SciDAC-5 Wilson fermion conventions

Peter A Boyle

March 17, 2025

Contents

1 Introduction **2**

1.1 Conventions . . . 2

1.1.1 Dirac matrices . . . 2

1.1.2 Tensor types . . . 3

1.1.3 Wilson matrix . . . 4

1.2 Correctness tests . . . 5

1.2.1 Unit gauge solution by Fourier transform . . . 5

1.2.2 Specific volume/mass results . . . 8

1.2.3 Lie algebra, Lie group and random gauge fields . . . 10

1.2.4 Free Wilson eigenvalues . . . 11

1.2.5 Gauge invariance . . . 11

1.3 Known properties of the non-Free D_W eigenvectors/eigenvalues . . . 12

1.3.1 γ_5 Hermiticity . . . 13

1.3.2 Shifted spectrum . . . 13

1.3.3 Pairing of complex eigenvalues . . . 13

1.3.4 γ_5 orthogonality of eigenvalues . . . 14

1.3.5 γ_5 symmetry, singular value decomposition and the normal equations . . . 14

1.3.6 Near null vectors . . . 15

1.3.7 Singular value decomposition deflation and multigrid . . . 15

1.3.8 Mathworks example matrix . . . 16

1.3.9 $PV^\dagger M$ deflation . . . 17

1.4 Optimised implementation . . . 17

1.4.1 Domain Wall Fermions . . . 19

1.5 Coarsening gauge theories . . . 21

2 General Cayley form **23**

Chapter 1

Introduction

This document summarises conventions used in the generalised chiral fermion support used in Grid and BFM.

It is quite obviously based on a lot of work by many authors in the field including Furman, Shamir, the RBC and UKQCD collaborations, Neuberger, Luscher, Kikukawa, Yamada, Borici, Edwards, Heller, Brower, Neff, Joo, Kennedy, Wenger.

I acknowledge access to the Chroma code base with which I have regressed my code (sometimes with an optional convention transformation). There are many tests within the tests subdirectory of the BFM package which verify that BFM/BAGEL reproduces the corresponding Chroma operator.

1.1 Conventions

First my conventions.

1.1.1 Dirac matrices

The Dirac matrices operate on spin indices of our Fermion fields. These are 4×4 complex valued matrices forming an anticommuting matrix basis, that generalises the Quaternions to a larger dimension.

A spinor is a rank 4 spin-vector which is acted upon with these Dirac matrices.

The Dirac matrices statisfy

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = 2\delta_{\mu\nu},$$

and in particular $\gamma_\mu^2 = 1$.

γ_5 is introduced and anticommutes with all the other Dirac matrices (so they form a totally anticommuting set).

Since a Lorentz vector p^μ is a commuting scalar while the Dirac matrices anticommute, we have

$$p_\mu\gamma_\mu p_\nu\gamma_\nu = p_\mu p_\nu \frac{1}{2}\left(\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu\right) = p_\mu p_\mu = p^2$$

We will focus initially on the internal indices as these are the building blocks assembled in Lattice container classes. Every Lattice container class constructor requires a Grid object pointer.

The spin basis is:

$$\gamma_x = \begin{pmatrix} 0 & 0 & 0 & i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}$$

$$\gamma_y = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

$$\gamma_z = \begin{pmatrix} 0 & 0 & i & 0 \\ 0 & 0 & 0 & -i \\ -i & 0 & 0 & 0 \\ 0 & i & 0 & 0 \end{pmatrix}$$

$$\gamma_t = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\gamma_5 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Multiplication of a spin vector or matrix can be implemented via subroutines that simply permute data and apply signs.

1.1.2 Tensor types

The Tensor data structures are built up from fundamental scalar matrix and vector classes::

```
template<class vobj      > class iScalar { private: vobj _internal ; }
template<class vobj,int N> class iVector { private: vobj _internal[N] ; }
template<class vobj,int N> class iMatrix { private: vobj _internal[N][N] ; }
```

These are template classes and can be passed a fundamental scalar or vector type, or nested to form arbitrarily complicated tensor products of indices. All mathematical expressions are defined to operate recursively, index by index.

Presently the constants N_c and N_d are globally predefined. However, this is planned for changed in future and policy classes for different theories (e.g. QCD, QED, SU2 etc...) will contain these constants and enable multiple theories to coexist more naturally.

Arbitrary tensor products of fundamental scalar, vector and matrix objects may be formed in principle by the basic Grid code.

For Lattice field theory, we define types according to the following tensor product structure ordering. The suffix "D" indicates either double types, and replacing with "F" gives the corresponding single precision type.

Type definitions are provided in qcd/QCD.h to give the internal index structures of QCD codes. For example::

```
template<typename vtype>
using iSinglet                  = iScalar<iScalar<iScalar<vtype> > >;
using iSpinMatrix               = iScalar<iMatrix<iScalar<vtype>, Ns> >;
using iColourMatrix             = iScalar<iScalar<iMatrix<vtype, Nc> > > ;
using iSpinColourMatrix         = iScalar<iMatrix<iMatrix<vtype, Nc>, Ns> >;
using iLorentzColourMatrix      = iVector<iScalar<iMatrix<vtype, Nc> >, Nd > ;
using iDoubleStoredColourMatrix = iVector<iScalar<iMatrix<vtype, Nc> >, Nds > ;
using iSpinVector               = iScalar<iVector<iScalar<vtype>, Ns> >;
using iColourVector             = iScalar<iScalar<iVector<vtype, Nc> > >;
using iSpinColourVector          = iScalar<iVector<iVector<vtype, Nc>, Ns> >;
using iHalfSpinVector            = iScalar<iVector<iScalar<vtype>, Nhs> >;
using iHalfSpinColourVector      = iScalar<iVector<iVector<vtype, Nc>, Nhs> >;
```

Giving the type table:

Lattice	Lorentz	Spin	Colour	scalartype	Field
Scalar	Scalar	Scalar	Scalar	RealD	RealD
Scalar	Scalar	Scalar	Scalar	ComplexD	ComplexD
Scalar	Scalar	Scalar	Matrix	ComplexD	ColourMatrixD
Scalar	Vector	Scalar	Matrix	ComplexD	LorentzColourMatrixD
Scalar	Scalar	Vector	Vector	ComplexD	SpinColourVectorD
Scalar	Scalar	Vector	Vector	ComplexD	HalfSpinColourVectorD
Scalar	Scalar	Matrix	Matrix	ComplexD	SpinColourMatrixD

The types are implemented via a recursive tensor nesting system.

Here, the prefix "i" indicates for internal use, preserving the template nature of the class. Final types are declared with vtype selected to be both scalar and vector, appropriate to a single datum, or stored in a partial SoA transformed lattice object:

```
// LorentzColour
typedef iLorentzColourMatrix<Complex  > LorentzColourMatrix;
typedef iLorentzColourMatrix<ComplexF > LorentzColourMatrixF;
typedef iLorentzColourMatrix<ComplexD > LorentzColourMatrixD;

typedef iLorentzColourMatrix<vComplex > vLorentzColourMatrix;
typedef iLorentzColourMatrix<vComplexF> vLorentzColourMatrixF;
typedef iLorentzColourMatrix<vComplexD> vLorentzColourMatrixD;
```

Arbitrarily deep tensor nests may be formed. Grid uses a positional and numerical rule to associate indices for contraction in the Einstein summation sense.

Symbolic name	Number	Position
LorentzIndex	0	left
SpinIndex	1	middle
ColourIndex	2	right

The conventions are that the index ordering left to right are: Lorentz, Spin, Colour. A scalar type (either real or complex, single or double precision) is be provided to the innermost structure.

