Parameters of the openQ*D main programs

adapted by: Agostino Patella, RC* collaboration March 2018

heavily based on: M. Lüscher, *Parameters of the openQCD main programs*, June 2016, `doc/openQCD-1.6/rhmc.pdf`

Contents

1 Introduction . . . 3

2 Preliminaries . . . 3

3 Actions . . . 4
- 3.1 Field content . . . 4
- 3.2 Boundary conditions . . . 5
- 3.3 Gauge action parameters . . . 6
- 3.4 Quark parameters . . . 9
- 3.5 Gauge actions . . . 11
- 3.6 Twisted-mass pseudo-fermion actions . . . 11
- 3.7 Even-odd preconditioned pseudo-fermion actions . . . 12
- 3.8 Rational function actions . . . 13

4 Molecular-dynamics parameters . . . 16
- 4.1 HMC parameters . . . 16
- 4.2 Forces . . . 17
- 4.3 Integration scheme . . . 18

5 Solver parameters . . . 20
- 5.1 Conjugate gradient solvers . . . 20
- 5.2 SAP preconditioned solvers . . . 21
- 5.3 Deflation parameters . . . 22

6 Reweighting factors . . . 24
- 6.1 Twisted-mass reweighting factors for doublets . . . 25
- 6.2 Reweighting factors for RHMC . . . 25

1 Introduction

The `openQ*D` 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.

Only the parameters used by the programs that generate configurations (`iso1`, `qcd1`, `ym1`, `mxw1`) are discussed in these notes. For explanations of the parameters used by the other programs (`ms*`), the reader is referred to the corresponding `README` files.

2 Preliminaries

The input parameter files are divided into sections such as

```
[Solver 3]
solver CGNE
nmx 256
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, while `nmx` specifies the maximal number of CG iterations that may be performed and `res` the desired relative residue of the calculated solutions.

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
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 database and can, from there, be retrieved by the subprograms that depend on some of the parameter values. The database 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 field content, 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 Field content

The `openQ*D` program can simulate the SU(3)×U(1), SU(3) or U(1) gauge group, with or without dynamical fermions. The relevant main programs are listed below with their respective field content.

Program	Gauge group	Fermions
`iso1`	SU(3)×U(1)	yes
`qcd1`	SU(3)	yes
`ym1`	SU(3)	no
`mxw1`	U(1)	no

3.2 Boundary conditions

The boundary conditions for the gauge fields are defined in a dedicated section. For instance

```
[Boundary conditions]
type    3               # (equivalently) type periodic
cstar   1
```

selects periodic boundary conditions in time and C* boundary conditions in one space direction (and periodic in the others).

The value of the parameter `cstar` must be a number from 0 to 3, equal to the number of spatial directions with C* boundary conditions [1]. The gauge field is periodic in the remaining spatial direction.

Boundary conditions in time [2] are chosen by setting the value of the parameter `type`. If a SF boundary exists, then the phases determining the value of the fields at the boundary need to be provided as well. In the case of open boundary conditions and no C* boundary conditions, the section should be

```
[Boundary conditions]
type    0               # (equivalently) type open
cstar   0
```

If SF boundary conditions are chosen, the parameter section

```
[Boundary conditions]
type    1               # (equivalently) type SF
cstar   0
su3phi  1.57 0.0        # relevant only if the SU(3) field is used
su3phi' 2.75 -1.89      # relevant only if the SU(3) field is used
u1phi   1.2             # relevant only if the U(1) field is used
u1phi'  -0.3            # relevant only if the U(1) field is used
```

must include:

- the angles $\phi_{\mathrm{SU}(3),1}$, $\phi_{\mathrm{SU}(3),1}$ defining the boundary values of the SU(3) gauge field at time 0 as the two values of the parameter `su3phi`; this line can be skipped if the SU(3) field is not used;
- the angles $\phi'_{\mathrm{SU}(3),1}$, $\phi'_{\mathrm{SU}(3),1}$ defining the boundary values of the SU(3) gauge field at time T as the two values of the parameter `su3phi'`; this line can be skipped if the SU(3) field is not used;
- the angle $\phi_{\mathrm{U}(1)}$ defining the boundary values of the U(1) gauge field at time 0 as value of the parameter `u1phi`; this line can be skipped if the U(1) field is not used;
- the angle $\phi'_{\mathrm{U}(1)}$ defining the boundary values of the U(1) gauge field at time T as value of the parameter `u1phi'`; this line can be skipped if the U(1) field is not used.

If open-SF boundary conditions are chosen, the boundary values of the gauge fields at time T must be specified as before. The parameter section then looks like

