Implementation of the lattice Dirac operator *

Martin Lüscher January 2012; revised November 2013

1. Introduction

In this note, the O(a)-improved Wilson–Dirac operator that is implemented in the openQCD package is defined, with all normalization conventions and boundary conditions specified explicitly. Even-odd preconditioning and some technical issues concerning the Sheikholeslami–Wohlert term are also covered.

2. Definition of the lattice Dirac operator

2.1 Quark fields

The theory is set up on a four-dimensional hypercubic lattice of size $N_0 \times N_1 \times N_2 \times N_3$, where N_μ is assumed to be even and not smaller than 4. For notational convenience, the lattice spacing is set to unity in this note. Unless stated otherwise, the Cartesian coordinates (x_0, x_1, x_2, x_3) of a lattice point x are assumed to be in the range $0 \le x_\mu < N_\mu$.

Quark fields $\psi(x)$ live on the sites x of the lattice. They have two indices, a Dirac index $A = 1, \ldots, 4$ and a colour index $\alpha = 1, 2, 3$ on which the gauge field acts. A quark field thus has $12N_0N_1N_3N_4$ complex components. In the openQCD package, SU(3) matrices, colour vectors and quark spinors are represented by structures that can be treated as single data items (see `include/su3.h`).

Depending on the boundary conditions for the gauge field [2], Schrödinger functional (SF) or anti-periodic boundary conditions are imposed on the quark fields in

* Based in part on notes written in collaboration with Peter Weisz and Ulli Wolff [1].

Table 1. Boundary conditions supported by the openQCD programs

Type	Time extent T	Gauge field	Quark fields
0	$N_0 - 1$	open	SF
1	N_0	SF	SF
2	N_0	open-SF	SF
3	N_0	periodic	anti-periodic

the time direction (see table 1). In all cases, periodic boundary conditions are chosen in the space directions. The boundary conditions can be selected in the openQCD main programs by specifying the type of boundary condition (the index listed in the first column of table 1) in the input parameter file [3]. Note that the effective time extent T of the lattice (second column) depends on the boundary conditions.

With SF boundary conditions, the quark fields satisfy

$$\psi(x)|_{x_0=0} = 0 \tag{2.1}$$

and additionally

$$\psi(x)|_{x_0=N_0-1} = 0 \tag{2.2}$$

in the case of the boundary conditions of type 0. The dimension of the space of quark fields is thus reduced to $12(N_0-1)N_1N_2N_3$ and $12(N_0-2)N_1N_2N_3$, respectively. All further specifications required for a complete description of the boundary conditions are deferred to the following subsections, since they concern the action of the lattice Dirac operator on the quark fields rather than the field space.

2.2 Wilson–Dirac operator

The (unimproved) Wilson–Dirac operator can be written in the compact form

$$D_{\rm w} = \sum_{\mu=0}^{3} \tfrac{1}{2} \left\{\gamma_\mu(\nabla^*_\mu + \nabla_\mu) - \nabla^*_\mu\nabla_\mu\right\}, \tag{2.3}$$

where

$$\nabla_\mu\psi(x) = U(x,\mu)\psi(x+\hat{\mu}) - \psi(x), \tag{2.4}$$

$$\nabla^*_\mu \psi(x) = \psi(x) - U(x-\hat{\mu},\mu)^{-1}\psi(x-\hat{\mu}), \tag{2.5}$$

denote the gauge-covariant forward and backward lattice derivatives in presence of the gauge field $U(x,\mu)$. The symbol $\hat{\mu}$ here stands for the unit vector in direction μ and the Dirac matrices γ_μ are specified in appendix A.

More explicitly, in the time range $0 < x_0 < N_0 - 1$, the action of the operator on a given quark field ψ is given by

$$D_{\mathrm{w}}\psi(x) = 4\psi(x) -$$

$$\sum_{\mu=0}^{3} \tfrac{1}{2}\left\{U(x,\mu)(1-\gamma_\mu)\psi(x+\hat{\mu}) + U(x-\hat{\mu},\mu)^{-1}(1+\gamma_\mu)\psi(x-\hat{\mu})\right\}, \tag{2.6}$$

where it is understood that the space coordinates $x_k \pm 1$ are taken modulo N_k for all $k = 1, 2, 3$ (thus imposing periodic boundary conditions in these directions). Note that the terms in eq. (2.6) refer to the quark field at time x_0 and $x_0 \pm 1$ only and are therefore unambiguously defined when x_0 is in the specified range.

The action of the Dirac operator at time $x_0 = 0$ and $x_0 = N_0 - 1$, on the other hand, depends on the boundary conditions and will be defined in subsection 2.4.

