

Implementation of the lattice Dirac operator^{*}

Martin Lüscher

January 2012

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 specified explicitly. Apart from the boundary conditions, the conventions coincide with the ones adopted in the public DD-HMC code [2,3]. Even-odd preconditioning and some technical issues concerning the Sheikholeslami–Wohlert term are also covered.

2. Definition of the lattice Dirac operator

2.1 Lattice fields

As usual, the theory is set up on a hypercubic lattice with spacing $a = 1$ and integer sizes N_0, \dots, N_3 in the four space-time directions, where N_μ is assumed to be even and not smaller than 4. With open boundary conditions [4], the time coordinate x_0 of the lattice sites x runs from 0 to $N_0 - 1$ inclusive, the terminal time-slices being the boundaries of the lattice. In particular, the physical time-like extent of the lattice is $N_0 - 1$ (rather than N_0).

A lattice gauge field $U(x, \mu)$ is an assignment of an $SU(3)$ matrix to all links (x, μ) in the specified range of time. For convenience, memory space is allocated for the gauge field on the links $(x, 0)$ at time $x_0 = N_0 - 1$ too, but the field variables residing there are set to zero from the beginning and remain unchanged in the course of the simulations.

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

Quark fields $\psi(x)$ live on the sites x of the lattice. They have two indices, a Dirac index $A = 1, \dots, 4$ and a colour index $\alpha = 1, 2, 3$ on which the gauge field acts. The values of the quark fields at the boundaries $x_0 = 0$ and $x_0 = N_0 - 1$ are not used and are set zero when the fields are initialized. 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`).

2.2 Wilson–Dirac operator

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

$$D_w = \sum_{\mu=0}^3 \frac{1}{2} \{ \gamma_\mu (\nabla_\mu^* + \nabla_\mu) - \nabla_\mu^* \nabla_\mu \}, \quad (2.1)$$

where

$$\nabla_\mu \psi(x) = U(x, \mu) \psi(x + \hat{\mu}) - \psi(x), \quad (2.2)$$

$$\nabla_\mu^* \psi(x) = \psi(x) - U(x - \hat{\mu}, \mu)^{-1} \psi(x - \hat{\mu}), \quad (2.3)$$

denote the gauge-covariant forward and backward lattice derivatives. 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 range $0 < x_0 < N_0 - 1$, the action of the operator on a given quark field ψ is given by

$$D_w \psi(x) = 4\psi(x) - \sum_{\mu=0}^3 \frac{1}{2} \{ U(x, \mu) (1 - \gamma_\mu) \psi(x + \hat{\mu}) + U(x - \hat{\mu}, \mu)^{-1} (1 + \gamma_\mu) \psi(x - \hat{\mu}) \}, \quad (2.4)$$

where it is understood that the coordinates $x_\mu \pm 1$ are taken modulo N_μ for all $\mu = 1, 2, 3$ (i.e. in these directions, periodic boundary conditions are imposed). At the boundaries $x_0 = 0$ and $x_0 = N_0 - 1$, on the other hand, $D_w \psi(x) = 0$ by definition. The operator is thus guaranteed to map quark fields onto quark fields satisfying the same boundary conditions.

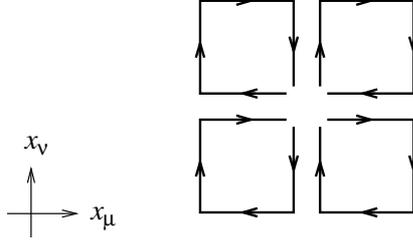


Fig. 1. Graphical representation of the products of gauge field variables contributing to the lattice field strength tensor (2.8). Each square corresponds to one of the terms in eq. (2.9).

2.3 $O(a)$ -improved operator

Apart from the Sheikholeslami–Wohlert term [5] and a boundary $O(a)$ correction, it is convenient to include the bare quark mass m_0 in the Dirac operator. The massive $O(a)$ -improved Wilson–Dirac operator is then given by

$$D = D_w + \delta D_v + \delta D_b + m_0, \quad (2.5)$$

where

$$\delta D_v \psi(x) = c_{sw} \sum_{\mu, \nu=0}^3 \frac{i}{4} \sigma_{\mu\nu} \widehat{F}_{\mu\nu}(x) \psi(x), \quad (2.6)$$

$$\delta D_b \psi(x) = (c_F - 1) \{ \delta_{x_0,1} + \delta_{x_0, N_0-2} \} \psi(x). \quad (2.7)$$