```
[Boundary conditions]
type    2               # (equivalently) type open-SF
cstar   0
su3phi' 2.75 -1.89      # relevant only if the SU(3) field is used
u1phi'  -0.3            # relevant only if the U(1) field is used
```

3.3 Gauge action parameters

If the SU(3) field is used, the parameters of the SU(3) gauge action have to be defined in the parameter section

```
[SU(3) action]
beta    6.0             # Inverse SU(3) gauge coupling
c0      1.6667          # Coefficient of the plaquette term
                        # in the SU(3) gauge action
cG      1.10            # SU(3) gauge action improvement coefficient
```

```
                        # at time 0
cG'     1.05            # SU(3) gauge action improvement coefficient
                        # at time NPROC0*L0
SFtype  0               # Type of SF boundary (this affects the
                        # LW gauge action):
                        #   0|orbifold|openQCD-1.4
                        #   1|AFW-typeB|openQCD-1.2
```

where `beta` is the parameter $\beta = 6/g_0^2$, `c0` is the c_0 parameter appearing in the Lüscher-Weisz action, `cG` and `cG'` are the boundary-improvement coefficients $c_{\rm G}$ and $c'_{\rm G}$ respectively, `SFtype` determines the particular form of the double-plaquettes at the SF boundary if the Lüscher-Weisz action is used. More details on these parameters can be found in [2].

Which paramters are requested depend on the choice of boundary conditions in time: with periodic boundary conditions

```
[SU(3) action]
beta    6.0
c0      1.6667
```

with open boundary conditions

```
[SU(3) action]
beta    6.0
c0      1.6667
cG      1.10
```

with SF boundary conditions and Lüscher-Weisz action

```
[SU(3) action]
beta    6.0
c0      1.6667
cG      1.10
SFtype  orbifold
```

with open-SF boundary conditions and Lüscher-Weisz action

```
[SU(3) action]
beta    6.0
c0      1.6667
cG      1.10
cG'     1.05
SFtype  orbifold
```

If the U(1) field is used, the parameters of the U(1) gauge action have to be defined in the parameter section

```
[U(1) action]
type    0               # Type of U(1) actions:
                        #   0|compact
                        #   1|non-compact
alpha   0.05            # Fine-structure constant
invqel  6.0             # Inverse of the elementary charge used in
                        # in the simulation. Quark electric charges
                        # must be even integer multiples of the
                        # elementary charge.
c0      1.6667          # Coefficient of the plaquette term
                        # in the compact U(1) gauge action
cG      1.10            # U(1) gauge action improvement coefficient
                        # at time 0
cG'     1.05            # U(1) gauge action improvement coefficient
                        # at time NPROC0*L0
SFtype  0               # Type of SF boundary (this affects the
                        # LW gauge action):
                        #   0|orbifold|openQCD-1.4
                        #   1|AFW-typeB|openQCD-1.2
```

In the current version of the code, only the compact U(1) action is available (parameter `type`). The value of `alpha` determines the fine-structure constant $\alpha = e_0^2/(4\pi)$, the value of `invqel` determines the parameter $q_{\rm el}^{-1}$ which determines the normalization of

the U(1) gauge action, and consequently the quantum of electric charge. `c0` is the c_0 parameter appearing in the Lüscher-Weisz action, `cG` and `cG'` are the boundary-improvement coefficients $c_{\rm G}$ and $c'_{\rm G}$ respectively, `SFtype` determines the particular form of the double-plaquettes at the SF boundary if the Lüscher-Weisz action is used. More details on these parameters can be found in [2]. As for the SU(3) gauge action, some parameters can be omitted depending on the boundary conditions in time.

3.4 Quark parameters

In the terminology of the `openQ*D` code, a *quark flavour* is identified by all adjustable parameters that define the Dirac operator. For instance, in a simulation in the isospin symmetric limit, the up and down quark count as a single *quark flavour*. The number of quark flavours in this sense is defined in the parameter section

```
[Quark action]
nfl          3
```

Each quark flavour is identified by an integer in the range $0, \ldots,$ `nfl` -1. The parameters defining the flavour 2 are defined in the parameter section