1.1.3 Wilson matrix

- x runs over four dimensional cartesian coordinate system, of size $L^3 \times T$.
- μ is a direction index running from 0 to 3
- s (where used) is fifth dimensional coordinate of size L_s
- $U_\mu(x)$ is a 3×3 complex color matrix valued gauge field
- $\psi(x)$ is a 4×3 complex spin-color vector valued Fermion field
- The Wilson (sparse) matrix operates on this vector space, with elements involving Dirac matrices on spin indices, and color matrices on color indices.
- M is a real valued "mass".
- We use Einstein summation convention where repeated indices are summed over their natural range
- Color indices use roman letters i, j, k running from 0 to 2
- Spin indices use Greek letters α, β running from 0 to 3

- Lorentz indices use Greek letters μ, ν running from 0 to 3

Usual Wilson matrix operates on Fermion spin-colour vector and is

$$D^{ij}_{W\alpha\beta}(x,y) = (M+4)\delta_{\alpha\beta}\delta_{ij}\delta_{x,y} - \frac{1}{2}D^{ij}_{\text{hop}\alpha\beta}(x,y),$$

where

$$D^{ij}_{\text{hop}\alpha\beta}(x,y)\psi^j_\beta(y) = \left[(1-\gamma_\mu)_{\alpha\beta}U^{ij}_\mu(x)\delta_{x+\mu,y} + (1+\gamma_\mu)_{\alpha\beta}U^{\dagger ij}_\mu(y)\delta_{x-\mu,y}\right]\psi^j_\beta(y) \tag{1.1}$$

1.2 Correctness tests

1.2.1 Unit gauge solution by Fourier transform

Fourier transform

- coordinate space to momentum space

$$\tilde{F}(p) = \frac{1}{V}\sum_x e^{-ip.x}F(x)$$

- momentum space to coordinate space

$$F(x) = \sum_p e^{ip.x}\tilde{F}(p)$$

- CHECK: Resolution of identity with $F(x) = e^{iq.x}$

$$\tilde{F}(p) = \frac{1}{V}\sum_x e^{-ip.x}e^{iq.x} = \delta_{p,q}$$

$$F(x) = \sum_p e^{ip.x}\delta_{p,q} = e^{iq.x}$$

https://github.com/paboyle/Grid/blob/develop/tests/core/Test_fft_matt.cc

The Wilson operator in the free field (U=1) is translation invariant:

$$D_W(x,y) = D_W(x-y)$$

The (two coordinate) Fourier transform of a translation invariant matrix takes simple form and leads to a momentum conserving delta:

$$\frac{1}{V^2}\sum_{xy} e^{-iqx}e^{ipy}D_{xy} = \frac{1}{V^2}\sum_{xy} e^{-iqx}e^{ipy}D(x-y) \tag{1.2}$$

$$= \frac{1}{V^2}\sum_{xu} e^{-iqx}e^{ip(x-u)}D(u) \tag{1.3}$$

$$= \frac{1}{V^2}\sum_{xu} e^{i(p-q)x}e^{-ip(u)}D(u) \tag{1.4}$$

$$= \delta_{p,q}\tilde{D}(p) \tag{1.5}$$

Application of D_W is therefore a convolution,

$$D_W(x,y)\psi(y) = \sum_y D_W(x-y)\psi(y)$$

$$\tilde{D}_W(p) = \frac{1}{V}\sum_x D_W(x)e^{-ipx}$$

Convolution theorem for this norm of FT is:

$$\begin{aligned}
\sum_y F(x-y)G(y) &= \sum_{p,q}\sum_y \tilde{F}(p)\tilde{G}(q)e^{ip.(x-y)}e^{iq.(y)} && (1.6)\\
&= \sum_{p,q}\tilde{F}(p)\tilde{G}(q)e^{ip.x}\sum_y e^{i(q-p).(y)} && (1.7)\\
&= \sum_{p,q}\tilde{F}(p)\tilde{G}(q)e^{ip.x}V\delta_{p,q} && (1.8)\\
&= \sum_p V\tilde{F}(p)\tilde{G}(p)e^{ip.x} && (1.9)\\
&= \sum_p \tilde{C}(p)e^{ip.x} && (1.10)
\end{aligned}$$

Thus we can identify the Fourier transform of the convolution $C = F * G$ as

$$\tilde{C}(p) = V\tilde{F}(p)\tilde{G}(p)$$

We can Fourier transform the Wilson matrix D_W in the left and right variables with $U_\mu = 1$.

$$\begin{aligned}
D_W(p) &= \frac{1}{V^2}\sum_{x,y} e^{-iq.x}e^{ip.y}D_W(x-y) && (1.11)\\
&= \frac{1}{V^2}\sum_x e^{i(p-q).x}\left[(M+4) - \sum_\mu e^{ip_\mu}\frac{1}{2}(1-\gamma_\mu) + e^{-ip_\mu}\frac{1}{2}(1+\gamma_\mu)\right] && (1.12)\\
&= \frac{1}{V}\delta_{q,p}\left(M + \sum_\mu (1-\cos p_\mu) + i\gamma_\mu \sin p_\mu\right) && (1.13)\\
&= \frac{1}{V}\delta_{q,p}\left(M + \sum_\mu 2\sin^2\frac{p_\mu}{2} + i\gamma_\mu \sin p_\mu\right) && (1.14)
\end{aligned}$$

Thus, coordinate space multiplication by D_W (i.e. a convolution with D_W) reduces to multiplication in Fourier space by

$$\left(M + \sum_\mu 2\sin^2\frac{p_\mu}{2} + i\gamma_\mu \sin p_\mu\right)$$

This is also relatively easy to invert, and the inverse is

$$\frac{M + \sum_\mu 2\sin^2\frac{p_\mu}{2} - i\gamma_\mu \sin p_\mu}{\sum_\mu \sin^2 p_\mu + (2\sum_\mu \sin^2\frac{p_\mu}{2} + m)^2}$$

Check 1: Wilson matrix check

Thus a convenient first test for a new code for $D_w(x,y)$ is to

- Pick a random vector b
- apply the coordinate space operator $D_w b$
- Fourier transform the result to obtain $\tilde{\psi}(p)$
- Fourier transform to obtain $\tilde{b}(p)$
- Multiply by $D_W(p)$ to obtain $D_W(p)\tilde{b}(p)$ Eq 1.14

- Compare $D_W(p)b(p)$ to $\tilde{\psi}(p)$

See Lines 82-138 of: https://github.com/paboyle/Grid/blob/develop/tests/core/Test_fft_matt.cc

```
RealD mass=0.1;
WilsonFermionD Dw(Umu,GRID,RBGRID,mass);

Dw.M(src,ref);
std::cout << "Norm src "<<norm2(src)<<std::endl;
std::cout << "Norm Dw x src "<<norm2(ref)<<std::endl;
{
  FFT theFFT(&GRID);

  ////////////////
  // operator in Fourier space
  ////////////////
  tmp =ref;
  theFFT.FFT_all_dim(result,tmp,FFT::forward);
  std::cout<<"FFT[ Dw x src ]  "<< norm2(result)<<std::endl;

  tmp = src;
  theFFT.FFT_all_dim(src_p,tmp,FFT::forward);
  std::cout<<"FFT[ src      ]  "<< norm2(src_p)<<std::endl;

  /////////////////////////////////////////////////////////////////
  // work out the predicted FT from Fourier
  /////////////////////////////////////////////////////////////////
  auto FGrid = &GRID;
  LatticeFermionD    Kinetic(FGrid); Kinetic = Zero();
  LatticeComplexD    kmu(FGrid);
  LatticeInteger     scoor(FGrid);
  LatticeComplexD    sk (FGrid); sk = Zero();
  LatticeComplexD    sk2(FGrid); sk2= Zero();
  LatticeComplexD    W(FGrid); W= Zero();
  LatticeComplexD    one(FGrid); one =ComplexD(1.0,0.0);
  ComplexD ci(0.0,1.0);

  for(int mu=0;mu<Nd;mu++) {
    LatticeCoordinate(kmu,mu);
    RealD TwoPiL =  M_PI * 2.0/ latt_size[mu];
    kmu = TwoPiL * kmu;
    sk2 = sk2 + 2.0*sin(kmu*0.5)*sin(kmu*0.5);
    sk  = sk  +     sin(kmu)    *sin(kmu);
    Kinetic = Kinetic + sin(kmu)*ci*(Gamma(Gmu[mu])*src_p);
  }

  W = mass + sk2;
  Kinetic = Kinetic + W * src_p;

  std::cout<<"Momentum space src         "<< norm2(src_p)<<std::endl;
  std::cout<<"Momentum space Dw x src    "<< norm2(Kinetic)<<std::endl;
  std::cout<<"FT[Coordinate space Dw]    "<< norm2(result)<<std::endl;

  result = result - Kinetic;
  std::cout<<"diff "<< norm2(result)<<std::endl; // Should be zero up to rounding
}
```