2.3 O(a)-improved lattice Dirac operator

Apart from the Sheikholeslami–Wohlert (SW) term [4] and a boundary O(a) correction term, it is now convenient to include the bare mass parameter m_0 in the Dirac operator (the operator thus becomes dependent on the quark flavour considered). The massive O(a)-improved Wilson–Dirac operator is then given by

$$D = D_{\mathrm{w}} + \delta D_{\mathrm{v}} + \delta D_{\mathrm{b}} + m_0, \tag{2.7}$$

where

$$\delta D_{\mathrm{v}}\psi(x) = c_{\mathrm{sw}} \sum_{\mu,\nu=0}^{3} \tfrac{i}{4}\sigma_{\mu\nu}\widehat{F}_{\mu\nu}(x)\psi(x), \tag{2.8}$$

$$\delta D_{\mathrm{b}}\psi(x) = \left\{(c_{\mathrm{F}}-1)\delta_{x_0,1} + (c'_{\mathrm{F}}-1)\delta_{x_0,T-1}\right\}\psi(x). \tag{2.9}$$

In these equations, c_{sw}, c_{F} and c'_{F} are improvement coefficients, the latter satisfying

$$c'_{\mathrm{F}} = c_{\mathrm{F}} \qquad \text{for boundary conditions of type 0 and 1,} \tag{2.10}$$

$$c'_{\mathrm{F}} = c_{\mathrm{F}} = 1 \quad \text{for boundary conditions of type 3} \tag{2.11}$$

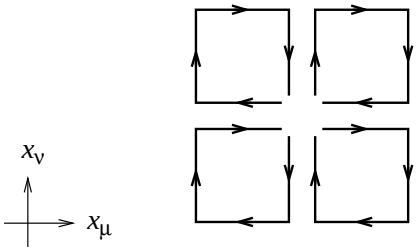

Fig. 1. Graphical representation of the products of gauge field variables contributing to the lattice field strength tensor (2.12). Each square corresponds to one of the terms in eq. (2.13).

(and thus $\delta D_{\rm b} = 0$ in the case of periodic boundary conditions). For the field strength of the gauge field,

$$\widehat{F}_{\mu\nu}(x) = \tfrac{1}{8}\left\{Q_{\mu\nu}(x) - Q_{\nu\mu}(x)\right\}, \tag{2.12}$$

$$\begin{aligned} Q_{\mu\nu}(x) = {} & U(x,\mu)U(x+\hat{\mu},\nu)U(x+\hat{\nu},\mu)^{-1}U(x,\nu)^{-1} + \\ & U(x,\nu)U(x-\hat{\mu}+\hat{\nu},\mu)^{-1}U(x-\hat{\mu},\nu)^{-1}U(x-\hat{\mu},\mu) + \\ & U(x-\hat{\mu},\mu)^{-1}U(x-\hat{\mu}-\hat{\nu},\nu)^{-1}U(x-\hat{\mu}-\hat{\nu},\mu)U(x-\hat{\nu},\nu) + \\ & U(x-\hat{\nu},\nu)^{-1}U(x-\hat{\nu},\mu)U(x+\hat{\mu}-\hat{\nu},\nu)U(x,\mu)^{-1}, \end{aligned} \tag{2.13}$$

the usual symmetric lattice expression is taken (see fig. 1). The normalization conventions adopted here coincide with the ones in ref. [5]. In particular, at tree-level of perturbation theory, on-shell O(a)-improvement is achieved by setting $c_{\rm sw} = c_{\rm F} = c'_{\rm F} = 1$.

2.4 Action of the Dirac operator near the boundaries

For all types of boundary conditions and at all times x_0 in the range $0 < x_0 < N_0 - 1$, the action of the Dirac operator D on a given quark field $\psi(x)$ is given by eqs. (2.6)–(2.13). In the specified range of time, all terms in these formulae are completely well defined and need no further explanation.

At time $x_0 = 0$ and $x_0 = N_0 - 1$, the action of the Dirac operator depends on the type of boundary conditions:

Type 0. Here one simply sets

$$D\psi(x)|_{x_0=0} = D\psi(x)|_{x_0=N_0-1} = 0. \tag{2.14}$$

This rule is in fact implied by the requirement that the Dirac operator must be a linear operator in the space of quark fields satisfying eqs. (2.1) and (2.2).

Type 1 and 2. Consistency with the condition (2.1) requires again that

$$D\psi(x)|_{x_0=0} = 0, \tag{2.15}$$

but at $x_0 = N_0 - 1$ one may choose the action of the Dirac operator to be given by the same equations as in the bulk of the lattice together with the prescription

$$\psi(x+\hat{0})\big|_{x_0=N_0-1} = 0, \tag{2.16}$$

which imposes Schrödinger functional boundary conditions at time N_0.

