Solvers II — Preconditioning and Deflation

Lattice Practices 2014

Andreas Frommer, Björn Leder

based on a lecture by Karsten Kahl

Bergische Universität Wuppertal

March 7, 2014

Outline

How to improve an optimal method?

Solvers I: Krylov subspace methods are all-duty solvers

- "Optimal" methods for any application
- Fast (i.e., short-recurrence) solvers for many applications
- Convergence dependent on conditioning of A, e.g.,
  - Conjugate Gradients

$$\|e^{(k)}\|_A \leq 2 \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \right)^k \|e^{(0)}\|_A, \quad \kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

How to improve convergence of Krylov subspace methods?

1. Preconditioning
2. Deflation

Scaling issues in Numerical Simulations

Numerical simulations of partial differential equations (PDEs)

$$\mathcal{L}\psi = \varphi$$

Discretization of $\mathcal{L}$ on mesh with spacing a yields

$$\mathbf{L}x = f$$

- Depending on PDE order and order of discretization

$$\kappa(\mathbf{L}) \sim a^{-\sigma}, \quad \sigma \in \mathbb{N}^+$$

- Increasing accuracy of discretization ($a \to 0$)

$$\kappa(\mathbf{L}) \longrightarrow \infty \quad (a \to 0)$$

Performance of Krylov methods deteriorates when $a \to 0$!

Preconditioning — Idea

Idea: Improve conditioning of A in $Ax = b$!

- Instead of solving $Ax = b$ consider solving

$$\begin{aligned} S_\ell A S_r y &= S_\ell b \\ x &= S_r y \end{aligned}$$

with preconditioners S_ℓ, S_r s.t. $\kappa(S_\ell A S_r) \ll \kappa(A)$

Open questions

- What are the design goals for preconditioners?
- What are suitable choices of S_ℓ, S_r?
- How does the preconditioner fit in the iteration
  - Ideally only $A\cdot, S_\ell\cdot$ and $S_r\cdot$ are required

For now consider only left-preconditioning with $S = S_\ell$

Preconditioning — Observations

Consider extreme cases

- $S = I$

  $\Rightarrow \; SA = A$ original setting
- $S = A^{-1}$

  $\Rightarrow \; SA = I$ and $\kappa(SA) = 1$ (ideal)
- $S = A^\dagger$

  $\Rightarrow \; SA = A^\dagger A$ hermitian, but $\kappa(SA) = \kappa(A)^2$

In order to speed up convergence preconditioner S should

- approximate A^{-1}
- be cheap to compute $(S\cdot)$

Preconditioning — CG

Recall: Conjugate Gradients requires A hermitian

Problem: SA in general no longer hpd even if S is hpd, but then

$$\langle SAx, y\rangle_{S^{-1}} = \langle Ax, y\rangle_2 = \langle x, Ay\rangle_2 = \langle x, SAy\rangle_{S^{-1}}$$

Solution: Replace all $\langle ., .\rangle_2$ by $\langle ., .\rangle_{S^{-1}}$

- Rewriting the algorithm one even gets rid of $\langle ., .\rangle_{S^{-1}}$
- CG variants exist for any A hermitian in some $\langle ., .\rangle_B$

Changing the inner product also works when preconditioning other methods which require a special relation between A and its adjoint $A^\dagger$, e.g., MINRES, SUMR

PCG — Algorithm

Preconditioned Conjugate Gradients

$r^{(0)} = b - Ax^{(0)}, z^{(0)} = Sr^{(0)}, p^{(0)} = z^{(0)}$

for $k = 1, 2, \dots$ **do**

$\alpha_{k-1} = \frac{\langle r^{(k-1)}, z^{(k-1)} \rangle_2}{\langle Ap^{(k-1)}, p^{(k-1)} \rangle_2}$

$x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)}$

$r^{(k)} = r^{(k-1)} - \alpha_{k-1} Ap^{(k-1)}$

$z^{(k)} = Sr^{(k)}$

$\beta_{k-1} = \frac{\langle r^{(k)}, z^{(k)} \rangle_2}{\langle r^{(k-1)}, z^{(k-1)} \rangle_2}$