This test passes:

```
./Test_fft_matt --grid 8.8.8.16
---
Grid : Message : 0.163137 s : Grid is setup to use 1 threads
Norm src 196235
Norm Dw x src 4.07469e+06
FFT[ Dw x src ]  3.33799e+10
FFT[ src      ]  1.60756e+09
Momentum space src         1.60756e+09
Momentum space Dw x src    3.33799e+10
FT[Coordinate space Dw]    3.33799e+10
diff 3.61034e-21
```

Check 2: Wilson matrix check

Grid also implements the inverse of the free Wilson operator via convolution. using equation **??**. A related check is that the Fourier based inverse (Greens function) indeed inverts the coordinate space operator we have coded up as D_w.

```
    std::cout << " =======================================" <<std::endl;
    std::cout << " Checking FourierFreePropagator x Dw = 1" <<std::endl;
    std::cout << " =======================================" <<std::endl;
    std::cout << "Dw src = " <<norm2(src)<<std::endl;
    std::cout << "Dw tmp = " <<norm2(tmp)<<std::endl;
    Dw.M(src,tmp);
    Dw.FreePropagator(tmp,ref,mass);

    std::cout << "Dw ref = " <<norm2(ref)<<std::endl;

    ref = ref - src;

    std::cout << "Dw ref-src = " <<norm2(ref)<<std::endl;
```

This test passes:

```
 =======================================
 Checking FourierFreePropagator x Dw = 1
 =======================================
Dw src = 196235
Dw tmp = 196235
Dw ref = 196235
Dw ref-src = 3.32075e-26
```

1.2.2 Specific volume/mass results

Check 3: full solver check

We can check solvers on the free field using Fourier convolution and equation **??**.

$$D_W\psi = b \iff \tilde{\psi}(p) = D_W^{-1}(p)\tilde{b}(p)$$

This can of course be FT'd back to coordinate space and the check applied directly on the result of a Krylov or multigrid solver. See the above file linked for examples.

```
    LatticeFermionD    src(&GRID); gaussian(pRNG,src);
    LatticeFermionD    tmp(&GRID);
    LatticeFermionD    ref(&GRID);
    LatticeFermionD    diff(&GRID);

    src=Zero();
    Coordinate point(4,0); // 0,0,0,0
    SpinColourVectorD ferm;
    ferm=Zero();
    ferm()(0)(0) = ComplexD(1.0);
    pokeSite(ferm,src,point);

    RealD mass=0.1;
    WilsonFermionD Dw(Umu,GRID,RBGRID,mass);

    // Momentum space prop
    std::cout << " Solving by FFT and Feynman rules" <<std::endl;
    Dw.FreePropagator(src,ref,mass) ;

    LatticeFermionD    result(&GRID);

    ////////////////////////////////////////////////////////////////////////
    // Conjugate gradient on normal equations system
    ////////////////////////////////////////////////////////////////////////
    std::cout << " Solving by Conjugate Gradient (CGNE)" <<std::endl;
    Dw.Mdag(src,tmp);
    src=tmp;
    MdagMLinearOperator<WilsonFermionD,LatticeFermionD> HermOp(Dw);
    ConjugateGradient<LatticeFermionD> CG(1.0e-10,10000);
    CG(HermOp,src,result);

    ////////////////////////////////////////////////////////////////////////
    std::cout << " Taking difference" <<std::endl;
    std::cout << "Dw result "<<norm2(result)<<std::endl;
    std::cout << "Dw ref     "<<norm2(ref)<<std::endl;

    diff = ref - result;
    std::cout << "result - ref     "<<norm2(diff)<<std::endl;

    DumpSliceNorm("Slice Norm Solution ",result,Nd-1);
```

We take an $8^3 \times 16$ the $N_c = 3$ Wilson action with input mass 0.1 and either unit gauge links or gauge links that are a random gauge transform of unity.

The source is a kronecker delta in space, spin and colour: $b^j_\beta(y) = \delta_{y,0}\delta_{j,0}\delta_{\beta,0}$, so contains a single real number 1 in the first element and zero everywhere else.

We compute by FFT as described above that the solution should have timeslice by timeslice norm, and obtain:

```
****************************************
Wilson Mom space 4d propagator
****************************************
 Solving by FFT and Feynman rules
 Solving by Conjugate Gradient (CGNE)
Grid : Message : 13.988509 s : ConjugateGradient Converged on iteration 110 Computed residual 6.4878e-1
 Taking difference
Dw result 0.0918289
Dw ref     0.0918289
result - ref     1.77143e-21
Slice Norm Solution  0 S {S {S {(0.0708605,0)}}}
Slice Norm Solution  1 S {S {S {(0.00509478,0)}}}
Slice Norm Solution  2 S {S {S {(0.00151241,0)}}}
Slice Norm Solution  3 S {S {S {(0.000945715,0)}}}
Slice Norm Solution  4 S {S {S {(0.000767837,0)}}}
Slice Norm Solution  5 S {S {S {(0.000674655,0)}}}
Slice Norm Solution  6 S {S {S {(0.000617168,0)}}}
Slice Norm Solution  7 S {S {S {(0.000584637,0)}}}
Slice Norm Solution  8 S {S {S {(0.000574033,0)}}}
Slice Norm Solution  9 S {S {S {(0.000584637,0)}}}
Slice Norm Solution  10 S {S {S {(0.000617168,0)}}}
Slice Norm Solution  11 S {S {S {(0.000674655,0)}}}
Slice Norm Solution  12 S {S {S {(0.000767837,0)}}}
Slice Norm Solution  13 S {S {S {(0.000945715,0)}}}
Slice Norm Solution  14 S {S {S {(0.00151241,0)}}}
Slice Norm Solution  15 S {S {S {(0.00509478,0)}}}
```

This should be true on either a unit gauge configuration or a random gauge transform of the unit gauge configuration. Next section will discuss enhanced tests with non-trivial gauge links.

1.2.3 Lie algebra, Lie group and random gauge fields

In order to perform random gauge transforms we must create random elements of the SU(3) special ($\det g = 1$) unitary ($g^\dagger g = 1$) group.

This is simplest to perform in the Lie algebra su(3) of the group SU(3).

We take eight real valued Gaussian random fields $\lambda^a(x)$ and write $\Lambda(x) = \lambda^a(x) T^a$ and

$$g(x) = e^{i\lambda^a(x)T^a} = \sum_{n=0}^{12} \frac{1}{n!} \Lambda(X)^n$$

Where in practice a 12th order Taylor expansion is sufficent since Λ is bounded and 12! is large.

Here, the generators T^a are related to the Gell Mann matrices and are:

$$T^0 = \frac{1}{2}\begin{pmatrix} 0 & i & 0 \\ -i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$T^1 = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$T^2 = \frac{1}{2}\begin{pmatrix} 0 & 0 & i \\ 0 & 0 & 0 \\ -i & 0 & 0 \end{pmatrix}$$

$$T^3 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$T^4 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & i \\ 0 & -i & 0 \end{pmatrix}$$

$$T^5 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$T^6 = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$T^7 = \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & -\frac{2}{\sqrt{3}} \end{pmatrix}$$

1.2.4 Free Wilson eigenvalues

As shown above, due to translational invariance, the free Wilson matrix is diagonal in Fourier space,

$$\frac{1}{V^2}\sum_{xy} e^{-iqx}e^{ipy}D_{xy} = \delta_{p,q}\tilde{D}(p), \tag{1.15}$$

where

$$\tilde{D}(p) = \left(M + \sum_\mu 2\sin^2\frac{p_\mu}{2} + i\gamma_\mu \sin p_\mu\right)$$

We can easily compute the eigenvalues of $D(p)$ and hence those of free Wilson fermions.