Type 3. In this case, the action of the Dirac operator is again given by eqs. (2.6)–(2.13), at all times in the range $0 \le x_0 \le N_0 - 1$, where

$$\psi(x)|_{x_0=-1} = -\,\psi(x)|_{x_0=N_0-1}\,, \qquad \psi(x)|_{x_0=N_0} = -\,\psi(x)|_{x_0=0}\,, \tag{2.17}$$

is to be inserted in eq. (2.6) for the otherwise undefined components of the quark field at time -1 and N_0.

In all cases, the field tensor and the SW term are only required at the interior points of the lattice. Both are unambiguously defined through eqs. (2.8),(2.12) and (2.13), at all these points, once the boundary conditions for the gauge field are taken into account. With the rules stated above, the definition of the lattice Dirac operator D is thus complete.

2.5 Block Dirac operators

Some of the solvers for the Dirac equation included in the openQCD package make use of a domain-decomposition (SAP) preconditioner. The domains are taken to be hypercubic blocks of lattice points. On each block Λ, the block Dirac operator D_Λ is defined by

$$D_\Lambda = P_\Lambda D P_\Lambda, \tag{2.18}$$

where

$$P_\Lambda \psi(x) = \begin{cases} \psi(x) & \text{if } x \in \Lambda, \\ 0 & \text{otherwise,} \end{cases} \tag{2.19}$$

projects any quark field ψ to the block.

The projection (2.18) amounts to imposing Dirichlet boundary conditions on the block boundaries. In addition, the quark fields on the blocks must satisfy the constraints (2.1),(2.2), if these are part of the boundary conditions on the global lattice.

3. Even-odd preconditioning

The discussion in the following applies to both the global and block Dirac operators, but for simplicity only the global Dirac operator is considered.

A lattice point x is classified as even or odd depending on whether the sum of its coordinates, $x_0 + x_1 + x_2 + x_3$, is even or odd. Any quark field ψ may be split into two parts,

$$\psi = \psi_{\mathrm{e}} + \psi_{\mathrm{o}}, \tag{3.1}$$

where ψ_{e} is supported on the even sites and ψ_{o} on the odd sites. If the lattice points are labeled such that the even ones come first, the Dirac operator assumes the block form

$$D = \begin{pmatrix} D_{\mathrm{ee}} & D_{\mathrm{eo}} \\ D_{\mathrm{oe}} & D_{\mathrm{oo}} \end{pmatrix}. \tag{3.2}$$

The operators D_{eo} and D_{oe}, for example, are the sums of the hopping terms in eq. (2.6) from the odd to the even and the even to the odd points respectively.

In the following, it is taken for granted that the diagonal part

$$D_{\mathrm{ee}} + D_{\mathrm{oo}} = M_0 + c_{\mathrm{sw}} \sum_{\mu,\nu=0}^{3} \tfrac{i}{4} \sigma_{\mu\nu} \widehat{F}_{\mu\nu}, \tag{3.3}$$

$$M_0 \psi(x) = \{4 + m_0 + (c_{\mathrm{F}} - 1)\delta_{x_0,1} + (c'_{\mathrm{F}} - 1)\delta_{x_0,T-1}\}\, \psi(x), \tag{3.4}$$

of the Dirac operator is invertible. For any given source field η, the Dirac equation $D\psi = \eta$ can then be solved by first solving

$$\hat{D}\psi_{\rm e} = \eta_{\rm e} - D_{\rm eo}D_{\rm oo}^{-1}\eta_{\rm o}, \tag{3.5}$$

for $\psi_{\rm e}$, where

$$\hat{D} = D_{\rm ee} - D_{\rm eo}D_{\rm oo}^{-1}D_{\rm oe} \tag{3.6}$$

denotes the even-odd preconditioned Dirac operator. After that the odd field component is obtained through

$$\psi_{\rm o} = D_{\rm oo}^{-1}\left\{\eta_{\rm o} - D_{\rm oe}\psi_{\rm e}\right\}. \tag{3.7}$$

Note that $\hat{D}$ is a linear operator acting on quark fields defined on the sublattice of all even points.

Even-odd preconditioning goes along with the factorization

$$\det D = \det D_{\rm oo}\det\hat{D} \tag{3.8}$$

of the quark determinant. An important detail to keep in mind is the fact that, by definition, $D_{\rm oe}\psi(x)$ vanishes at time 0 and $N_0 - 1$ if the boundary conditions require $D\psi(x)$ to vanish there. The operator product on the right of eq. (3.6) then involves the inverse of $D_{\rm oo}$ on the odd sites x at all other times only, and for the same reason, the determinant $\det D_{\rm oo}$ is a product of determinants, one for each odd point in this range of time.