$p^{(k)} = z^{(k)} + \beta_{k-1} p^{(k-1)}$

end for

Preconditioned GMRES(m)

while not converged **do**
  $r^{(0)} = S(b - Ax^{(0)}), \beta = \|r^{(0)}\|_2, v_1 = \beta^{-1} r^{(0)}$
  for $j = 1, \ldots, m$ **do**
    $w = SAv_j$
    for $i = 1, \ldots, j$ **do**
      $h_{i,j} = \langle w, v_j \rangle_2$
      $w = w - h_{i,j} v_j$
    end for
    $h_{j+1,j} = \|w\|_2$
    $v_{j+1} = h_{j+1,j}^{-1} w$
  end for
  Define $V_m = [v_1 \mid \ldots \mid v_m], H_{m+1,m} = \{h_{i,j}\}_{1 \le j \le m, 1 \le i \le j+1}$
  Solve $y_m = \operatorname{argmin}_y \|\beta e_1 - H_{m+1,m} y\|_2$
  $x^{(0)} = x^{(0)} + V_m y_m$
end while

Preconditioned BiCGstab

$r^{(0)} = b, \beta_0 = 0$

$\hat{r} = r$ shadow residual $\langle r, \hat{r} \rangle_2 \neq 0$

for $k = 0, 1, \dots$ **do**

$\rho_k = \langle r^{(k)}, \hat{r} \rangle_2$

$\beta_k = \frac{\rho_k}{\rho_{k-1}} \cdot \frac{\alpha_{k-1}}{\omega_{k-1}}$

$p^{(k)} = r^{(k)} + \beta_k (p^{k-1} - \omega_{k-1} v^{(k-1)})$

$\hat{p}^{(k)} = S p^{(k)}$

$\alpha_k = \frac{\rho_k}{\langle A\hat{p}^{(k)}, \hat{r} \rangle_2}$

$x^{(k+\frac{1}{2})} = x^{(k)} + \alpha_k \hat{p}^{(k)}$

$s^{(k)} = r^{(k)} - \alpha_k A \hat{p}^{(k)}$ $s^{(k)} \equiv r^{(k+\frac{1}{2})}$

$\hat{s}^{(k)} = S s^{(k)}$

$\omega_k = \frac{\langle s^{(k)}, A\hat{s}^{(k)} \rangle_2}{\langle A\hat{s}^{(k)}, A\hat{s}^{(k)} \rangle_2}$

$x^{(k+1)} = x^{(k+\frac{1}{2})} + \omega_k \hat{s}^{(k)}$

$r^{(k+1)} = s^{(k)} - \omega_k A \hat{s}^{(k)}$

end for

Preconditioners

Aims for the construction of preconditioners S

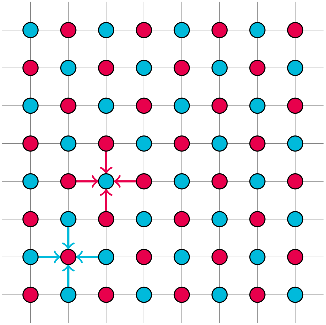
1. $S \approx A^{-1}$ to get speed-up
2. $S\cdot$ should be cheap (1 application per iterate)

Classes of preconditioners to be discussed

- Structural preconditioners
- Splitting-based preconditioners
- Domain decomposition preconditioners
- Multigrid preconditioners
- Incomplete decomposition preconditioners

Odd-even preconditioning

Discretizations on lattices with next neighbor coupling



- Nodes are odd or even

Ordering by odd-even

$$A = \begin{bmatrix} A_{oo} & A_{oe} \\ A_{eo} & A_{ee} \end{bmatrix}$$

with diagonal A_{oo} and A_{ee}

- A_{oo}^{-1}, A_{ee}^{-1} trivial
- odd decoupled
- even decoupled

Solve first even then odd

Odd-even preconditioning

With $\hat{A}_{ee} = A_{ee} - A_{eo}A_{oo}^{-1}A_{oe}$ solution of $Ax = b$ given by

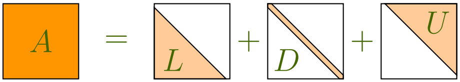
Odd-Even Reduction

$y_o = A_{oo}^{-1}b_o$
Solve $\hat{A}_{ee}x_e = b_e - A_{eo}y_o$
$x_o = y_o - A_{oo}^{-1}A_{oe}x_e$