The term proportional to the identity, $M + \sum_\mu 2\sin^2\frac{p_\mu}{2}$ dictates the real part of the eigenvalues, while the eigenvectors are dictated by the eigenvalue equation of the non-diagonal term,

$$i\gamma_\mu \sin p_\mu \psi = \alpha\psi$$

Recalling $a_\nu\gamma_\nu a_\mu\gamma_m u = a^2$ arises from the anti-commutavity and squaring to one property of the Euclidean Dirac matrices, we see that multiplying by $i\gamma_\nu \sin p_\nu$ we have

$$i^2\gamma_\nu \sin p_\nu \gamma_\mu \sin p_\mu \psi = i^2\left(\sum_\mu \sin^2 p_\mu\right) = i\gamma_\nu \sin p_\nu \alpha\psi = \alpha^2\psi,$$

and so

$$\alpha^2 = -\left(\sum_\mu \sin^2 p_\mu\right)$$

and the eigenvalue λ_i of the Dirac matrix portion of the Wilson term is imaginary with $\lambda_i = \pm i\sqrt{\sum_\mu \sin^2 p_\mu}$.

The possible eigenvalues for each Fourier momentum mode p are thus

$$\lambda_p = \lambda_r + \lambda_i = \left(M + \sum_\mu 2\sin^2\frac{p_\mu}{2} \pm i\sqrt{\sum_\mu \sin^2 p_\mu}\right)$$

1.2.5 Gauge invariance

The (energy) action is invariant under a local redefintion of the fields $U_\mu(x)$ and $\psi(x)$.

In the case of U(1) gauge theory this is equivalent to the phase of a quantum mechanical wave function being unobservable, but when the electromagnetic vector potential can be redefined a local phase can be absorbed.

We introduce a free (randomly chosen) gauge rotation matrix

$$g(x) \in SU(3)$$

so that

$$g^\dagger(x)g(x) = 1\forall x$$

. The action is gauge invariant if we *transform* simultaneously transform $U_\mu(x)$ and $\psi(x)$ as follows:

$$U_\mu(x) \to U'_\mu(x) = g(x)U_\mu(x)g^\dagger(x+\mu)$$

$$\psi(x) \to \psi'(x) = g(x)\psi(x)$$

Fermion fields live on nodes of the grid and transform in the fundemental representation of SU(3). Gauge fields live on edges of the grid and transform in the adjoint representation of SU(3). The gauge links are gauge covariant parallel transporters along an edge of the cartesian grid.

sketch

Consequently,

- Any path made out of naturally connecting products of $\psi^\dagger$, ψ and gauge links U is invariant:

$$\psi^\dagger U_1 U_2 \ldots U_n \psi \to \psi^\dagger g^\dagger g U_1 g^\dagger g U_2 g^\dagger g \ldots g^\dagger g U_n g^\dagger g \psi = \psi^\dagger U_1 U_2 \ldots U_n \psi$$

- Any trace of a closed loop of gauge links is invariant:

$$\mathrm{Tr}\left[U_1 U_2 \ldots U_n\right] \to \mathrm{Tr}\left[g^\dagger U_1 g^\dagger g U_2 g^\dagger g \ldots g^\dagger U_n g\right]$$

  were all factors of $g^\dagger g = 1$ and using the cyclic property of the trace.

In particular, the local (and hence global) site norm of a Fermion field ψ is invariant under a gauge transform, since $\psi^\dagger(x)\psi(x) \to \psi^\dagger(x)g(x)^\dagger g(x)\psi = \psi^\dagger(x)\psi(x)$

Thus, the solution from a point source right hand side has a norm that is gauge invariant, and can be computed by Fourier transform on a unit gauge, but the time-sliced norm does not change if instead computed on a *random* gauge transform of the unit gauge.

We compute by FFT as described above that the solution should have timeslice by timeslice norm as follows on $8^3 \times 16$ with a mass of $m = 0.1$.

```
0 0.0708605
1 0.00509478
2 0.00151241
3 0.000945715
4 0.000767837
5 0.000674655
6 0.000617168
7 0.000584637
8 0.000574033
9 0.000584637
10 0.000617168
11 0.000674655
12 0.000767837
13 0.000945715
14 0.00151241
15 0.00509478
```

This provides an excellent and robust test of the code with non-trival coefficient data.

1.3 Known properties of the non-Free D_W eigenvectors/eigenvalues

The spectrum of the gauged Wilson operator is only tractable numerically.

However there are known properties to the spectrum which we derive/explain here.

1.3.1 γ_5 Hermiticity

The D_w operator has γ_5 Hermiticity, because γ_5 anticommutes with the other Dirac matrices and due to the structure of the Wilson operator.

The adjoint is

$$
\begin{aligned}
D_W(x,y) &= (M+4)\delta_{x,y} - \frac{1}{2}\left(\delta_{y,x+\mu}(1-\gamma_m u)U(x,y) + \frac{1}{2}\delta_{y,x-\mu}(1+\gamma_m u)U^\dagger(y,x)\right) && (1.16)\\
D_W^\dagger(y,x) &= (M+4)\delta_{y,x} - \frac{1}{2}\left(\delta_{y,x+\mu}(1-\gamma_m u)U^\dagger(y,x) + \frac{1}{2}\delta_{y,x-\mu}(1+\gamma_m u)U^\dagger(y,x)\right) && (1.17)\\
&= (M+4)\delta_{x,y} - \frac{1}{2}\left(\delta_{x,y-\mu}(1-\gamma_m u)U(x,y) + \frac{1}{2}\delta_{x,y+\mu}(1+\gamma_m u)U(x,y)\right) && (1.18)\\
&= (M+4)\delta_{x,y} - \frac{1}{2}\left(\frac{1}{2}\delta_{x,y+\mu}(1+\gamma_m u)U(x,y) + \delta_{x,y-\mu}(1-\gamma_m u)U(x,y)\right) && (1.19)\\
&= \gamma_5 D_W(x,y)\gamma_5 && (1.20)
\end{aligned}
$$

Thus, because γ_5 anticommutes with the xyzt Dirac matrces, we can implement the adjoint operator by merely flipping the signs of the Dirac matrices in D_W.

With this we see that the operator $H_W = \gamma_5 D_W$ is Hermitian ($H_W^\dagger = D_W^\dagger\gamma_5 = \gamma_5 D_W \gamma_5\gamma_5 = \gamma_5 D_W = H_W$).

We denote right eigenvectors of D_W as ϕ_i and H_W as χ_i, satisfying,

$$D_W\phi_i = \lambda_i\phi_i,$$

and

$$H_W\chi_i = \mu_i\chi_i,$$

1.3.2 Shifted spectrum

If the massless Wilson eigenvectors are ϕ_i, given by

$$D_W(m=0)\phi_i = \lambda^0\phi_i$$

Then the massive Wilson operator $D_W(m)$ obviously has the same eigenvectors ϕ_i and eigenvalues $\lambda_i^m = \lambda_0^i + m$.

A consequence of this is that any real eigenvalue becomes a zero eigenvalue for some mass parameter $m = -\lambda_i^0$.

If $D_W(m)\phi = 0$ then $\gamma_5 D_W(m)\phi = 0$ so zero modes of D_W and H_W coincide. By continuity arguments their near null-spectrum must be similar.

1.3.3 Pairing of complex eigenvalues

Consider an eigenvector

$$
\begin{aligned}
D\phi &= \lambda_i\phi && (1.21)\\
\Rightarrow \gamma_5 D\gamma_5^2\phi &= \lambda_i\gamma_5\phi && (1.22)\\
\Rightarrow D^\dagger\gamma_5\phi &= \lambda_i\gamma_5\phi && (1.23)\\
\Rightarrow \phi^\dagger\gamma_5 D &= \lambda_i^*\phi^\dagger\gamma_5 && (1.24)\\
\Rightarrow \det(D-\lambda_i^* I) &= 0 && (1.25)
\end{aligned}
$$

Since right and left eigenvalues are the same (eigenvectors differ in a non-hermitian operator), we see that if λ_i is complex (has non-zero imaginary component) then λ_i^* is ALSO an eigenvalue.

