Computational Methods (practice) - Lecture 3

Peter Boyle (BNL, Edinburgh)

- Krylov methods and approximate matrix inversion
- GMRES
- Conjugate Gradients
- Preconditioning
- Red black preconditioning
- Checkerboarded implementation
- Eigensolvers and Deflation
- Multigrid preconditioning

Krylov methods and approximate matrix inversion

- Algorithms minimise a residual $|r|^2$, where

$$r = M\psi - b$$

  - $r = 0 \Leftrightarrow \psi = M^{-1}b$ so minimise r under some norm
- *Krylov space* is the span of all polynomials of M and of O(N) $\mathrm{sp}b, Mb, \ldots M^N b$
- *Krylov solvers* iteratively apply a (sparse) matrix to build up this space
- Unlike Chebyshev approximation these algorithms require no prior knowledge of the spectral range of M
- Different algorithms invoke different rules for selecting these coefficients...
  ...and have different storage requirements

Lanczos/Arnoldi orthogonal sequence

http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter10.pdf
The Lanczos and conjugate gradient algorithms, Meurant

- Krylov space $K_n(b) = \{b, Ab, \ldots, A^n b\}$
- Seek orthonormal basis: normalise components perpendicular to all prior vectors
- $\beta_{j+1}|v_{j+1}\rangle = (1 - \sum_{i=1}^{j} |v_i\rangle\langle v_i|)|Av_j\rangle$
- Rewrite as:

$$|Av_j\rangle = \sum_{i=1}^{j+1} |v_i\rangle H_{ij}$$

  Where $H_{ij} = \langle v_i|A|v_j\rangle$ and is zero for $i > j + 1$ by virtue of our sequential orthogonalisation.
- The "Householder matrix" H is tridiagonal when A has Hermitian symmetry and an orthonormal basis for the Krylov space is mapped out with a three term recurrence relation.
- **Removes large storage requirements**
- Can use this basis to build solutions $\Rightarrow$ Conjugate Gradients & GMRES

GMRES

https://github.com/paboyle/Grid/blob/develop/Grid/algorithms/iterative/GeneralisedMinimalResidual.h

- Consider the orthonormal basis for the Krylov space: $x_n = \sum_{j=1}^{n} c_j |v_j\rangle$
- GMRES minimises the Euclidean norm $|r|^2$ with respect to the coefficients c_n where $|r\rangle = |b\rangle - c_j |Av_j\rangle$
- We know how to represent the matrix in this basis

$$|Av_j\rangle = \sum_{i=1}^{j+1} |v_i\rangle H_{ij}$$

- So this reduces to minimising $|\beta e_1 - Hc|$, where $\beta = |b|$
- Dense linear algebra (QR factorisation) in H yields the minimisation
- This requires all N vectors to be retained, and coefficients selected *after* n-iterations
- GMRES(k) runs for k iterations and then *restarts* required to limit storage
- Often used in a preconditioner/smoother

https://en.wikipedia.org/wiki/Generalized_minimal_residual_method

Conjugate Gradients

https://en.wikipedia.org/wiki/Conjugate_gradient_method

- Generate A-orthogonal sequence of search directions based on Lanczos sequence
- Residuals are parallel to the Lanczos basis, mutually orthogonal set
- Krylov solution to $Ax = b$ has $x = \sum_k \alpha_k p_k$:

$$p_j^\dagger A x = p_j^\dagger b = \sum_k \alpha_k p_j^\dagger A p_k = \alpha_j p_j^\dagger A p_j \Rightarrow \alpha_j = \frac{p_j^\dagger b}{p_j^\dagger A p_j}$$

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
  $\alpha_k := \dfrac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
  $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
  if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
  $\beta_k := \dfrac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
  $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
  $k := k + 1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

```
void operator()(LinearOperatorBase<Field> &Linop, const Field &src, Field &psi)
{
  RealD cp, c, a, d, b, ssq, qq;

  Field p(src), mmp(src), r(src);

  Linop.HermOpAndNorm(psi, mmp, d, b);

  r = src - mmp;
  p = r;

  a = norm2(p);
  cp = a;
  ssq = norm2(src);

  RealD rsq = Tolerance * Tolerance * ssq;

  for (int k = 1; k <= MaxIterations; k++) {
    c = cp;

    Linop.HermOp(p, mmp);

    ComplexD dc  = innerProduct(p,mmp);
    d = dc.real();
    a = c / d;

    cp = axpy_norm(r, -a, mmp, r);
    b = cp / c;

    psi = psi + a* p ;
    p   = r   + b* p ;

    if (cp <= rsq) {
      return;
    }
  }
  assert(0);
}
```