- Iteratively solving $\hat{A}_{ee}x_e = b_e - A_{eo}y_o$
  $\Rightarrow$ Odd-Even preconditioner
- If A has constant diagonal $\kappa(\hat{A}_{ee}) < \kappa(A)$
  $\Rightarrow$ Solving $\hat{A}_{ee}$ is easier than solving A
- Since A_{oo}^{-1} is cheap (diagonal!)
  $\Rightarrow$ Cost for $\hat{A}_{ee}\cdot \quad \approx \quad$ Cost for $A\cdot$

Splitting methods

Splitting methods use the additive decomposition of A



- Jacobi: $x^{(k+1)} = x^{(k)} + D^{-1}r^{(k)}$
- Gauss-Seidel: $x^{(k+1)} = x^{(k)} + (D+L)^{-1}r^{(k)}$
- SOR: $x^{(k+1)} = x^{(k)} + (\frac{1}{\omega}D+L)^{-1}r^{(k)}$

General splitting method: $A = M + N$

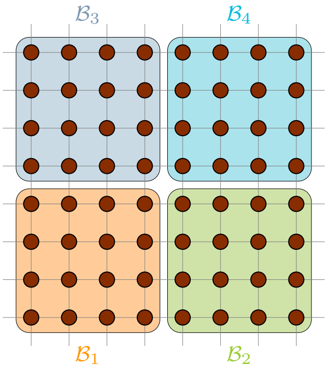
$$x^{(k+1)} = x^{(k)} + M^{-1}r^{(k)} \Longrightarrow e^{(k+1} = e^{(k)} - M^{-1}Ae^{(k)}$$

Convergent iff $\|I - M^{-1}A\| < 1$ for some norm $\|\cdot\|$

$\|I - M^{-1}A\|$ small $\Rightarrow$ $M^{-1}A \approx I$ $\Rightarrow$ M^{-1} preconditioner

Domain Decomposition*

- Split the computational domain into subdomains $\mathcal{B}_i$
- Solve system iteratively on each subdomain



- Canonical injection $\mathcal{I}_j$

$$\mathcal{I}_j e_i = e_{(B_j)_i}$$

- Restriction of x onto $\mathcal{B}_j$

$$x_{\mathcal{B}_j} = \mathcal{I}_j^\dagger x$$

- Restriction of A onto $\mathcal{B}_j$

$$A_{\mathcal{B}_j} = \mathcal{I}_j^\dagger A \mathcal{I}_j$$

*Domain decomposition dates back to H. Schwarz (1870)

Additive and Multiplicative Schwarz

Additive Schwarz

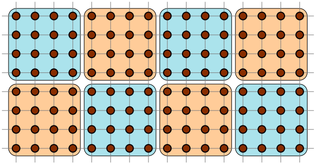
for $k = 0, 1, \dots$ **do**
  $r^{(k)} = b - Ax^{(k)}$
  for $j = 1, 2, \dots, n_B$ **do**
    $x^{(k+1)}_{\mathcal{B}_j} = x^{(k)}_{\mathcal{B}_j} + A^{-1}_{\mathcal{B}_j} r^{(k)}_{\mathcal{B}_j}$
  end for
end for

- Block-Jacobi
- Embarrassingly parallel

Multiplicative Schwarz

for $k = 0, 1, \dots$ **do**
  for $j = 1, 2, \dots, n_B$ **do**
    $r = b - Ax$
    $x_{\mathcal{B}_j} = x_{\mathcal{B}_j} + A^{-1}_{\mathcal{B}_j} r_{\mathcal{B}_j}$
  end for
end for

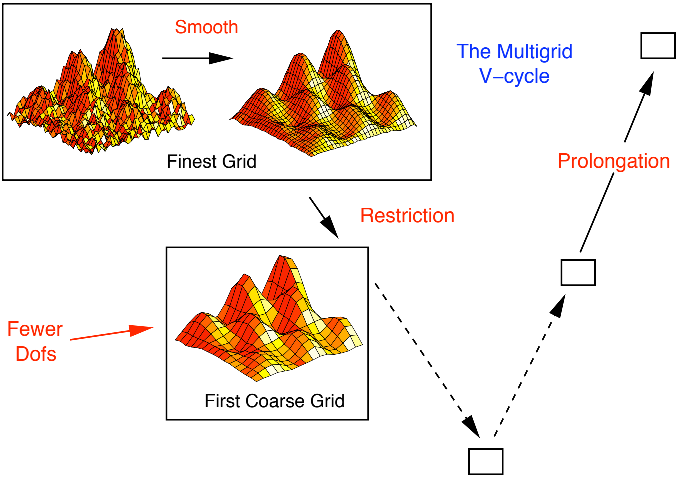
- Block-Gauss-Seidel
- Sequential ($\rightarrow$ coloring)