4. Computation of $D_{\rm ee}$ and $D_{\rm oo}$

The computation of the SW term tends to be slow, because there are quite many products and sums of SU(3) matrices that must be evaluated. Taking the inverse of $D_{\rm oo}$, as is required for even-odd preconditioning, makes things even worse. The diagonal parts of the Dirac operator are therefore computed separately and stored in an array before the operator is applied. As long as the gauge field is unchanged, the array does not need to be updated, which saves a lot of work.

4.1 How many multiplications are required?

In the representation of the Dirac matrices specified in appendix A, the Pauli term

$$\sum_{\mu,\nu=0}^{3} \tfrac{i}{4}\sigma_{\mu\nu}\widehat{F}_{\mu\nu} = \sum_{k=1}^{3} \tfrac{i}{16} \begin{pmatrix} \sigma_k\left(\mathcal{E}_k - \mathcal{B}_k\right) & 0 \\ 0 & -\sigma_k\left(\mathcal{E}_k + \mathcal{B}_k\right) \end{pmatrix} \tag{4.1}$$

has a block diagonal form. A factor 8 was included here in the definition

$$\mathcal{E}_k = 8\widehat{F}_{0k}, \qquad \mathcal{B}_k = \sum_{l,j=1}^{3} 4\epsilon_{klj}\widehat{F}_{lj}, \tag{4.2}$$

of the electric and magnetic components of the field strength. They are then simply equal to $Q_{\mu\nu} - Q_{\mu\nu}^{\dagger}$ for some μ and ν.

The computation of the matrix $Q_{\mu\nu}(x)$ at fixed x, μ, ν requires 12 multiplications of SU(3) matrices. A straightforward code for the SW term will therefore perform 72 multiplications per point. One can do better by running through all plaquettes in the $(0,1)$-plane, then those in the $(0,2)$-plane, and so on. On each plaquette there are four ordered products of link variables to be added to the matrices $(Q_{\mu\nu} - Q_{\mu\nu}^{\dagger})(y)$ at the corners y of the plaquette. This requires 8 matrix multiplications per plaquette and thus a total of 48 multiplications per lattice point.

4.2 Storage format

The program that performs these computations stores the computed matrices

$$M(x) = M_0(x) + c_{\mathrm{sw}} \sum_{\mu,\nu=0}^{3} \tfrac{i}{4}\sigma_{\mu\nu}\widehat{F}_{\mu\nu}(x) \tag{4.3}$$

in an array of data structures. First come the matrices at all even points and then those at all odd points. In Dirac space the matrices have the block form

$$M(x) = \begin{pmatrix} A_+(x) & 0 \\ 0 & A_-(x) \end{pmatrix}, \tag{4.4}$$

$$A_{\pm}(x) = M_0(x) \pm c_{\mathrm{sw}} \sum_{k=1}^{3} \tfrac{i}{16}\sigma_k \left\{\mathcal{E}_k(x) \mp \mathcal{B}_k(x)\right\}, \tag{4.5}$$

as is the case for the Pauli term (4.1).

The fields $\mathcal{E}_k(x)$ and $\mathcal{B}_k(x)$ are antihermitian 3×3 matrices in colour space. So if the Pauli matrices in eq. (4.5) are written out as in

$$\sum_{k=1}^{3} i\sigma_k a_k = \begin{pmatrix} ia_3 & ia_1 + a_2 \\ ia_1 - a_2 & -ia_3 \end{pmatrix}, \tag{4.6}$$

the matrices $A_\pm(x)$ assume the form of hermitian 6×6 matrices. Such matrices can be represented by an array $u_0, \ldots, u_{35}$ of real numbers according to

$$\begin{pmatrix} u_0 & u_6 + iu_7 & u_8 + iu_9 & u_{10} + iu_{11} & u_{12} + iu_{13} & u_{14} + iu_{15} \\ . & u_1 & u_{16} + iu_{17} & u_{18} + iu_{19} & u_{20} + iu_{21} & u_{22} + iu_{23} \\ . & . & u_2 & u_{24} + iu_{25} & u_{26} + iu_{27} & u_{28} + iu_{29} \\ . & . & . & u_3 & u_{30} + iu_{31} & u_{32} + iu_{33} \\ . & . & . & . & u_4 & u_{34} + iu_{35} \\ . & . & . & . & . & u_5 \end{pmatrix}, \tag{4.7}$$

where the entries below the diagonal are related to the other entries by hermiticity. The matrices $A_\pm(x)$ are thus stored in structures whose only element is an array of this kind.