1.3.4 γ_5 orthogonality of eigenvalues

Consider two right eigenvectors ϕ_i and ϕ_j with eigenvalues λ_i and λ_j

$$\begin{aligned} \phi_i^\dagger \gamma_5 D \phi_j &= \lambda_i^* \phi_i^\dagger \gamma_5 \phi_j && (1.26) \\ &= \lambda_j \phi_i^\dagger \gamma_5 \phi_j && (1.27) \end{aligned}$$

So,

$$(\lambda_i^* - \lambda_j)\phi_i^\dagger \gamma_5 \phi_j = 0$$

and either

$$\lambda_i^* = \lambda_j$$

OR

$$\phi_i^\dagger \gamma_5 \phi_j = 0$$

When $i = j$ we see that either the eigenvalue is real and $\phi_i^\dagger \gamma_5 \phi_j \neq 0$ is allowed, or the eigenvalue has non-zero imaginary part and $\phi_i^\dagger \gamma_5 \phi_j = 0$.

In other words:

- complex eigenvalues have zero chirality and come in conjugate pairs
- real eigenvalues in general have non-zero chirality $\chi = \phi_i^\dagger \gamma_5 \phi_i$
- Eigenvectors with unrelated eigenvalues have vanishing chiral inner product $\phi_i^\dagger \gamma_5 \phi_j = 0$

1.3.5 γ_5 symmetry, singular value decomposition and the normal equations

Suppose D_W has right eigenvectors R_i such that,

$$D_W R_i = \lambda_i R_i,$$

or in matrix notation with diagonal matrix Λ of eigenvalues:

$$D_W R = R\Lambda.$$

Assuming R is invertible (i.e. assuming D_W is not a defective matrix) we have,

$$D_W = R\Lambda R^{-1}$$

The columns of R are not necessarily mutually orthogonal. The inverse R^{-1} resolves linear combinations of non-orthogonal eigenvectors, such as $aR_1 + bR_2$ into the fundamental a and b components.

The left eigenvectors are such that

$$L_i^\dagger D_W = \lambda_i L_i^\dagger,$$

and,

$$L^\dagger D_W = \Lambda L^\dagger.$$

As noted above, the left eigenvectors can be related to right eigenvectors with the complex conjugate eigenvalue via γ_5 hermititicy:

$$\begin{aligned} L\Lambda^\dagger &= D_W^\dagger L = \gamma_5 D_W \gamma_5 L && (1.28) \\ D_W \gamma_5 L &= \gamma_5 L \Lambda^\dagger, && (1.29) \end{aligned}$$

implying that $\gamma_5 L = R$.

The action of γ_5 transforms a left eigenvector with eigenvalue λ into a right eigenvector with eigenvalue λ^*. If λ_i is complex then the left eigenvector for λ_i and right eigenvector for λ_i^* are related by γ_5. If λ_i is real then the corresponding left and right eigenvectors are related by γ_5.

If we build a multigrid subspace that preserves the action of γ_5 then... WHAT IS BETTER! (There's been debate between Luscher and US MG folks about γ_5 compatible coarsening being better – read up). (Evan alluded to instanton preservation in his talk)

The **singular value decomposition** of D_W is,

$$D_W = U\Sigma V^\dagger,$$

where there is a remnant phase abiguity between rows of U and V.

The singular value decomposition of the Hermitian Wilson operator $\gamma_5 D_W$ is almost identical to its eigen-decomposition, and is simply related,

$$H_W = V\tilde{\Sigma}V^\dagger = (\gamma_5 U)\Sigma V^\dagger,$$

and since H_W is Hermitian we see that $\gamma_5 U = \text{sgn}(\tilde{\Sigma})V$, where the sign of the (indefinite) eigenvalues of H_W relates the positive valued Σ from the SVD of D_W to the signed eigenvalues of H_W.

For the squared Dirac operator, these signs are removed and manifest positivity results

$$D_W^\dagger D_W = H_W^2 = V(\Sigma)^2 V^\dagger,$$

$$D_W D_W^\dagger = \gamma_5 H_W^2 \gamma_5 = U(\Sigma)^2 U^\dagger.$$

The singular values and absolute values of the eigenvalues of H_W are necessarily the same as the singular values of D_W (but differ from the eigenvalues of D_W). They are trivially related to the singular values of the squared operator.

1.3.6 Near null vectors

With (hypothetical) exact null vectors $D_W\psi = 0 = \gamma_5 D_W\psi = D_W^\dagger D_W\psi$, we expect:

- D_W has an exact zero right eigenmode, $\psi \in R$ (where "in" means is a column of)
- $\gamma_5\psi$ is the corresponding left zero eigenmode
- The SVD of D_W *also* has a corresponding exactly zero singular value with $\psi \in V$.
- $|\gamma_5\psi\rangle(\sigma = 0)\langle\psi|$ is the corresponding null element of the SVD expansion with zero singular value (notation abuse).

This equivalence arises only algebraically for exact zero modes. The question arises whether near null singular values and singular vectors can be used to set up a non-Hermitian multigrid.

This may "obviously" be the case for an SVD based multigrid.

If we generate a space in the "V" space and make a γ_5 compatible coarsening (doubling the space to include both V and $\gamma_5 V$, then the "U")

1.3.7 Singular value decomposition deflation and multigrid

A second approach to deflation, and potentially a novel approach to multigrid in the non-hermitian case, is based on the singular value decomposition:

$$D = U\Sigma V^\dagger,$$

where $\Sigma = \text{diag}(\sigma_i)$ is diagonal real, positive and U and V are unitary. The diagonal elements are the positive square roots of the eigenvalues of the squared operator.

$$D^\dagger D = V\text{diag}(|\sigma_i|^2)V^\dagger$$

$$DD^\dagger = U\text{diag}(|\sigma_i|^2)U^\dagger$$

These may be solved by using Lanczos to find the eigenpairs of the hermitian positive definite problem for either $D^\dagger D$ or $DD^\dagger$. Once V and Σ have been found, U may be constructed via the application of D,

$$DV = U\Sigma,$$

though in practice with approximate eigenvectors a Lanczos restart my be preferable to retain convergence precision. Within a truncated subspace we can solve for a deflated guess via

$$U\Sigma V^\dagger x = r_0$$

with

$$x = \sum_i |v_i\rangle \frac{\langle u_i|r_0\rangle}{\sigma_i}.$$

This suggests a potential new class of lattice multigrid methods based on the singular value decomposition, rather than right eigenvectors.

The restriction operator can be based on a block projection of the V vectors and prolongation on U. The matrix elements can be computed

$$C^{ii'}_{bb'} = U_b^\dagger D V_{b'}$$

The filtering task of finding V (and U) is over the real domain of singular values rather than a complex spectrum, and may allow better setup methods than inverse iteration.

does this have overlap with the staggered MG coarse grid approaches? Saw reference to SVD deflation in Evan's talk at CERN (I previously wrote this here)

1.3.8 Mathworks example matrix

We know that the real spectrum of H_W and complex spectrum of D_W differ massively, as do their eigenvectors. The exact zeros must coincide, however.

c.f. Saul Cohen's animation of $\Gamma_5^\epsilon D_{dwf}$ for *epsilon* $\in [0,1]$

Useful document by CTO of Mathworks (Matlab) Cleve Moler:

https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/eigs.pdf

$$A = \begin{pmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{pmatrix} = X\Lambda X^{-1}$$

Eigenvalues

$$\Lambda = \mathrm{diag}(1,2,3)$$

Singular values

$$\Sigma = \mathrm{diag}(817.7597, 2.4750, 0.0030)$$

Eigenvectors

$$X = \begin{pmatrix} 1 & -4 & 7 \\ -3 & 9 & -49 \\ 0 & 1 & 9 \end{pmatrix}$$

$$X^{-1} = \begin{pmatrix} 130 & 43 & 133 \\ 27 & 9 & 28 \\ -3 & -1 & -3 \end{pmatrix}$$

Peturbation analysis on matrix; introduces *eigenvalue condition number* for left (y) and right (x) eigenvectors

$$1 \le \kappa(\lambda, A) = \frac{||y_\lambda|| ||x_\lambda||}{\langle y_\lambda | x_\lambda\rangle} = \frac{1}{\cos\theta_{xy}}$$

This factor is always 1 for Hermitian matrices, and amplifies the numerical uncertainty on the eigenvalue for increasingle non-Hermitian

1.3.9 $PV^\dagger M$ deflation

Use variable preconditioned GCR. Equivalent to Flexible GMRES up to rounding. Minimises the residual norm, difference is in orthonormal basis vs. $A^\dagger A$ orthogonal basis of search directions. (Saad's paper on GMRES says "theoretically equivalent").