Schwarz methods in general

- $\oplus$ Data parallel
- $\oplus$ Computation parallel

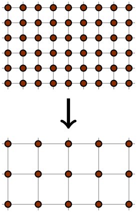
Multigrid


Smooth
Finest Grid
The Multigrid
V–cycle
Prolongation
Restriction
Fewer
Dofs
First Coarse Grid

(Algebraic) Multigrid

Given:
- $Ax = b$
- Iterative method S ("smoother")

Wanted:
- Hierarchy of systems
  $A_\ell x_\ell = b_\ell, \quad \ell = 0, \dots, L$
- Intergrid transfer operators
  $P_{\ell+1}^{\ell} : \mathbb{C}^{n_{\ell+1}} \longrightarrow \mathbb{C}^{n_\ell}$
  $R_\ell^{\ell+1} : \mathbb{C}^{n_\ell} \longrightarrow \mathbb{C}^{n_{\ell+1}}$



Smoother

$$S_\ell : \mathbb{C}^{n_\ell} \longrightarrow \mathbb{C}^{n_\ell}$$

"High modes"

Interpolation

$$P_{\ell+1}^{\ell} : \mathbb{C}^{n_{\ell+1}} \longrightarrow \mathbb{C}^{n_\ell}$$

"Low modes"

Complementarity of Smoother and Interpolation

Generic Multigrid Algorithm — $\mathsf{MG}_\ell(A_\ell, b_\ell)$

if $\ell = L$ **then**
  $x_L = A_L^{-1} b_L$
else
  $x_\ell = 0$
  for $i = 1, \ldots, \nu_1$ **do**
    $x_\ell = S_\ell(x_\ell, b_\ell)$ $\quad (x_\ell \leftarrow x_\ell + M_\ell^{-1} r_\ell, r_\ell = b_\ell - A_\ell x_\ell)$
    "Pre-smooothing"
  end for
  $x_{\ell+1} = \mathsf{MG}(A_{\ell+1}, R_{\ell+1}^{\ell}(b_\ell - A x_\ell))$
  $x_\ell = x_\ell + P_{\ell+1}^{\ell} x_{\ell+1}$ "Coarse-grid correction"
  for $i = 1, \ldots, \nu_2$ **do**
    $x_\ell = S_\ell(x_\ell, b_\ell)$ "Post-smoothing"
  end for
end if

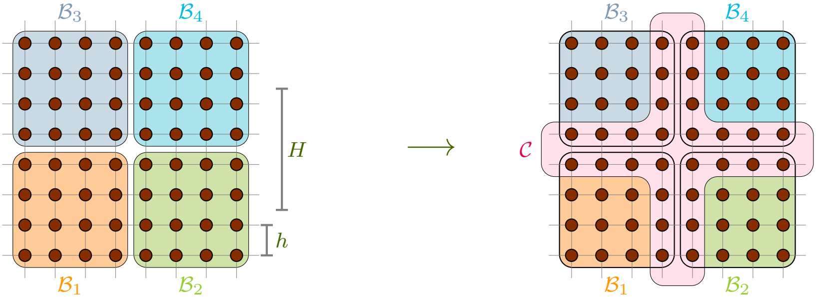
Optimality of Multigrid

For certain classes of discretizations of certain types of PDEs and appropriate variants of **multigrid** we have

- Multigrid can be used as a **stand alone** solver (no wrapping as a preconditioner into a Krylov subspace method)
- no. of iterations for given accuracy **independent** of no. of variables.
  "optimal method"

Even when not optimal as a stand alone solver, multigrid is often a very efficient preconditioner.

To be efficient, **domain decomposition** needs an additional small system $A_{\mathcal{C}}$ which couples the boundaries of the domains.