In these equations, c_{sw} and c_F are adjustable improvement coefficients and

$$\widehat{F}_{\mu\nu}(x) = \frac{1}{8} \{ Q_{\mu\nu}(x) - Q_{\nu\mu}(x) \}, \quad (2.8)$$

$$\begin{aligned} Q_{\mu\nu}(x) = & U(x, \mu) U(x + \hat{\mu}, \nu) U(x + \hat{\nu}, \mu)^{-1} U(x, \nu)^{-1} + \\ & U(x, \nu) U(x - \hat{\mu} + \hat{\nu}, \mu)^{-1} U(x - \hat{\mu}, \nu)^{-1} U(x - \hat{\mu}, \mu) + \\ & U(x - \hat{\mu}, \mu)^{-1} U(x - \hat{\mu} - \hat{\nu}, \nu)^{-1} U(x - \hat{\mu} - \hat{\nu}, \mu) U(x - \hat{\nu}, \nu) + \\ & U(x - \hat{\nu}, \nu)^{-1} U(x - \hat{\nu}, \mu) U(x + \hat{\mu} - \hat{\nu}, \nu) U(x, \mu)^{-1}, \end{aligned} \quad (2.9)$$

a symmetric lattice representation of the field strength of the gauge field (see fig. 1).

The normalization conventions in these equations coincide with the ones adopted in ref. [6] and in most other papers on the subject. Boundary conditions are imposed as described before. In particular, $D\psi(x) = 0$ at time $x_0 = 0$ and $x_0 = N_0 - 1$, and the field strength is calculated assuming periodic boundary conditions for the gauge field in the space directions. Note that the field tensor is only needed in the range $0 < x_0 < N_0 - 1$.

2.4 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 rectangular blocks of lattice points. On each block Λ , the block Dirac operator D_Λ is defined by

$$D_\Lambda = P_\Lambda D P_\Lambda, \quad (2.10)$$

where

$$P_\Lambda \psi(x) = \begin{cases} \psi(x) & \text{if } x \in \Lambda, \\ 0 & \text{otherwise,} \end{cases} \quad (2.11)$$

projects any quark field ψ to the block.

The projection (2.10) amounts to imposing Dirichlet boundary conditions on the block boundaries. In addition, the boundary conditions imposed at time $x_0 = 0$ and $x_0 = N_0 - 1$ on the global lattice are inherited.

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,

$$\psi = \psi_e + \psi_o, \quad (3.1)$$

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

$$D = \begin{pmatrix} D_{ee} & D_{eo} \\ D_{oe} & D_{oo} \end{pmatrix}. \quad (3.2)$$

The operators D_{eo} and D_{oe} , for example, are the sums of the hopping terms in eq. (2.4) 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

$$D_{ee} + D_{oo} = M_0 + c_{\text{sw}} \sum_{\mu, \nu=0}^3 \frac{i}{4} \sigma_{\mu\nu} \hat{F}_{\mu\nu}, \quad (3.3)$$

$$M_0 \psi(x) = \begin{cases} (4 + m_0 + c_{\text{F}} - 1) \psi(x) & \text{if } x_0 = 1 \text{ or } x_0 = N_0 - 2, \\ (4 + m_0) \psi(x) & \text{otherwise,} \end{cases} \quad (3.4)$$

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

$$\hat{D}\psi_e = \eta_e - D_{eo} D_{oo}^{-1} \eta_o, \quad (3.5)$$

for ψ_e , where

$$\hat{D} = D_{ee} - D_{eo} D_{oo}^{-1} D_{oe} \quad (3.6)$$

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

$$\psi_o = D_{oo}^{-1} \{ \eta_o - D_{oe} \psi_e \}. \quad (3.7)$$

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

$$\det D = \det D_{oo} \det \hat{D} \quad (3.8)$$

of the quark determinant. An important detail to keep in mind is the fact that, by definition, $D_{oe}\psi(x)$ vanishes at the boundaries $x_0 = 0$ and $x_0 = N_0 - 1$ of the lattice. As a consequence, the operator product on the right of eq. (3.6) involves the inverse of D_{oo} on the odd sites x in the range $0 < x_0 < N_0 - 1$ only. For the same reason, the determinant $\det D_{oo}$ is a product of determinants, one for each odd point in this range of time.

4. Computation of D_{ee} and D_{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_{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

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

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

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

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

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

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

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

$$A_{\pm}(x) = M_0(x) \pm c_{\text{sw}} \sum_{k=1}^3 \frac{i}{16} \sigma_k \{ \mathcal{E}_k(x) \mp \mathcal{B}_k(x) \}, \quad (4.5)$$

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

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

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

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

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 sect. 3, the SW term is not needed at time $x_0 = 0$ and $x_0 = N_0 - 1$. Independently of the quark mass and the improvement coefficients, the matrices $A_{\pm}(x)$ at these lattice points are set to unity when the SW term is calculated.

In the openQCD package, the field tensor (2.8) 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 D_{ee} and D_{oo}

As explained in sect. 3, even-odd preconditioning requires the inversion of the diagonal part D_{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. [7,8], for example). The precise form of the algorithm that is implemented in the openQCD package is discussed in this section.