4.3 Miscellaneous remarks

As explained in subsect. 2.4, the SW term is only needed at the interior points of the lattice. Independently of the quark mass and the improvement coefficients, the matrices $A_\pm(x)$ at the boundaries of the lattice (if any) are set to unity when the SW term is calculated.

In the openQCD package, the field tensor (2.12) is stored in memory and is reused when it is up-to-date. Changes in the quark mass are therefore propagated to the SW term at nearly no cost.

5. Inversion of $\boldsymbol{D}_{\rm ee}$ and $\boldsymbol{D}_{\rm oo}$

As explained in sect. 3, even-odd preconditioning requires the inversion of the diagonal part $D_{\rm oo}$ of the Dirac operator on the odd sites of the lattice. This amounts to computing the inverse of a large number of 6×6 matrices. Householder triangularization with subsequent inversion by back-substitution is a numerically safe and reasonably fast method that can be applied here. The technique is widely used and

is described in many text books (see refs. [6,7], for example). The precise form of the algorithm that is implemented in the openQCD package is discussed in this section.

The fact that $D_{\rm oo}$ may turn out to be ill-conditioned is a possible source of difficulty. Catastrophically large rounding errors are then practically unavoidable when the matrices $M(x)$ are inverted (see appendix B). The program that performs the inversion returns with a non-zero error code in this case, which is propagated to the top level of the simulation program.

5.1 Householder triangularization

In the following, the inversion of a general complex $n\times n$ matrix A is considered. The Householder algorithm transforms the matrix to upper triangular form by applying a series of reflections of the type

$$R=1-2\frac{u\otimes u^{\dagger}}{\|u\|^2}, \tag{5.1}$$

where u is some non-zero complex vector in n dimensions. One needs $n-1$ reflections to completely triangularize A, i.e. the triangular matrix T that one constructs is given by

$$T=R_{n-1}R_{n-2}\ldots R_1A. \tag{5.2}$$

The basic idea is to choose the reflections R_k recursively in such a way that the matrix

$$R_{j-1}R_{j-2}\ldots R_1A \tag{5.3}$$

has no non-zero entries below the diagonal in the first $j-1$ columns.

Suppose we have achieved this for all $j\leq k$ and let us denote the k'th column vector of the matrix $R_{k-1}\ldots R_1A$ by v. We then take R_k to be of the form (5.1) with

$$u_l=\begin{cases}0 & \text{if } l<k,\\ v_k-y_k & \text{if } l=k,\\ v_l & \text{if } l>k,\end{cases} \tag{5.4}$$

where y_k is given by

$$y_k=-\frac{v_k}{|v_k|}\,r,\qquad r^2=\sum_{j=k}^{n}|v_j|^2,\qquad r\geq 0. \tag{5.5}$$

It is easy to check that the first $k-1$ columns of $R_{k-1}\dots R_1 A$ are left intact when this matrix is multiplied with R_k, while the k'th column vector becomes

$$(v_1, \dots, v_{k-1}, y_k, 0, \dots, 0). \tag{5.6}$$

If we continue in this way up to $k = n-1$, the final matrix (5.2) will hence be upper triangular with diagonal elements $y_1, \dots, y_{n-1}, -y_n$.

Evidently the algorithm breaks down if u vanishes for some k. Noting

$$\tfrac{1}{2}\|u\|^2 = r^2 + r|v_k|, \tag{5.7}$$

it follows that this happens if and only if A is singular. During the execution of the program one can easily check that r is positive and take the appropriate action if it vanishes.

Another potentially unstable situation occurs when $v_k = 0$. It turns out, however, that this does not present a fundamental difficulty. If v_k is so small that $|v_k| + r = r$ to machine precision, we simply set $y_k = -r$. The reflection R_k, defined through eqs. (5.1),(5.4), then transforms the column vector v to the vector (5.6) up to rounding errors. Equation (5.7) remains true in the same sense, i.e. to machine precision. The choice $y_k = -r$ is hence completely satisfactory in this case.

5.2 Inversion of T

We now describe the computation of the inverse S of the triangular matrix T, assuming that none of the diagonal elements of T vanish. It can be shown that S is also upper triangular, i.e. the matrix elements t_{ij} and s_{ij} of T and S are equal to zero for all $i > j$.