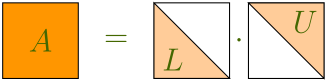


For certain classes of discretizations of certain types of PDEs and appropriate variants of **domain decomposition** we have

- Domain decomp. can be used as a **stand alone** solver
- no. of iterations for given accuracy $\propto \log(H/h)$

Incomplete LU (ILU)

Recall: Direct methods are based on factorization of A



Drawback: Fill-In in L and U for sparse A

Idea: Incomplete factorizations with sparse L and U

1. Prescribe the non-zero pattern (e.g., non-zeroes of A)
   - Minimize the error-matrix E in $A = \tilde{L}\tilde{U} + E$
2. Use drop-tolerance θ to drop small entries in L and U
   - Often: $\left(A^{-1}\right)_{i,j} \sim \alpha^{\mathrm{dist}_G(i,j)}, \quad \alpha < 1$

     $\Rightarrow$ If i is "far" from j, L_{ij} and U_{ij} will be dropped

ILU is a black-box preconditioner

Flexible Krylov subspace methods

The preconditioner may be an iterative process by itself

- choice 1: fixed no. of iterations or stopping criterion?
- choice 2: stationary or non-stationary iteration
- For red choices: $S\cdot$ changes in each iteration $\rightarrow S = S_k$
- There is no longer a Krylov subspace defined by

$$\mathcal{K}_k(SA, b) = \{b, SAb, (SA)^2 b, \dots, (SA)^{k-1} b\}$$

$\Rightarrow$ Convergence theory does not hold anymore

- Algorithmic realizations have to be modified!
  - $\Rightarrow$ Flexible Krylov subspace methods

Flexible CG — Algorithm

Flexible Conjugate Gradients

$r^{(0)} = b - Ax^{(0)}, z^{(0)} = S_0 r^{(0)}, p^{(0)} = z^{(0)}$

for $k = 1, 2, \ldots$ **do**

$$\alpha_{k-1} = \frac{\langle r^{(k-1)}, z^{(k-1)} \rangle_2}{\langle Ap^{(k-1)}, p^{(k-1)} \rangle_2}$$

$$x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)}$$

$$r^{(k)} = r^{(k-1)} - \alpha_{k-1} A p^{(k-1)}$$

$$z^{(k)} = S_k r^{(k)}$$

$$\beta_{k-1} = \frac{\langle r^{(k)} - r^{(k-1)}, z^{(k)} \rangle_2}{\langle r^{(k-1)}, z^{(k-1)} \rangle_2}$$

$$p^{(k)} = z^{(k)} + \beta_{k-1} p^{(k-1)}$$

end for

- If $S_k = S$ for all k $\implies$ $z^{(k)} \perp r^{(k-1)}$
- Flexible CG preserves local optimality

Flexible GMRES(m)

while not converged **do**
  $r^{(0)} = S_0(b - Ax^{(0)}), \beta = \|r^{(0)}\|_2, v_1 = \beta^{-1} r^{(0)}$
  for $j = 1, \ldots, m$ **do**
    $z_j = S_j v_j$
    $w = Az_j$
    for $i = 1, \ldots, j$ **do**
      $h_{i,j} = \langle w, v_j \rangle_2$
      $w = w - h_{i,j} v_j$
    end for
    $h_{j+1,j} = \|w\|_2$
    $v_{j+1} = h_{j+1,j}^{-1} w$
  end for
  Define $Z_m = [z_1 \mid \ldots \mid z_m], H_{m+1,m} = \{h_{i,j}\}_{1 \leq j \leq m, 1 \leq i \leq j+1}$
  Solve $y_m = \operatorname{argmin}_y \|\beta e_1 - H_{m+1,m} y\|_2$
  $x^{(0)} = x^{(0)} + Z_m y_m$
end while

Preconditioners — Summary

Preconditioning **improves convergence** if $\kappa(SA) \ll \kappa(A)$

- There is a wide variety of preconditioners available
  - Most of them require knowledge about A or its origins
- Goals when constructing preconditioners S are
  - $S \approx A^{-1}$ and $S\cdot$ cheap

Preconditioning makes Krylov subspace methods **more robust**

- Reducing $\kappa(A)$ helps controlling the error $e^{(k)}$, since

$$\|e\|_2 \leq c\kappa(A)^{-1}\|r\|_2$$

$\Rightarrow$ If $\kappa(A) \gg 1$ results based on $\|r\|_2$ should not be trusted!
$\Rightarrow$ If $\kappa(A) \gg 1$ a preconditioner is mandatory!