```
[Flavour 2]
```

In case of periodic boundary conditions in time and space for the gauge field, the quarks satisfy anti-periodic boundary conditions in time and phase-periodic boundary conditions in space. The following parameter section has to be included in case of a QCD simulation

```
[Flavour 2]
kappa        0.1300
su3csw       1.234
theta        0.5 1.0 -0.5
```

where `kappa` is the hopping parameter, `su3csw` is the SW coefficient $c_{\rm sw}^{\rm SU(3)}$, and the three values in `theta` define the phase-periodic boundary conditions in the three special directions. More details about these parameters are given in [3]. If C* boundary conditions are chosen in any of the spatial directions, the `theta` parameters should be omitted. For a QCD+QED simulation with (anti-)periodic boundary conditions in time and C* boundary conditions in space, the following parameter section has to be included

```
[Flavour 2]
qhat          -2
kappa         0.1300
su3csw        1.234
u1csw         0.89
```

where `qhat` is the electric charge in units of $q_{\rm el}$ (defined in the gauge action parameters) and `u1csw` is the SW coefficient $c_{\rm sw}^{\rm U(1)}$.

If different boundary conditions in time are chosen, one needs to include also the parameters `cF` and/or `cF'` defining the boundary improvement coefficients $c_{\rm F}$ and/or $c'_{\rm F}$ respectively. For intance, a QCD simulation with SF or open boundary conditions in time and C* boundary conditions in space should include

```
[Flavour 2]
kappa         0.1300
su3csw        1.234
cF            0.95
```

Instead a QCD+QED simulation with open-SF boundary conditions in time and C* boundary conditions in space should include

```
[Flavour 2]
qhat          -2
kappa         0.1300
su3csw        1.234
u1csw         0.89
cF            0.95
cF'           0.90
```

3.5 Gauge actions

Currently the supported gauge actions include the Wilson plaquette action, the tree-level improved Symanzik action and, more generally, any linear combination of these [2], both for the SU(3) and U(1) gauge field. Since the parameters of the gauge actions are already specified in the `Boundary conditions`, `SU(3) action` and `U(1) action` sections, it is enough to include the lines

```
[Action 0]
action ACG_SU3
```

in the parameter file to instruct the program that the total action of the theory includes the SU(3) gauge action. This section is the first example of an action section. It has a single line containing the parameter `action` with value `ACG_SU3`, which specifies that the action number 0 is the SU(3) gauge action. Similarly it is enough to include the lines

```
[Action 1]
action ACG_U1
```

in the parameter file to instruct the program that the total action of the theory includes the U(1) gauge action as action number 1.

3.6 Twisted-mass pseudo-fermion actions

Pseudo-fermion actions

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

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

```
[Action 2]
action       ACF_TM1
ipf          0
ifl          2
imu          1
isp          0
```

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 $0, 1, \ldots, 31$, but must be unambiguous, i.e. there may be at most one action section in the parameter file with a given index. The action (1) depends on the pseudo-fermion field ϕ, the quark flavour index `ifl` and the twisted mass μ_0. Pseudo-fermion fields are distinguished by an index `ipf` $= 0, 1, \ldots,$ `npf` $- 1$, where `npf` is the total number of these fields. The quark flavour index `ifl` is a number in the range $0, 1, \ldots,$ `nfl` $- 1$, and determines that the Dirac-operator parameters to be used for this action are the ones defined in the corresponding `Flavour` section. Twisted masses are specified by giving their index `imu` in an array of values defined elsewhere in the parameter section `HMC parameters` (see subsect. 4.1). 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 (1) (see sect. 5 for the list of the available solvers).

Hasenbusch pseudo-fermion actions

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

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

```
[Action 3]
action ACF_TM2
ipf 1
ifl 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).

3.7 Even-odd preconditioned pseudo-fermion actions

When even-odd preconditioning is used, the pseudo-fermion action (1) gets replaced by

$$S_{\rm pf} = (\phi, (\hat{D}^\dagger \hat{D} + \mu_0^2)^{-1}\phi) - 2\ln\det D_{\rm oo} \ , \tag{3}$$

where $\hat{D}$ is the even-odd preconditioned Dirac operator and $D_{\rm oo}$ the odd-odd part of the Dirac operator [5]. The factor $\det D_{\rm oo}$ 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 section

```
[Action 4]
action ACF_TM1_EO_SDET
ipf 2
ifl 0
imu 0
isp 3
```

describing the action (3) look the same as the ones describing the actions without even-odd preconditioning. The same comment applies to the section

```
[Action 5]
action ACF_TM2_EO
ipf 3
ifl 0
imu 0 1
isp 1 0
```

that describe the even-odd preconditioned version

$$S_{\rm pf} = (\phi, (\hat{D}^\dagger \hat{D} + \mu_1^2)(\hat{D}^\dagger \hat{D} + \mu_0^2)^{-1}\phi) \tag{4}$$