Singular vector setup

- $P^\dagger M = U\Sigma V^\dagger$
- First attempted Chebyshev set up in $M^\dagger P P^\dagger M = V\Sigma^\dagger \Sigma V^\dagger$
- Produces a polynomial of singular values and vectors
- Measure "nulliness" of singular vector (squared operator): $\langle v|M^\dagger P^\dagger M|v\rangle \in \mathbb{C}$ has magnitude around 0.3
- Measure "nulliness" of wrt. eigenvector subspace: $\langle v|P^{dag}M|v\rangle \in \mathbb{C}$ has magnitude around 0.2
  - U / V space misalignment might have made this worse!
- Chebyshev parameters lowpass of order 0.1, filter orders 400.. 1800
- Coarse solver, smoother (low pass GCR 16 iters, shift 1.0
- Solver converged, but deflation effect negligible - around 350 outer iterations
- Will revisit singular vector space setup

Eigenvector setup

- Produce vectors by inverse iteration 2/3/4 steps with sloppy solves
- $P^\dagger M = R\Lambda R^{-1}$
- Measure "nulliness" of wrt. eigenvector subspace: $\langle v|P^{dag}M|v\rangle \in \mathbb{C}$ has magnitude around 0.008
- Solver converges with real deflation effect around 50 outer iterations
- Playing with number of vectors (8 - 24), block size solver parameters (4^4, 2^4)

1.4 Optimised implementation

Following sections are implementation notes for optimisation and benchmarking of a matrix free implementation to usual standards in the field.

Spin projection optimisation

https://github.com/paboyle/Grid/blob/develop/Grid/qcd/spin/TwoSpinor.h

The matrix $P^\pm_\mu = \frac{1}{2}(1 \pm \gamma_\mu)_{\alpha\beta}$ is a projector,

$$(P^\pm_\mu)^2 = \frac{1}{4}(1 \pm \gamma_\mu)(1 \pm \gamma_\mu) = \frac{1}{2}(1 \pm \gamma_\mu) = P^\pm_\mu$$

.

Since $P^+_\mu + P^-_\mu = 1$ the action of P^+ and P^- is to project a four spinor into only two linearly independent two component spinors. This may be used below to cut the arithmetic and MPI bandwidth cost of our operator in half.

Observing that $\gamma_{x...t}$ are off-diagonal, with structure

$$\gamma_{xyzt} = \begin{pmatrix} 0 & \sigma_{xyzt} \\ \bar{\sigma}_{xyzt} & 0 \end{pmatrix}$$

The spin projectors are implemented with inline functions acting on a four-spinor f and producing a two spinor h, observing that the rows of $(1 \pm \gamma_\mu)f$ are not all linearly independent. These are inline routines such as `spProjXp` in Grid. The routines spProj5p and spProj5m use a different normalisation to avoid a factor of two since this removes arithmetic.

```
spProjXp (+) / spProjXm (-)
h[0] = f[0]+-i f[3]
h[1] = f[1]+-i f[2]
h[2] = f[2]-+i f[1] = -+i h[1]
h[3] = f[3]-+i f[0] = -+i h[0]

spProjYp (+) / spProjYm (-)
h[0]=f[0] -+ f[3]
h[1]=f[1] +- f[2]
h[2]=f[2] +- f[3] = h[1]
h[3]=f[3] -+ f[0] =-h[0]

spProjZp (+) / spProjZm (-)
h[0]=f[0]+-i f[2]
h[1]=f[1]-+i f[3]
h[2]=f[2]-+i f[0] =-+i h[0]
h[3]=f[3]+-i f[1] =+-i h[1]

spProjTp (+) / spProjTm (-)
h[0]=f[0]+-f[2]
h[1]=f[1]+-f[3]
h[2]=f[2]+-f[0] =+-i h[0]
h[3]=f[3]+-f[1] =+-i h[1]

spProj5p
h[0]=f[0]
h[1]=f[1]

spProj5m
h[0]=f[2]
h[1]=f[3]
```

Since rows 2 and 3 are linearly dependent on rows 1 and 2 the spin projection can be performed prior to the 3×3 color matrix multiplication in the Wilson operator, meaning the SU3 multiplies act on two component spinors rather than four component spinors, saving a factor of two in floating point arithmetic.

Floating point operation counts

Operation	Madd	Mul	Add	Flops
Complex mul	2	2	0	6
Complex madd	4	0	0	8
SpinProj	0	0	12	12
SU3x2spin	3 . 2 . (6+8+8)			132
Accumulate	0	0	24	24
Wilson	8 . (132+12)+7 . 24			**1320**

Arithmetic intensity and performance

Arithmetic intensity determination for single RHS Wilson matrix and multiRHS wilson matrix. The SU3 field access are reused for many RHS and may be dropped in large N limit.

Operation	Level	Words	Bytes (fp32)
Neigbour Spinor	memory	24 . 1	96
Neigbour Spinor	cache	24 . (8-1)	672
SU3	memory	18 . 8	576
Result	memory store	24	96
Memory			768
Cache+memory			1440
Spinor only memory			192
Spinor only cache+mem			864
Wilson bytes/flop	cache	1.09 = 1440/1320	
Wilson bytes/flop	mem	0.58 = 768/1320	
WilsonRHS bytes/flop	cache	0.65 = 864/1320	
WilsonRHS bytes/flop	mem	0.145 = 192/1320	

A well optimised code should obtain the lower of

- Memory bytes/second times memory flops/byte
- Cache bytes/second times cache flops/byte

The WilsonRHS byte count is relevant for Domain Wall Fermions (next section).

A key implementation feature is the *replication* of the same 4d color matrix field is distributed across multiple slices in a five dimensional Fermion field.

Implicitly in the above byte counting the fact that the same color matrix is distributed across all four (two) spin spin components.

Replicating duplicate matrix coefficients with multiple copies is not desirable in terms of performance or memory footprint.

1.4.1 Domain Wall Fermions

$$S^5 = \bar\psi D^5_{DW} \psi \tag{1.30}$$

where:

$$D^5_{DW} = 5 - M_5 - \frac{1}{2} D_{\rm hop} - \frac{1}{2} D_\perp = D_\parallel \delta_{s,s'} - \frac{1}{2} D_\perp \delta_{xx'} \tag{1.31}$$

where

$$D_\perp = P_- \delta_{s+1,s'} + P_+ \delta_{s-1,s'}$$

and

$$D_\parallel = 5 - M_5 - \frac{1}{2} D_{\rm hop} = D_W(-M_5) + 1.$$

and being a bit sloppy and ignoring the mass terms / boundary

$$\frac{1}{2} D_\perp \sim P_- \delta_{s+1,s'} + P_+ \delta_{s-1,s'}$$

$$D^5_{DW} = \begin{pmatrix} D_\parallel & -P_- & 0 & \cdots & 0 & mP_+ \\ -P_+ & \ddots & \ddots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \ddots & \ddots & -P_- \\ mP_- & 0 & \cdots & 0 & -P_+ & D_\parallel \end{pmatrix} \tag{1.32}$$

Even odd preconditioning

For many years single level solvers in QCD have used various forms of red-black or even-odd preconditioning. The 4d space time is divided into red and black sites, based on $(x+y+z+t)|2 \in \{0,1\}$

It may be important to be able to support vectors that contain only half the lattice sites, and not propagate zeroes.

