

## Computational Methods (practice) - Lecture 3

Peter Boyle (BNL, Edinburgh)

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

# 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  $O(N)$   $spb, Mb, \dots M^N b$
- *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

# GMRES

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

- Consider  $\psi = (c_n M^n) b$  in the Krylov space
- GMRES minimises the Euclidean norm  $|r|^2$  with respect to the coefficients  $c_n$
- Matrix elements are  $c_m^* c_n \langle M^{m+1} b | M^{n+1} b \rangle$
- This requires all  $N$  vectors to be retained, and coefficients selected after dense matrix algebra to determine  $c_m$
- GMRES(k) runs for  $k$  iterations and then *restarts* to limit storage
- Often used in a preconditioner/smoothers

[https://en.wikipedia.org/wiki/Generalized\\_minimal\\_residual\\_method](https://en.wikipedia.org/wiki/Generalized_minimal_residual_method)

# Lanczos orthogonal sequence

<http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter10.pdf>

The Lanczos and conjugate gradient algorithms, Meurant

- Krylov space  $K_n(b) = \{b, Ab, \dots, A^n b\}$
- 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

# Conjugate Gradients

[https://en.wikipedia.org/wiki/Conjugate\\_gradient\\_method](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  $x = \sum_k \alpha_k p_k$ :

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

$\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0$

if  $\mathbf{r}_0$  is sufficiently small, then return  $\mathbf{x}_0$  as the result

$\mathbf{p}_0 := \mathbf{r}_0$

$k := 0$

repeat

$$\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top A \mathbf{p}_k}$$

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

$$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k A \mathbf{p}_k$$

if  $\mathbf{r}_{k+1}$  is sufficiently small, then exit loop

$$\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$$

$$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

$$k := k + 1$$

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](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 number

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

- 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  $\sigma$  governs the long tail convergence of CG in practice
- (left) Preconditioning: changing  $\kappa$  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

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

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:

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

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

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

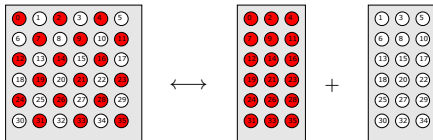
The even checkerboard solution can be inferred via

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

$M_{pc}$  (empirically) better conditioned than  $M$ : red black solvers converge  $O(3\times)$  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.
- Grid has a Chebyshev polynomial preconditioned Lanczos algorithm:

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

- We will not discuss Lanczos in detail: it is used to produce the lowest lying eigenvectors of the Dirac operator
- These can be handled exactly, and removed 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.

[https://github.com/paboyle/Grid/blob/develop/examples/Example\\_Laplacian\\_solver.cc](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