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

3.8 Rational function actions

The RHMC algorithm is based on the rational approximation of the operator $(\hat{D}^\dagger \hat{D} + \hat{\mu}^2)^\alpha$ (see ref. [6] for detailed explanations). If $\alpha = -1/2$, then the code calculates the Zolotarev rational functions internally. Such rational functions are defined by a section

```
[Rational 0]
power   -1 2
degree 10
range 0.025 6.02
mu 0.01
```

that specifies the degree (number of poles) of the function, the approximation range on the axis of eigenvalues of $(\hat{D}^\dagger \hat{D})^{1/2}$, and the twisted mass $\hat{\mu}$. The range should be sufficiently large to include, with probability practically equal to 1, the whole spectrum of the operator. Rational approximations with $\alpha = p/q$ where p and q are integers are defined by a section

```
[Rational 1]
power            -1 4
degree           3
range            2.74000000e-02 6.29000000e+00
mu               1.00000000e-02
delta            1.6178479482115836e-02
A                3.2735518371365751 5998e-01
nu[0]            4.0003111992177595 7238e+00
mu[0]            2.3540358393000744 2859e+00
nu[1]            5.4460916334120523 7963e-01
mu[1]            3.3708094379707498 3337e-01
nu[2]            7.8576261111302808 6596e-02
mu[2]            4.6940529053469119 6913e-02
x[0]             7.5076000000000005 5770e-04
x[1]             1.8853825932533933 5910e-03
x[2]             1.0689614232166900 3483e-02
x[3]             7.1145154837215493 4929e-02
x[4]             4.7234811059653442 0669e-01
x[5]             3.1195332014755985 3837e+00
x[6]             1.6953629173601171 9422e+01
x[7]             3.9564100000000003 3765e+01
```

The two values of `power` are two integers p and q such that $\alpha = p/q$, the two values of `range` are the extrema of the approximation range on the axis of eigenvalues of $(\hat{D}^\dagger \hat{D})^{1/2}$,

the value of `mu` is the twisted mass $\hat{\mu}$, the value of `degree` is the number of poles. All other values specify the rational function. This section should not be filled by hand, instead it should be produced with the `MinMax` code (in the `minmax` directory of the `openQ*D` code) which implements the minmax approximation algorithm in multiple precision. The general rational functions accepted by the `openQ*D` code are described in [6]. A full documentation of the `MinMax` program can be found in its `README` file.

The poles and zeros of the 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 [6]. The pseudo-fermion action associated with such a factor is described by a section

```
[Action 6]
action ACF_RAT
ipf 4
im0 1
irat 0 7 9
isp 1
```

The parameters `ipf`, `ifl` and `isp` in this section have the same meaning as in the case of the twisted-mass pseudo-fermion actions, while `irat` specifies the index of the rational function and the range of poles included in the factor. 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 RHMC, 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 7]
action ACF_RAT_SDET
ipf 5
im0 1
irat 0 0 6
isp 4
```

For a given rational function and flavour index `ifl`, 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 algorithm implemented in the `openQ*D` package involves 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
facc 1
```

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 some of the action sections refer to these values by quoting their position `imu` (counting from 0) in the array. The parameter `facc` specifies whether the Fourier acceleration should be used for the U(1) field. 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. 4.3).

4.2 Forces

The forces that derive from the actions ACG_SU3,...,ACF_RAT_SDET discussed in sect. 3 are distinguished by the symbols FRG_SU3,...,FRF_RAT_SDET. The corresponding sections are

```
[Force 0]
force FRG_SU3

[Force 1]
force FRG_U1

[Force 2]
force FRF_TM1
isp 6
ncr 4

[Force 3]
force FRF_TM2
isp 7
ncr 0

[Force 4]
force FRF_TM1_EO_SDET
isp 6
ncr 4

[Force 5]
force FRF_TM2_EO
isp 7
ncr 0

[Force 6]
force FRF_RAT
isp 8
```

```
[Force 7]
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.

4.3 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 parameters tau and nlv are part of the HMC parameter set (see subsect. 4.1). 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,...,31 in much the same way as flavours, actions and 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
res 1.0e-10

[Solver 4]
solver CGNE
nmx 256
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 Conjugate gradient solvers

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

$$(D^\dagger D + \mu^2)\psi = \eta \quad \text{and} \quad (\hat{D}^\dagger \hat{D} + \mu^2)\psi = \eta \ , \tag{5}$$

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

$$(\hat{D}^\dagger \hat{D} + \mu_k^2)\psi_k = \eta \qquad k = 0, 1, \ldots, n-1 \ . \tag{6}$$

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
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.

5.2 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
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. This solver is described by a section