BiCGstab

For Wilson Fermions, BICGstab is the fastest conventional Krylov solver It is suited to solving the non-Hermitian system

$$D_w \psi = b$$

https://github.com/paboyle/Grid/blob/develop/Grid/algorithms/iterative/BiCGSTAB.h
https://en.wikipedia.org/wiki/Biconjugate_gradient_stabilized_method

Convergence rate, critical slowing down, and preconditioning

- The uniformity of the Chebyshev polynomial oscillations can be used to bound convergence rate via a maximum error over the spectral range
  - Krylov solvers can do better than this worst case bound as polynomial coefficients are selected based on the actual spectrum
  - QCD often (index theorem) has a detached number of low modes (topological nature)
- Condition numer

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$$

- Theoretical convergence factor per iteration (c.f. R-number in epidemiology!)

$$\sigma = \frac{\sqrt{k} - 1}{\sqrt{k} + 1}$$

- In infinite volume limit spectrum is dense and the worst case is the guaranteed case
- Empirically, the theoretical σ governs the long tail convergence of CG in practice
- (left) Preconditioning: changing κ by solving a related system

$$PM\psi = Pb$$

- If the condition number of PM is substantially reduced, preconditioned system converges faster

Convergence rate, critical slowing down, and preconditioning

- If the condition number of PM is substantially reduced, preconditioned system converges faster
- Ideally P is a cheap-to-apply *approximate inverse* of M
- Left preconditioning
$$PM\psi = Pb$$
- Right preconditioning
$$MP\psi' = b; \psi = P\psi'$$
- Approximating M^{-1} can be focussed regions of spectrum
- Lower $\lambda_{\max}$
  - Polynomial preconditioner (e.g. Chebyshev $1/x$ over high end of spectrum); reduce rate of inner-products / reductions
  - Domain decomposed smoother such Schwarz Alternating procedure : works for high end of spectrum; reduces communication and rate of inner-products / reductions
- Raise $\lambda_{\min}$
  - Deflation of low modes $P = (1 - \sum |i\rangle\langle i|) + \sum_i \frac{|i\rangle\langle i|}{\lambda_i}$
  - Up to rounding, deflation can be applied infrequently in CG due to orthogonal search sequence

Schur decomposition

https://github.com/paboyle/Grid/blob/develop/Grid/algorithms/iterative/SchurRedBlack.h
A matrix can be LDU factorised as follows. Each of A, B, C or D can themselves be sub-matrices

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ CA^{-1} & 1 \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & D - CA^{-1}B \end{pmatrix} \begin{pmatrix} 1 & A^{-1}B \\ 0 & 1 \end{pmatrix}, \tag{1}$$

where the Schur complement,

$$S = D - CA^{-1}B.$$

Red-Black preconditioning

We can write the Dirac operator in terms of even and odd lattice sites and perform an LDU decomposition:

$$M = \begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ M_{oe}M_{ee}^{-1} & 1 \end{pmatrix} \begin{pmatrix} M_{ee} & 0 \\ 0 & M_{oo} - M_{oe}M_{ee}^{-1}M_{eo} \end{pmatrix} \begin{pmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{pmatrix}, \tag{2}$$

where the Schur complement, is written as $M_{pc} = M_{oo} - M_{oe}M_{ee}^{-1}M_{eo}$.

- For Wilson Fermions the M_{ee} is proportional to the identity.
- For DWF and Wilson Clover Fermions the terms are non-trivial.
- For the Wilson Clover term M_{ee} depends on the gauge fields.
- For DWF M_{ee} is independent of the gauge fields.