In five dimensional meshes, we must be able to choose whether or not the fifth dimension participates in the checkboarding.

i.e. checkerboard is either $(x+y+z+t+s)|2 \in \{0,1\}$ (five dim red-black) OR $(x+y+z+t)|2 \in \{0,1\}$ (four dim red-black).

Given a red-black definition, any matrix M may be decomposed as

$$\begin{aligned} M &= \begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} && (1.33) \\ &= \begin{pmatrix} 1 & 0 \\ M_{oe}M_{ee}^{-1} & 1 \end{pmatrix} \begin{pmatrix} M_{ee} & M_{eo} \\ 0 & M_{oo} - M_{oe}M_{ee}^{-1}M_{eo} \end{pmatrix} \begin{pmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & M_{oo} \end{pmatrix} && (1.34) \\ &= LDU && (1.35) \end{aligned}$$

Observe:

$$L^{-1} = \begin{pmatrix} 1 & 0 \\ -M_{oe}M_{ee}^{-1} & 1 \end{pmatrix}$$

$$U^{-1} = \begin{pmatrix} 1 & -M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{pmatrix}$$

We may split $M\psi = \eta$ into odd and even systems via this Schur decomposition as follows.

Even-Odd Schur solvers: Take $D_{oo} = M_{oo} - M_{oe}M_{ee}^{-1}M_{eo}$ and

$$D_{oo}\psi_o = (L^{-1}\eta)_o = \eta_o'$$

where

$$\eta_o' = (\eta_o - M_{oe}M_{ee}^{-1}\eta_e)$$

If we use the normal equations in the red-black system then, Wilson CGNR:

$$(D_{oo})^\dagger D_{oo}\psi_o = (D_{oo})^\dagger (L^{-1}\eta)_o$$

For staggered fermions the operator is Hermitian and we solve

$$D_{oo}\psi_o = L^{-1}\eta = (\eta_o - M_{oe}M_{ee}^{-1}\eta_e)$$

The even checker solution can be reconstructed from the odd solution:

$$M_{ee}\psi_e + M_{eo}\psi_o = \eta_e \Rightarrow \psi_e = M_{ee}^{-1}(\eta_e - M_{eo}\psi_o)$$

1.5 Coarsening gauge theories

Coarsening a field with gauge degrees of freedom is more complicated than coarsening a simple scalar field because the coarsening must be compatible with gauge invariance. Suppose $\psi(n)$, $n \in \Lambda$ is a fermion field with gauge group G ($G = SU(3)$ for QCD). We wish to coarsen $\psi(x)$ to $\tilde{\psi}(\tilde{n})$, with $\tilde{n} \in \tilde{\Lambda} \subset \Lambda$ a sublattice, by averaging over blocks $\mathcal{B} = \{B_{\tilde{n}}\}_{\tilde{n}\in\tilde{\Lambda}} \subset 2^{\Lambda}$ ($\mathcal{B}$ partitions Λ; typically each $B_{\tilde{n}}$ is a hypercubic block containing $\tilde{n}$). Averaging over blocks requires the connection $U_\mu(n)$, since if the gauge is not fixed then one cannot add together the field at different sites on the lattice (because in that case you could rotate the field freely at one point in the sum and change the answer in a non-covariant way).

Let $\tilde{n} \in \tilde{\Lambda}$. To average over the block $\mathcal{B}_{\tilde{n}}$, we parallel transport the field at each point in the block to $\tilde{n}$; $\tilde{n}$ is our reference point for the block. Recall the Wilson line $W(n, m)$ transforms under gauge transformations $\Omega(n)$ as $W(n, m) \mapsto \Omega(n)W(n, m)\Omega^\dagger(m)$ and performs parallel transport of the fermion field from m to n. The Wilson line is constructed from the gauge field $U_\mu(n)$ as a product over a path C connecting n to m,

$$W(n, m) = \prod_{(n',\mu)\in C} U_\mu(n'). \tag{1.36}$$

The coarse gauge field $\tilde{\psi}(\tilde{n})$ is then the average of the fine gauge field $\psi(n)$ with the correct parallel transport,

$$\tilde{\psi}(\tilde{n}) = \frac{1}{|\mathcal{B}_{\tilde{n}}|} \sum_{m\in\mathcal{B}_{\tilde{n}}} W(\tilde{n}, m)\psi(m). \tag{1.37}$$

If the theory is gauge fixed, the expression for the block average is simpler. Suppose that one fixes a gauge, implemented with a gauge transformation $g(x) \in G$. As previously, one desires the field in each block to transform homogeneously, which is implemented by $g(n)g^\dagger(m)$. The block average is then,

$$\tilde{\psi}_g(\tilde{n}) = \frac{1}{|\mathcal{B}_{\tilde{n}}|} \sum_{m\in\mathcal{B}_{\tilde{n}}} g(\tilde{n})g^\dagger(m)\psi(m). \tag{1.38}$$

A cartoon depiction of this setup is shown in Fig. 1.1.

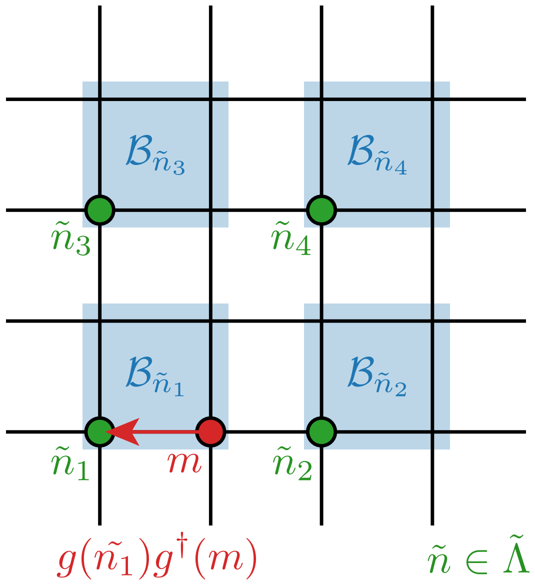
$\mathcal{B}_{\tilde{n}_3}$
$\mathcal{B}_{\tilde{n}_4}$
$\tilde{n}_3$
$\tilde{n}_4$
$\mathcal{B}_{\tilde{n}_1}$
$\mathcal{B}_{\tilde{n}_2}$
$\tilde{n}_1$
m
$\tilde{n}_2$
$g(\tilde{n}_1)g^\dagger(m)$
$\tilde{n} \in \tilde{\Lambda}$

Figure 1.1: Cartoon depiction for coarsening a gauge theory. The fine lattice is denoted by the black lines, and the coarse lattice points $\tilde{n} \in \tilde{\Lambda}$ are shown in green. Four blocks $\mathcal{B}_{\tilde{n}}$ are shown in this cartoon, each of which is shaded in blue. Averaging over each block is performed by parallel transporting each point m in the block to its reference point $\tilde{n}$ with $g(\tilde{n})g^\dagger(m)\psi(m)$, which is shown in red.

Prolongation from the coarse field $\tilde{\psi}(\tilde{n})$ to the fine field $\psi(n)$ can be defined in a similar way: one can use standard MG prolongation stencils and techniques, as long as the prolongated field transforms covariantly under gauge rotations on the fine lattice Λ. Suppose that $m \in \Lambda$ is contained in block $\mathcal{B}_{\tilde{n}}$. As a simple example, a piecewise-constant prolongation over each coarse-grid block has a fine field $\psi(m)$ taking the form

$$\psi(m) = g(m)g^\dagger(\tilde{n})\tilde{\psi}(\tilde{n}). \tag{1.39}$$

In this case, one evaluates $\psi(m)$ by with the value of the coarse field at $\tilde{n}$, and then parallel transports the value appropriately to transform under gauge transformations at m as it must.