```
[Solver 8]
solver DFL_SAP_GCR
idfl 0
nkv 16
isolv 1
nmr 4
ncy 5
nmx 24
res 1.0e-8
```

that coincides with the section for the ordinary SAP preconditioned solver except for the solver symbol, and for the parameter `idfl` which specifies which deflation subspace should be used (see subsect. 5.3).

5.3 Deflation parameters

When the deflated solver is used, the parameters related to the deflation subspace must be specified in a few further sections.

In contrast to `openQCD`, the `openQ*D` package can use seveal deflation subspaces simultaneously. This is useful in QCD+QED simulation, since quarks with different electric charges must use different deflation subspaces. The deflation subspaces are labeled by an index `idfl` in the range 0,1,...,31 in much the same way as flavours, actions, forces and solvers. In practice this index is used only in the parameters sections of the `DFL_SAP_GCR` solver (see subsect. 5.2). The deflation parameters are distributed in four different sections. The of them, `Deflation subspace`, `Deflation projection` and `Deflation update scheme` are common to all deflation subspaces, while `Deflation subspace generation` must appear as many times as the number of requested deflation subspaces.

The block size `bs` of the deflation block grid and the number `Ns` of deflation modes per block are set by including the section

```
[Deflation subspace]
bs 8 4 4 4
Ns 20
```

For the generation of the deflation subspace, approximate eigenvalues of the Dirac operator are calculated. Each deflation subspace can use different parameters for the Dirac operator and different values for the twisted mass `mu`, which need to be specified in the section

```
[Deflation subspace generation 2]
qhat     0             # Dirac operator parameters start here
kappa    0.128
su3csw   1.234
u1csw    0.000
cF       0.95
cF'      1.0
theta    0.2 -0.4 0.5  # Dirac operator parameters end here
mu       0.02
ninv     5
nmr      4
ncy      5
```

The other parameters set in this section include 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 when the SAP preconditioner is invoked.

In the course of the inverse iteration, and in each iteration of the deflated solver, a deflation projection is applied that requires the little Dirac equation to be solved with low precision. The section

```
[Deflation projection]
nkv 24
nmx 256
res 1.0e-2
```

sets the parameters of the GCR solver used for this task. As usual, `nkv`, `nmx` and `res` are, respectively, the number of Krylov vectors generated before the GCR algorithm is restarted, the maximal total number of vectors that may be generated and the desired relative residue of the calculated solution.

Along the molecular-dynamics trajectories, the deflation subspace looses its efficiency and must therefore 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 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 `openQ*D` simulations must be reweighted. For example, the rational approximation requires reweighting in order to correct for the approximation error. In the case of the light quarks, reweighting is required if a twisted mass term was added as infrared regulator. For a given ensemble of field configurations, the program `ms1` computes stochastic estimates of the reweighting factors specified in a parameter file.

6.1 Twisted-mass reweighting factors for doublets

In ref. [8], two kinds of twisted-mass regularizations of the determinant $\det\{D^\dagger D\}$ were proposed which amount to replacing

$$\det\{D^\dagger D\} \to \det\{D^\dagger D + \mu^2\} \tag{7}$$

and

$$\det\{D^\dagger D\} \to \det\{(D^\dagger D + \mu^2)^2 (D^\dagger D + 2\mu^2)^{-1}\} \tag{8}$$

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,

$$W_1 = \det\det\{D^\dagger D (D^\dagger D + \mu^2)^{-1}\} \ , \tag{9}$$

$$W_2 = \det\{D^\dagger D (D^\dagger D + 2\mu^2)(D^\dagger D + \mu^2)^{-2}\} \tag{10}$$

can be estimated stochastically using Gaussian random quark fields. The input files parameters for these reweigthing factors are identical to the `openQCD` package and are described in sect. 6.1 of [9].

6.2 Reweighting factors for RHMC

Let $\tilde{R}$ and R be the optimal rational approximations of order $[n, n]$ for $(\hat{D}^\dagger \hat{D})^{-\alpha}$ and $(\hat{D}^\dagger \hat{D} + \hat{\mu}^2)^{-\alpha}$ respectively. As explained in [6], two reweighting factors are needed to correct for the twisted mass in the rational approximation ($W_{\rm rtm}$), and for the error due to the rational approximation itself ($W_{\rm rat}$).

The calculation of the reweighting factor

$$W_{\rm rtm} = \det[\tilde{R}^{-1} R] \tag{11}$$

requires two rational functions, R and $\tilde{R}$, that need to be defined in two parameter sections of the type