U and L have determinant 1 and are trivially invertible:

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

For the odd checkerboard, $M\psi = \eta$ becomes

$$M_{pc}\psi_o = \eta'_o = (L^{-1}\eta)_o = \eta_o - M_{oe}M_{ee}^{-1}\eta_e$$

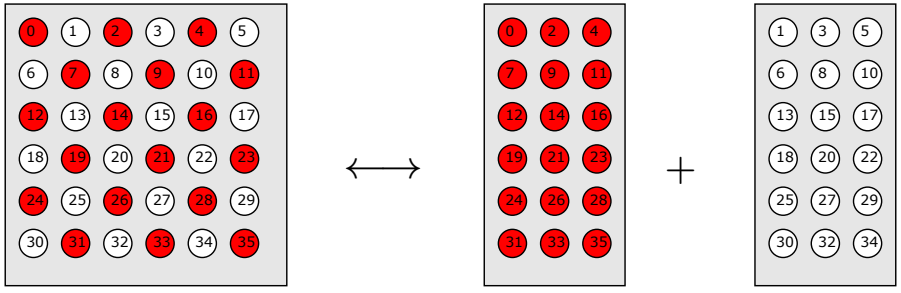
The even checkerboard solution can be inferred via

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

M_{pc} (empirically) better conditioned than M: red black solvers converge O(3x) faster

Checkerboarding

- Lattice QCD makes use of red-black preconditioning in many algorithms
- Support for checkerboarded grids is required
  - e.g. a field that lives only on the white or black sites of a chessboard
  - Shifting a "black" field by one site produces a white field and vice versa
    Indexing and neighbour indexing is complicated by this
  - Stencil operators work with checkerboarded grids



- Checkerboarded Grid objects can have arbitrary subset of dimensions involved in checkerboarding
- Dimension "collapsed" can be selected (typically x-direction)
- Natural support for 4d and 5d checkerboarded chiral fermions
  - Neighbour indexing is integer heavy divide/modulo arithmetic
  - Precompute neighbour tables in high performance Stencil objects
  - Calculate dynamically for Cshift, looping over planes

Eigensolvers

- The Lanczos sequence can also be used to solve for eigenvectors, known as the Lanczos algorithm.
- For a Hermitian matrix the Householder matrix is tridiagonal and represents the matrix A within the Lanczos basis

$$H_{ij} = T_{ij} = \langle v_i | A | v_j \rangle$$

- Diagonalising this (small) tridiagonal matrix tells us the basis rotation that maximally diagonalises the matrix A
- Keeping the lowest N and repeating leads to restarted Lanczos
- Grid has a Chebyshev polynomial preconditioned Lanczos algorithm:

https://github.com/paboyle/Grid/blob/develop/Grid/algorithms/iterative/ImplicitlyRestartedLanczos.h

- Lanczos is used to produce the lowest lying eigenvectors of the Dirac operator
- Polynomial preconditioning: chebyshev polynomials suppress unwanted part of spectrum (univariate wiggles) - diverges outside this region.
- Compute eigenvectors of the polynomial, map back to eigenvalues of the original matrix.
- Why? Low lying eigenvectors can now be handled exactly: remove from the problem to eliminate critical slowing down
- Deflation of low modes $G = \sum_i \frac{|i\rangle \langle i|}{\lambda_i}$ can be used as a guess.
  - These will not reenter the Krylov space other than through:
  - a) Rounding errors
  - b) Imprecision in the eigenvectors/eigenvalues

Exercise

- Write a conjugate gradients algorithm to invert the (massive) Laplacian
  - Introduce a small mass m^2 to the Laplacian example in Lecture 1 to regulate the spectrum.
- Example Solution: *strongly* suggest you write your own solution.
  https://github.com/paboyle/Grid/blob/develop/examples/Example_Laplacian_solver.cc
- Extension:
  - Verify the results on the free field via Fourier methods
  - Check gauge covariance

Multigrid preconditioners

Multigrid is introduced as a *Preconditioner*
Low mode subspace vectors ϕ generated in some way: tried

- Inverse iteration (c.f. Luscher)
- Lanczos vectors
- Chebyshev filters

$$\phi_k^b(x) = \begin{cases} \phi_k(x) & ; \quad x \in b \\ 0 & ; \quad x \notin b \end{cases}$$

$$\mathrm{span}\{\phi_k\} \subset \mathrm{span}\{\phi_k^b\}.$$

$$P_S = \sum_{k,b} |\phi_k^b\rangle\langle\phi_k^b| \qquad ; \qquad P_{\bar{S}} = 1 - P_S$$

$$M = \begin{pmatrix} M_{\bar{S}\bar{S}} & M_{S\bar{S}} \\ M_{\bar{S}S} & M_{SS} \end{pmatrix} = \begin{pmatrix} P_{\bar{S}} M P_{\bar{S}} & P_S M P_{\bar{S}} \\ P_{\bar{S}} M P_S & P_S M P_S \end{pmatrix}$$