Any standard prolongation operator can be defined respecting gauge invariance as long as the fields are correctly parallel transported with $g(n)g^\dagger(m)$ before being added together. For a general stencil $s(n,m)$ that defines the prolongation operator[1], the stencil can correctly take into account gauge invariance by being augmented with the correct gauge transformation,

$$\psi(m) = \sum_{n\in\Lambda} s(m,n)g(m)g^\dagger(n)\tilde{\psi}(n). \tag{1.41}$$

Here the coarse field $\tilde{\psi}(n)$ is defined on the fine lattice $n \in \Lambda$ with the canonical embedding,

$$\tilde{\psi}(n) = \begin{cases} \tilde{\psi}(n) & n \in \tilde{\Lambda} \\ 0 & n \notin \tilde{\Lambda} \end{cases}. \tag{1.42}$$

In the case where the gauge fixing matrix $g(n)$ is unity, or when we do not have to worry about a gauge field, this should reduce to the usual interpolation operator from the stencil $s(n,m)$.

[1]For example, on a 2D grid, the stencil

$$s(n,m) \to \frac{1}{4}\left]\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}\right[\tag{1.40}$$

that implements a 2D linear interpolation.

Chapter 2

General Cayley form

Now consider a generalisation of the action that includes both Tanh and Zolotarev approximations to Moebius fermions [24, 25] which have Shamir and Neuberger kernels as limiting cases as shown by Borici[16, 17].

$$S^5 = \bar{\psi} D^5_{GDW} \psi \tag{2.1}$$

We define

$$\begin{aligned}
D^s_+ &= \alpha_s(b_s D_W + 1) && (2.2)\\
D^s_- &= \alpha_s(1 - c_s D_W) && (2.3)\\
\tilde{D}^s &= (D^s_-)^{-1} D^s_+ && (2.4)
\end{aligned}$$

and take

$$\begin{aligned}
D^5_{GDW} &= (D_-)^{-1} D^5_{Brower} && (2.5)\\
&= \big((D^s_-)^{-1} D^s_+ \delta_{s,s'} - P_- \delta_{s,s'-1} - P_+ \delta_{s,s'+1}\big) && (2.6)\\
&= \begin{pmatrix}
(D^1_-)^{-1}D^1_+ & -P_- & 0 & \dots & 0 & mP_+ \\
-P_+ & \ddots & \ddots & 0 & \dots & 0 \\
0 & \ddots & \ddots & \ddots & 0 & \vdots \\
\vdots & 0 & \ddots & \ddots & \ddots & 0 \\
0 & \dots & 0 & \ddots & \ddots & -P_- \\
mP_- & 0 & \dots & 0 & -P_+ & (D^{L_s}_-)^{-1}D^{L_s}_+
\end{pmatrix} && (2.7)\\
&= \begin{pmatrix}
\tilde{D}^1 & -P_- & 0 & \dots & 0 & mP_+ \\
-P_+ & \ddots & \ddots & 0 & \dots & 0 \\
0 & \ddots & \ddots & \ddots & 0 & \vdots \\
\vdots & 0 & \ddots & \ddots & \ddots & 0 \\
0 & \dots & 0 & \ddots & \ddots & -P_- \\
mP_- & 0 & \dots & 0 & -P_+ & \tilde{D}^{L_s}
\end{pmatrix} && (2.8)
\end{aligned}$$

Standard Domain Wall Fermions correspond to $\alpha_s = 1; b = 1; c = 0$

For a tanh approximation b_s, c_s are constant and the matrix remains translationally invariant in the fifth dimension (anti-periodic).

Bibliography

[1] D. B. Kaplan, "A Method for simulating chiral fermions on the lattice," Phys. Lett. B **288** (1992) 342 [hep-lat/9206013].

[2] Y. Shamir, "Chiral fermions from lattice boundaries," Nucl. Phys. B **406** (1993) 90 [hep-lat/9303005].

[3] V. Furman and Y. Shamir, "Axial symmetries in lattice QCD with Kaplan fermions," Nucl. Phys. B **439** (1995) 54 [hep-lat/9405004].

[4] T. Blum and A. Soni, "QCD with domain wall quarks," Phys. Rev. D **56** (1997) 174 [hep-lat/9611030].

[5] R. Narayanan and H. Neuberger, "Chiral determinant as an overlap of two vacua," Nucl. Phys. B **412** (1994) 574 [hep-lat/9307006].

[6] R. Narayanan and H. Neuberger, "Chiral fermions on the lattice," Phys. Rev. Lett. **71** (1993) 3251 [hep-lat/9308011].

[7] R. Narayanan and H. Neuberger, "A Construction of lattice chiral gauge theories," Nucl. Phys. B **443** (1995) 305 [hep-th/9411108].

[8] H. Neuberger, "Exactly massless quarks on the lattice," Phys. Lett. B **417** (1998) 141 [hep-lat/9707022].

[9] H. Neuberger, "Vector - like gauge theories with almost massless fermions on the lattice," Phys. Rev. D **57** (1998) 5417 [arXiv:hep-lat/9710089].

[10] Y. Kikukawa and T. Noguchi, "Low-energy effective action of domain wall fermion and the Ginsparg-Wilson relation," hep-lat/9902022.

[11] R. G. Edwards and U. M. Heller, "Domain wall fermions with exact chiral symmetry," Phys. Rev. D **63** (2001) 094505 [hep-lat/0005002].

[12] H. Neuberger, "More about exactly massless quarks on the lattice," Phys. Lett. B **427** (1998) 353 [hep-lat/9801031].

[13] M. Luscher, "Exact chiral symmetry on the lattice and the Ginsparg-Wilson relation," Phys. Lett. B **428** (1998) 342 [hep-lat/9802011].

[14] Y. Kikukawa and A. Yamada, "Axial vector current of exact chiral symmetry on the lattice," Nucl. Phys. B **547** (1999) 413 [hep-lat/9808026].

[15] Y. Kikukawa and A. Yamada, "Weak coupling expansion of massless QCD with a Ginsparg-Wilson fermion and axial U(1) anomaly," Phys. Lett. B **448** (1999) 265 [arXiv:hep-lat/9806013].

[16] A. Borici, "Truncated overlap fermions," Nucl. Phys. Proc. Suppl. **83** (2000) 771 [hep-lat/9909057].

[17] A. Borici, "Truncated overlap fermions: The Link between overlap and domain wall fermions," hep-lat/9912040.

[18] A. Borici, "Computational methods for the fermion determinant and the link between overlap and domain wall fermions," hep-lat/0402035.

[19] A. Borici, A. D. Kennedy, B. J. Pendleton and U. Wenger, "The Overlap operator as a continued fraction," Nucl. Phys. Proc. Suppl. **106** (2002) 757 [hep-lat/0110070].

[20] R. G. Edwards, B. Joo, A. D. Kennedy, K. Orginos and U. Wenger, "Comparison of chiral fermion methods," PoS LAT **2005** (2006) 146 [hep-lat/0510086].

[21] R. G. Edwards *et al.* [SciDAC and LHPC and UKQCD Collaborations], "The Chroma software system for lattice QCD," Nucl. Phys. Proc. Suppl. **140** (2005) 832 [hep-lat/0409003].

[22] A. D. Kennedy, "Algorithms for dynamical fermions," hep-lat/0607038.

[23] D. J. Antonio *et al.* [RBC and UKQCD Collaborations], "Localization and chiral symmetry in three flavor domain wall QCD," Phys. Rev. D **77** (2008) 014509 [arXiv:0705.2340 [hep-lat]].

[24] R. C. Brower, H. Neff and K. Orginos, "Mobius fermions: Improved domain wall chiral fermions," Nucl. Phys. Proc. Suppl. **140** (2005) 686 [hep-lat/0409118].

[25] R. C. Brower, H. Neff and K. Orginos, "Mobius fermions," Nucl. Phys. Proc. Suppl. **153** (2006) 191 [hep-lat/0511031].

[26] R. Narayanan and H. Neuberger, "An Alternative to domain wall fermions," Phys. Rev. D **62** (2000) 074504 [hep-lat/0005004].

[27] Peter A. Boyle "The BAGEL assembler generation library" Computer Physics Communications, Volume 180, Issue 12, December 2009, Pages 2739-2748