The linear equations from which S is to be determined are

$$\sum_{j=i}^{k} t_{ij} s_{jk} = \delta_{ik}. \tag{5.8}$$

Choosing $i = k$ one immediately concludes that

$$s_{ii} = 1/t_{ii}. \tag{5.9}$$

For $i < k$ the equations can then be written in the form

$$s_{ik} = -s_{ii} \sum_{j=i+1}^{k} t_{ij} s_{jk}. \tag{5.10}$$

They can be solved recursively, first setting $k = n$, then $k = n - 1$ and so on, while for each k one starts with $i = k - 1$ and continues to $i = 1$. It is not difficult to see that the element t_{ik} may be overwritten by s_{ik} in this process, since t_{ik} is not needed at the later stages of the recursion. In other words, the inversion can be achieved in place without a second array.

5.3 Final steps

Using the orthogonality of the Householder reflections, it is trivial to show that

$$A^{-1} = SR_{n-1}R_{n-2}\dots R_1. \tag{5.11}$$

The computation can thus be completed by evaluating the product on the right-hand side of this equation recursively.

6. Programs

The programs that compute the field tensor (2.12),(2.13) and the SW term (4.3) are contained in the directories `modules/tcharge` and `modules/sw_term` respectively. A list of all available functions is included in the `README` files in these directories, while the functionality of the programs is briefly described at the top of the program files.

The programs for the Dirac operator are in the directory `modules/dirac`. There are programs implementing the action of D and $\hat{D}$ on quark fields on the full lattice and on blocks of lattice points. Most programs admit a further mass parameter μ and add the twisted-mass term $i\mu\gamma_5$ to the Dirac operator. Moreover, it is possible to restrict the twisted-mass term to the even sites of the lattice by setting a flag (see `modules/flags/lat_parms.c`).

If boundary conditions of type 3 are chosen, the programs implement periodic boundary conditions for all fields and switch to anti-periodic boundary conditions for the quark fields by changing the sign of the link variables $U(x, 0)$ at time $x_0 = N_0 - 1$ before calling any programs for the Dirac operator. The function that changes the sign of these link variables is `chs_ubnd()` in the module `lattice/bcnds.c`. When gauge configurations are written to disk, all link variables are guaranteed to have the sign-change undone and thus have determinant 1 within rounding errors. Measurement programs based on openQCD modules should therefore call `chs_ubnd()` immediately after importing a field configuration from disk in order to ensure that

type 3 boundary conditions are correctly implemented (see `main/ms4.c` for example).

Appendix A

It is advantageous to work with a chiral representation of the Dirac matrices, where

$$\gamma_\mu = \begin{pmatrix} 0 & e_\mu \\ (e_\mu)^\dagger & 0 \end{pmatrix}. \tag{A.1}$$

A possible choice for the 2×2 matrices e_μ is

$$e_0 = -1, \qquad e_k = -i\sigma_k \tag{A.2}$$

($k = 1, 2, 3$, and σ_k are the Pauli matrices). It is then easy to check that

$$(\gamma_\mu)^\dagger = \gamma_\mu, \qquad \{\gamma_\mu, \gamma_\nu\} = 2\delta_{\mu\nu}. \tag{A.3}$$

Furthermore, if we define $\gamma_5 = \gamma_0\gamma_1\gamma_2\gamma_3$, we have

$$\gamma_5 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{A.4}$$

In particular, $(\gamma_5)^\dagger = \gamma_5$ and $(\gamma_5)^2 = 1$. The hermitian matrices

$$\sigma_{\mu\nu} = \frac{i}{2}\,[\gamma_\mu, \gamma_\nu] \tag{A.5}$$

that appear in the SW term are explicitly given by

$$\sigma_{0k} = \begin{pmatrix} \sigma_k & 0 \\ 0 & -\sigma_k \end{pmatrix}, \qquad \sigma_{ij} = -\epsilon_{ijk} \begin{pmatrix} \sigma_k & 0 \\ 0 & \sigma_k \end{pmatrix}, \tag{A.6}$$

where ϵ_{ijk} is the totally anti-symmetric tensor with $\epsilon_{123} = 1$.

The hopping terms in the Wilson–Dirac operator involve the projected spinors

$$\phi = (1 - s\gamma_\mu)\psi, \qquad s = \pm 1. \tag{A.7}$$

In the programs that implement the operator, these terms are hand-programmed, following the lines

$$
\begin{aligned}
& s = +1, \quad \mu = 0 : \\
& \phi_1 = \psi_1 + \psi_3 \\
& \phi_2 = \psi_2 + \psi_4 \\
& \phi_3 = \phi_1 \\
& \phi_4 = \phi_2
\end{aligned} \tag{A.8}
$$