```
[Rational 0]
power        -1 2
degree       12
range        0.02 6.05
mu           0.0
```

```
[Rational 1]
power        -1 2
degree       12
range        0.02 6.05
mu           0.02
```

The parameter section for the reweighting factor $W_{\rm rtm}$ looks like

```
[Reweighting factor 2]
rwfact       RWRTM
ifl          1
irp          1 0
np           6 4 2
isp          1 0 2
nsrc         4
```

The parameter `ifl` is used to specify which flavour needs to be used. The line `irp` declares that the rational approximation 1 is used for R, and the rational approximation 0 is used for $\tilde{R}$. It is required that the two functions have the same degree (12 in this case). The operator $R^{-1}\tilde{R}$ is also a rational function of $\hat{Q}^2 = \hat{D}^\dagger \hat{D}$, which can broken up in factors as explained in [6]. The line `np` specifies that the first factor contains the first group of 6 poles and zeroes, the second factor contains the second group of 4 poles and zeroes, and finally the third factor contains the third group of 2 poles and zeroes, according to the formula

$$W_{\rm rtm} = \text{constant} \times \det \tilde{P}^{-1}_{1,5} \ \det \tilde{P}^{-1}_{6,10} \ \det \tilde{P}^{-1}_{11,12} \ . \tag{12}$$

The calculation of each determinant requires the solution of the Dirac equation. Solvers can be chosen independently for each factor by setting the values of `isp`. Each determinant is evaluated stochastically, using `nsrc` stochastic sources.

The parameters for the reweighting factor

$$W_{\rm rat} = \det[(\hat{D}^\dagger \hat{D})^\alpha \tilde{R}] \tag{13}$$

are specified in a section of the type

```
[Reweighting factor 1]
rwfact       RWRAT
ifl          1
irp          0
np           6 4 2
isp          1 0 2
nsrc         2
```

The parameter `ifl` is used to specify which flavour needs to be used. The rational function for $\tilde{R}$ is specified in `irp`. Notice that the exponent α is inherited from the rational function. The reweighting factor is estimated stochastically with a procedure described in detail in [6]. `nsrc` stochastic sources are used for this determination. Every time the operator $\tilde{R}$ needs to be applied to a stochastic sources, the rational function is broken up in factors pretty much in the same way as in the discussion for $W_{\rm rtm}$. The splitting is defined in the line `np`. For each factor, the solvers to be used are chosen independently in the line `isp`.

References

[1] A. Patella, *C* boundary conditions*, code documentation, `doc/cstar.pdf`.

[2] A. Patella, *Gauge actions*, code documentation, `doc/gauge_action.pdf`.

[3] A. Patella, *Dirac operator*, code documentation, `doc/dirac.pdf`.

[4] M. Luscher, *Molecular-dynamics quark forces*, code documentation, `doc/openQCD-1.6/forces.pdf`.

[5] M. Luscher, *Implementation of the lattice Dirac operator*, code documentation, `doc/openQCD-1.6/dirac.pdf`.

[6] A. Patella, *RHMC algorithm in `openQ*D`*, code documentation, `doc/rhmc.pdf`.

[7] M. Luscher, *Multi-shift conjugate gradient algorithm*, code documentation, `doc/openQCD-1.6/mscg.pdf`.

[8] M. Luscher and F. Palombi, *Fluctuations and reweighting of the quark determinant on large lattices*, PoS LATTICE **2008**, 049 (2008) [arXiv:0810.0946 [hep-lat]].

[9] M. Luscher, *Parameters of the openQCD main programs*, code documentation, `doc/openQCD-1.6/parms.pdf`.