We can represent the matrix M exactly on this subspace by computing its matrix elements, known as the *little Dirac operator* (coarse grid matrix in multi-grid)

$$A_{jk}^{ab} = \langle\phi_j^a|M|\phi_k^b\rangle \qquad ; \qquad (M_{SS}) = A_{ij}^{ab}|\phi_i^a\rangle\langle\phi_j^b|.$$

the subspace inverse can be solved by Krylov methods and is:

$$Q = \begin{pmatrix} 0 & 0 \\ 0 & M_{SS}^{-1} \end{pmatrix} \qquad ; \qquad M_{SS}^{-1} = (A^{-1})_{ij}^{ab}|\phi_i^a\rangle\langle\phi_j^b|$$

It is important to note that A inherits a sparse structure from M because well separated blocks do *not* connect through M.

Multigrid preconditioners

Equivalence of a sequence of multigrid correction steps to a preconditioner can be seen if consider the V_{11} with a pre-smoother (S), coarse correction (Q), and post-smoother (S) in sequence,

$$x_1 = x_0 + Sr_0 \tag{3}$$
$$x_2 = x_1 + Qr_1 \tag{4}$$
$$x_3 = x_2 + Sr_2. \tag{5}$$

Substitute and reduce the final update in terms of $r_0 = b - Mx_0$ and x_0,

$$r_1 = b - Mx_1 = r_0 - MSr_0 \tag{6}$$
$$r_2 = b - Mx_2 = r_0 - MSr_0 - MQr_0 + MQMSr_0. \tag{7}$$

The final update sequence is then,

$$x_3 = x_0 + [S(1 - MQ) + Q + (1 - QM)S + S(MQM - M)S]\, r_0 \tag{8}$$
$$= x_0 + [SP_L + Q + P_R S + SP_L MS]\, r_0. \tag{9}$$

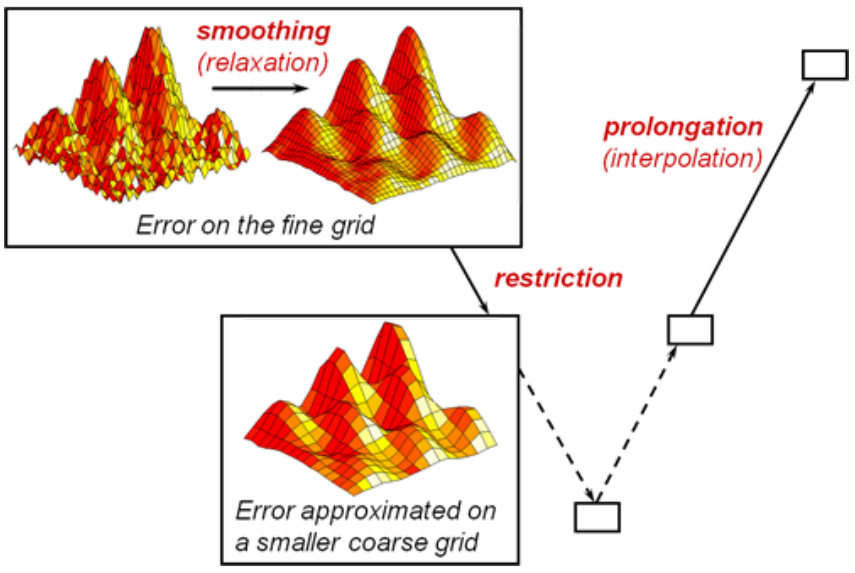
This $V(1,1)$ multigrid error cycle suggests the adoption of the matrix,

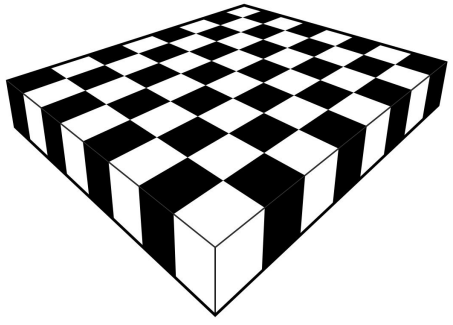
$$[SP_L + Q + P_R S + SP_L MS]$$

applied to the current residual as a preconditioner in an outer Krylov solver, with its implementation being as the above sequence of error correction steps based on the current residual r_0 as input.