$$
\begin{aligned}
& s = -1, \quad \mu = 0 : \\
& \phi_1 = \psi_1 - \psi_3 \\
& \phi_2 = \psi_2 - \psi_4 \\
& \phi_3 = -\phi_1 \\
& \phi_4 = -\phi_2
\end{aligned} \tag{A.9}
$$

$$
\begin{aligned}
& s = +1, \quad \mu = 1 : \\
& \phi_1 = \psi_1 + i\psi_4 \\
& \phi_2 = \psi_2 + i\psi_3 \\
& \phi_3 = -i\phi_2 \\
& \phi_4 = -i\phi_1
\end{aligned} \tag{A.10}
$$

$$
\begin{aligned}
& s = -1, \quad \mu = 1 : \\
& \phi_1 = \psi_1 - i\psi_4 \\
& \phi_2 = \psi_2 - i\psi_3 \\
& \phi_3 = i\phi_2 \\
& \phi_4 = i\phi_1
\end{aligned} \tag{A.11}
$$

$$
\begin{aligned}
& s = +1, \quad \mu = 2 : \\
& \phi_1 = \psi_1 + \psi_4 \\
& \phi_2 = \psi_2 - \psi_3
\end{aligned}
$$

$$\phi_3 = -\phi_2$$
$$\phi_4 = \phi_1 \tag{A.12}$$

$$s = -1, \quad \mu = 2 :$$
$$\phi_1 = \psi_1 - \psi_4$$
$$\phi_2 = \psi_2 + \psi_3$$
$$\phi_3 = \phi_2$$
$$\phi_4 = -\phi_1 \tag{A.13}$$

$$s = +1, \quad \mu = 3 :$$
$$\phi_1 = \psi_1 + i\psi_3$$
$$\phi_2 = \psi_2 - i\psi_4$$
$$\phi_3 = -i\phi_1$$
$$\phi_4 = i\phi_2 \tag{A.14}$$

$$s = -1, \quad \mu = 3 :$$
$$\phi_1 = \psi_1 - i\psi_3$$
$$\phi_2 = \psi_2 + i\psi_4$$
$$\phi_3 = i\phi_1$$
$$\phi_4 = -i\phi_2 \tag{A.15}$$

Appendix B

In this appendix, the numerical stability of the matrix inversion algorithm described in sect. 5 is discussed. For the matrices of interest the inversion requires a relatively small number of arithmetic operations and the accumulation of rounding errors is hence not expected to produce an instability. The algorithm is in any case known to be well behaved in this respect [6,7].

A problem may however arise if the matrix is ill-conditioned. Any initial numerical uncertainty in the matrix may lead to large changes in the calculated inverse in this case. The result that one obtains is only meaningful up to such variations.

B.1 Condition number

Let A be any complex invertible $n \times n$ matrix. In the following the norm $\|v\|$ of an n-component complex vector v is defined through

$$\|v\|^2 = \sum_{k=1}^{n} |v_k|^2. \tag{B.1}$$

The associated norm of A and its condition number are then given by

$$\|A\| = \sup_{\|v\|=1} \|Av\|, \tag{B.2}$$

$$k(A) = \|A\| \, \|A^{-1}\|. \tag{B.3}$$

In terms of the eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \tag{B.4}$$

of the hermitean matrix $A^\dagger A$ we have

$$k(A) = \sqrt{\lambda_1/\lambda_n}. \tag{B.5}$$

In particular, matrices with condition numbers near 1 are close to being unitary up to an overall normalization factor. When the condition number is large the matrix is said to be ill-conditioned.

If A is hermitean its eigenvalues $\mu_1, \ldots, \mu_n$ may be ordered such that $\mu_k^2 = \lambda_k$ and the condition number $k(A)$ is then equal to $|\mu_1/\mu_n|$. It should be emphasized, however, that the eigenvalues of A are in general not directly related to its condition number. If we take

$$A = \begin{pmatrix} 1 & z \\ 0 & 1 \end{pmatrix}, \tag{B.6}$$

for example, the eigenvalues of A are both equal to 1, while

$$k(A) = \rho + \sqrt{\rho^2 - 1}, \qquad \rho = 1 + \tfrac{1}{2}|z|^2, \tag{B.7}$$

depends on z and can be made arbitrarily large.

B.2 Stability of the linear system $Av = b$

We now consider the perturbed linear system

$$(A + \epsilon B)v = b, \tag{B.8}$$

where ϵ is a small parameter and $\|B\| \leq \|A\|$. The perturbation ϵB may be regarded as a model for the numerical uncertainty in A which may have incurred during its computation. We are then interested in the sensitivity of the solution vector v on the perturbation. Neglecting higher orders in ϵ, we have

$$v = (1 - \epsilon A^{-1}B)A^{-1}b, \tag{B.9}$$

and the relative deviation of the perturbed from the unperturbed solution is hence bounded by

$$\frac{\|v - A^{-1}b\|}{\|A^{-1}b\|} \leq \epsilon k(A). \tag{B.10}$$

This suggests that the numerical error on the solution vector may be amplified by a factor $k(A)$ relative to the error on the matrix A.

