
[5] *Charm and strange quark in openQCD simulations*, `doc/rhmc.pdf`
[6] M. Lüscher, *Stochastic locality and master-field simulations of very large lattices*, EPJ Web Conf. 175 (2018) 01002
[7] *Multi-shift conjugate gradient algorithm*, `doc/mscg.pdf`
[8] *Deflated solver for the Dirac equation in openQCD*, `doc/dfl.pdf`
[9] M. Lüscher, F. Palombi, *Fluctuations and reweighting of the quark determinant on large lattices*, PoS (LATTICE2008) 049
[10] M. Hasenbusch, *Speeding up the Hybrid Monte Carlo algorithm for dynamical fermions*, Phys. Lett. B519 (2001) 177
[11] A. Hasenfratz, R. Hoffmann, S. Schaefer, *Reweighting towards the chiral limit*, Phys. Rev. D78 (2008) 014515
[12] J. Finkenrath, F. Knechtli, B. Leder, *One flavor mass reweighting in lattice QCD*, Nucl. Phys. B877 (2013) 441