Multigrid : how it works

- Project to low dimensional basis that captures the low mode space
- Represent the original matrix in this truncated basis
- Inverse of this truncated representation corrects the current solution



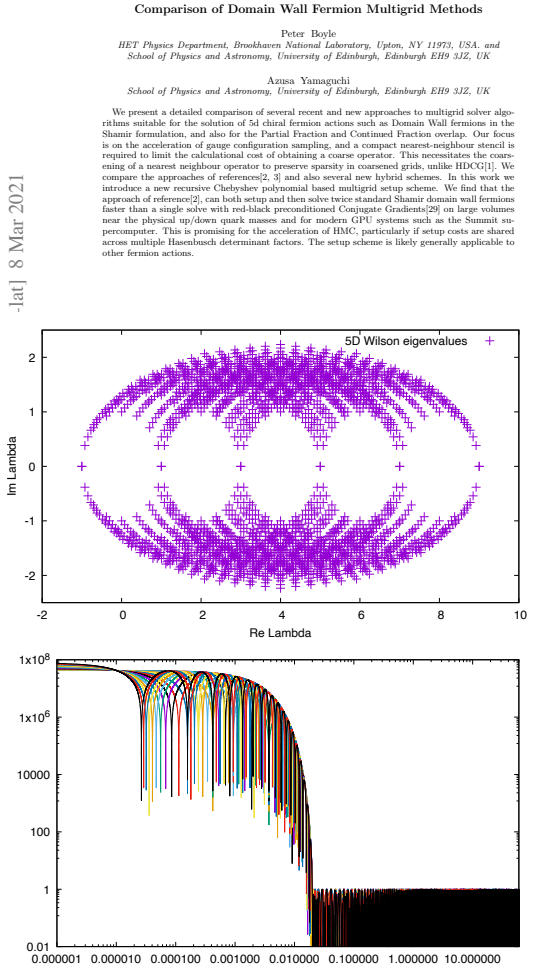


	Fine	Coarse	CoarseCoarse	Evecs
$\lambda_{\min}$	1.0e-6	1.0e-6	1.0e-6	1.0e-6
$\lambda_{\max}$	60	11	5.0	4.0e-3

- Improve the condition number by lowering the cut-off as you go coarser
- Arguably a surface to volume suppression of the high modes as you block
- Smoother step helps cheaply wipe out the effects while preserving the low mode element of coarse correction

Domain Wall Multigrid

- Preprint: https://arxiv.org/pdf/2103.05034.pdf
- Spectrum of DWF makes coarsening nearest neighbour operator *hard*
  - Polynomial approximation to $\frac{1}{z}$ in region of complex plane enclosing origin
  - Typically solve normal equations on positive definite $M^\dagger M$
  - Nearest neighbour coarsenings of $\gamma_5 R_5 D_{dwf}$ (Herm, indefinite)
- Novel chebyshev polynomial setup of multigrid
- Result:
  Set up and solve twice D_{dwf} faster than red-black CG
- HMC focus; use compressed Lanczos for valence analysis



Comparison of Domain Wall Fermion Multigrid Methods

Peter Boyle
HET Physics Department, Brookhaven National Laboratory, Upton, NY 11973, USA. and School of Physics and Astronomy, University of Edinburgh, Edinburgh EH9 3JZ, UK

Azusa Yamaguchi
School of Physics and Astronomy, University of Edinburgh, Edinburgh EH9 3JZ, UK

We present a detailed comparison of several recent and new approaches to multigrid solver algorithms suitable for the solution of 5d chiral fermion actions such as Domain Wall fermions in the Shamir formulation, and also for the Partial Fraction and Continued Fraction overlap. Our focus is on the acceleration of gauge configuration sampling, and a compact nearest-neighbour stencil is required to limit the calculational cost of obtaining a coarse operator. This necessitates the coarsening of a nearest neighbour operator to preserve sparsity in coarsened grids, unlike HDCG[1]. We compare the approaches of references[2, 3] and also several new hybrid schemes. In this work we introduce a new recursive Chebyshev polynomial based multigrid setup scheme. We find that the approach of reference[2], can both setup and then solve twice standard Shamir domain wall fermions faster than a single solve with red-black preconditioned Conjugate Gradients[29] on large volumes near the physical up/down quark masses and for modern GPU systems such as the Summit supercomputer. This is promising for the acceleration of HMC, particularly if setup costs are shared across multiple Hasenbusch determinant factors. The setup scheme is likely generally applicable to other fermion actions.