The fact that D_{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

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

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

$$T = R_{n-1} R_{n-2} \dots R_1 A. \quad (5.2)$$

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

$$R_{j-1} R_{j-2} \dots R_1 A \quad (5.3)$$

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 $R_{k-1} \dots R_1 A$ by v . We then take R_k to be of the form (5.1)

with

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

where y_k is given by

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

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

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

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

$$\frac{1}{2} \|u\|^2 = r^2 + r|v_k|, \quad (5.7)$$

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

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

Choosing $i = k$ one immediately concludes that

$$s_{ii} = 1/t_{ii}. \quad (5.9)$$

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

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

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

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

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 (2.8) 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.

Appendix A

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

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

A possible choice for the 2×2 matrices e_μ is

$$e_0 = -1, \quad e_k = -i\sigma_k \quad (\text{A.2})$$

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

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

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

$$\gamma_5 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (\text{A.4})$$

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

$$\sigma_{\mu\nu} = \frac{i}{2} [\gamma_\mu, \gamma_\nu] \quad (\text{A.5})$$

that appear in the SW term are explicitly given by

$$\sigma_{0k} = \begin{pmatrix} \sigma_k & 0 \\ 0 & -\sigma_k \end{pmatrix}, \quad \sigma_{ij} = -\epsilon_{ijk} \begin{pmatrix} \sigma_k & 0 \\ 0 & \sigma_k \end{pmatrix}, \quad (\text{A.6})$$

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

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

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

$$s = +1, \quad \mu = 0 :$$

$$\phi_1 = \psi_1 + \psi_3$$

$$\begin{aligned}
\phi_2 &= \psi_2 + \psi_4 \\
\phi_3 &= \phi_1 \\
\phi_4 &= \phi_2
\end{aligned} \tag{A.8}$$

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

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

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

$$\begin{aligned}
s &= +1, \quad \mu = 2 : \\
\phi_1 &= \psi_1 + \psi_4 \\
\phi_2 &= \psi_2 - \psi_3 \\
\phi_3 &= -\phi_2 \\
\phi_4 &= \phi_1
\end{aligned} \tag{A.12}$$

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

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

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

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 [7,8].

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

$$\|v\|^2 = \sum_{k=1}^n |v_k|^2. \quad (\text{B.1})$$

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

$$\|A\| = \sup_{\|v\|=1} \|Av\|, \quad (\text{B.2})$$

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

In terms of the eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \quad (\text{B.4})$$

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

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

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 μ_1, \dots, μ_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

$$A = \begin{pmatrix} 1 & z \\ 0 & 1 \end{pmatrix}, \quad (\text{B.6})$$

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

$$k(A) = \rho + \sqrt{\rho^2 - 1}, \quad \rho = 1 + \frac{1}{2}|z|^2, \quad (\text{B.7})$$

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

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

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

$$\|A\|_{\text{F}}^2 = \sum_{i,j=1}^n |a_{ij}|^2, \quad (\text{B.13})$$

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

$$\|A\|_{\text{F}}^2 = \text{tr} \{A^\dagger A\} = \sum_{k=1}^n \lambda_k, \quad (\text{B.14})$$

it is immediately clear that

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

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

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

With some additional work one may also establish the inequality

$$k_{\text{F}}(A) \leq \frac{1}{2}n [k(A) + 1/k(A)], \quad (\text{B.16})$$

which shows that $k_{\text{F}}(A)$ 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

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

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

$$|t_{ii}| \geq \|A\|_{\text{F}}/k_{\text{max}} \quad (\text{B.18})$$

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

$$\|A^{-1}\|_{\text{F}} = \|S\|_{\text{F}} \geq |s_{ii}| \quad (\text{B.19})$$

in the definition of $k_{\text{F}}(A)$ 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] M. Lüscher, *Schwarz-preconditioned HMC algorithm for two-flavour lattice QCD*, Comput. Phys. Commun. 165 (2005) 199;
- [3] <http://cern.ch/luscher/DD-HMC/index.html>
- [4] M. Lüscher, Stefan Schaefer, *Lattice QCD without topology barriers*, JHEP 1107 (2011) 036
- [5] B. Sheikholeslami, R. Wohlert, *Improved continuum limit lattice action for QCD with Wilson fermions*, Nucl. Phys. B259 (1985) 572
- [6] M. Lüscher, S. Sint, R. Sommer, P. Weisz, *Chiral symmetry and $O(a)$ improvement in lattice QCD*, Nucl. Phys. B478 (1996) 365
- [7] G. W. Stewart, *Introduction to Matrix Computations* (Academic Press, New York, 1973).
- [8] G. H. Golub, C. F. van Loan, *Matrix Computations* (Johns Hopkins University Press, Baltimore, 1989)