In any given case the actual error may be significantly smaller, but the bound (B.10) is not unrealistic in general. For illustration we again consider the matrix (B.6) and take

$$B = \begin{pmatrix} 0 & 0 \\ z^* & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{B.11}$$

One easily checks that $\|B\| \leq \|A\|$ and a short calculation then yields

$$\frac{\|v - A^{-1}b\|}{\|A^{-1}b\|} = \epsilon|z|\sqrt{1 + |z|^2} \tag{B.12}$$

up to terms of order ϵ^2. For large $|z|$ (large condition numbers in other words) the inequality (B.10) is hence saturated.

We thus conclude that there can be large significance losses when calculating the inverse of an ill-conditioned matrix. In particular, the inversion algorithm described in sect. 5 may not be safe in such cases.

B.3 Estimating condition numbers

An exact calculation of the condition number $k(A)$ requires the computation of the extremal eigenvalues of $A^\dagger A$. This may be rather time-consuming and a faster method to estimate $k(A)$ is clearly needed.

To this end we introduce the Frobenius norm

$$\|A\|_{\mathrm{F}}^2 = \sum_{i,j=1}^{n} |a_{ij}|^2, \tag{B.13}$$

where a_{ij} are the matrix elements of A. Noting

$$\|A\|_{\mathrm{F}}^2 = \mathrm{tr}\,\{A^\dagger A\} = \sum_{k=1}^{n} \lambda_k, \tag{B.14}$$

it is immediately clear that

$$\|A\| \leq \|A\|_{\mathrm{F}},$$

and an upper bound on the condition number $k(A)$ is hence given by

$$k(A) \leq k_{\mathrm{F}}(A) = \|A\|_{\mathrm{F}}\, \|A^{-1}\|_{\mathrm{F}}. \tag{B.15}$$

With some additional work one may also establish the inequality

$$k_{\mathrm{F}}(A) \leq \tfrac{1}{2} n \left[k(A) + 1/k(A)\right], \tag{B.16}$$

which shows that $k_{\mathrm{F}}(A)$ overestimates $k(A)$ by at most a factor $\frac{1}{2}n$ if $k(A)$ is large (which is the case of interest).

B.4 Stability criterion

In the program that computes the inverse of D_{oo}, the inversion is considered to be safe if the 6×6 matrices $A = A_{\pm}(x)$ satisfy the bound

$$k_{\mathrm{F}}(A) \leq k_{\max} = 100. \tag{B.17}$$

Significance losses of more than 2 decimal places are then excluded and the inverted matrices are obtained with an estimated numerical precision of at least 14 decimal places (assuming standard 64 bit floating-point arithmetic). The program returns 1 if the bound is violated and 0 otherwise.

A last point to be mentioned here is that the diagonal elements of the triangular matrix T are bounded from below through

$$|t_{ii}| \geq \|A\|_{\mathrm{F}}/k_{\max} \tag{B.18}$$

if (B.17) holds. In particular, the Householder triangularization is guaranteed to be numerically safe. Equation (B.18) is obtained straightforwardly by inserting

$$\|A^{-1}\|_{\mathrm{F}} = \|S\|_{\mathrm{F}} \geq |s_{ii}| \tag{B.19}$$

in the definition of $k_{\mathrm{F}}(A)$ and using eq. (5.9).

References

[1] M. Lüscher, P. Weisz, U. Wolff, TAO programs for the Dirac operator in O(a)-improved lattice QCD, ALPHA collaboration internal report (May 1997)

[2] *Gauge actions in openQCD simulations*, `doc/gauge_action.pdf`

[3] *Parameters of the openQCD main programs*, `doc/parms.pdf`

[4] B. Sheikholeslami, R. Wohlert, *Improved continuum limit lattice action for QCD with Wilson fermions*, Nucl. Phys. B259 (1985) 572

[5] M. Lüscher, S. Sint, R. Sommer, P. Weisz, *Chiral symmetry and O(a) improvement in lattice QCD*, Nucl. Phys. B478 (1996) 365

[6] G. W. Stewart, *Introduction to Matrix Computations* (Academic Press, New York, 1973).

[7] G. H. Golub, C. F. van Loan, *Matrix Computations* (Johns Hopkins University Press, Baltimore, 1989)