Deflation — Idea (A hermitian and positive definite)

Assume A hermitian and positive definite
Then convergence is slowed down by small eigenmodes

- Given the "troublesome" modes $v_1, \dots, v_\ell$
  $\Rightarrow$ deflate the subspace $\mathcal{V} = \operatorname{colspan}(\underbrace{[v_1 \mid \dots \mid v_\ell]}_{=V})$

Similar to preconditioning, instead of $Ax = b$ solve

$$\begin{aligned} A\,(I - \pi_A(V))\,\hat{x} &= (I - \pi_A(V))\,b \\ x &= \hat{x} + V(V^\dagger AV)^{-1}V^\dagger b \end{aligned}$$

with $\pi_A(V) = V(V^\dagger AV)^{-1}V^\dagger A$

- In case v_i are eigenmodes, $V^\dagger AV = \operatorname{diag}(\lambda_1, \dots, \lambda_\ell)$
  $\Rightarrow$ $(V^\dagger AV)^{-1}$ nothing to worry about

Deflation — Conjugate Gradients Theory

The effective condition number κ_{eff} replaces κ in theory

$$
\begin{aligned}
\kappa_{\text{eff}} &= \frac{\mu_1}{\mu_\ell} \\
\mu_1 &= \max_{x \neq 0} \frac{\langle A(I - \pi_A(V))x, x\rangle_2}{\langle x, x\rangle_2} \\
\mu_\ell &= \min_{x \in \mathcal{V}^\perp \setminus \{0\}} \frac{\langle A(I - \pi_A(V))x, x\rangle_2}{\langle x, x\rangle_2}
\end{aligned}
$$

- If v_i are smallest ℓ eigenmodes

$$\kappa_{\text{eff}} = \frac{\lambda_{\max}}{\lambda_{\ell+1}}$$

where $\lambda_{\ell+1}$ is the $(\ell+1)^{st}$ smallest eigenvalue

Deflated CG — Algorithm

Deflated CG (Deflation space $\mathcal{V} = \mathrm{colspan}(V)$)

$x^{(0)} = x^{(0)} + \pi_A(V)b$
$r^{(0)} = b - Ax^{(0)}$
$p^{(0)} = (I - \pi_A(V))r^{(0)}$
for $k = 1, 2, \dots$ **do**
  $\alpha_{k-1} = \frac{\langle r^{(k-1)}, r^{(k-1)} \rangle_2}{\langle Ap^{(k-1)}, p^{(k-1)} \rangle_2}$
  $x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)}$
  $r^{(k)} = r^{(k-1)} - \alpha_{k-1} Ap^{(k-1)}$
  $\beta_{k-1} = \frac{\langle r^{(k)}, r^{(k)} \rangle_2}{\langle r^{(k-1)}, r^{(k-1)} \rangle_2}$
  $p^{(k)} = (I - \pi_A(V))r^{(k)} + \beta_{k-1} p^{(k-1)}$
end for

GMRES(m)

On restart all information about $\mathcal{K}_m(A, r^{(0)})$ is lost!

- Use deflation technique to transfer information

Note: Due to the Arnoldi relation $V_m^\dagger A V_m = H_{m,m}$ we have

- Eigenmodes $w_1, \ldots, w_m$ of $H_{m,m}$ give approximations $V_m w_1, \ldots, V_m w_m$ for eigenmodes of A

$$H_{mm} w_i = \lambda_i w_i \;\Rightarrow\; V_m^\dagger (A V_m w_i - \lambda_i V_m w_i) = 0$$

- Vectors $V_m w_i$ are called Ritz vectors ($\rightarrow$ ARPACK)

Idea: Use smallest eigenmodes of $H_{m,m}$ in deflation

Deflated GMRES(m) — Sketch

$\tilde{V} = \emptyset$
for $\ell = 0, 1, \dots$ **do**
  $r^{(0)} = b - Ax^{(0)}, \beta = \|r^{(0)}\|_2, v_1 = \beta^{-1} r^{(0)}$
  Compute $V_m, H_{m+1,m}$ based on initial $\tilde{V}$ *(Arnoldi)*
  Compute smallest Ritz vectors $V_m w_1, \dots, V_m w_\ell$
  $y_m = \operatorname{argmin}_y \|\beta e_1 - H_{m+1,m} y\|_2$
  $x^{(0)} = x^{(0)} + V_m y_m$
  $\tilde{V} = [V_m w_1 \mid \dots \mid V_m w_\ell]$
end for

- For a more detailed description see [4]
- Reusing information upon restart is also known as...
  - ...recycling
  - ...augmenting

Deflation — Summary

Deflation "hides" most difficult part of the problem

- Computation of eigenmodes necessary
  - possibly on-the-fly (Deflated GMRES(m))
  - possibly a priori knowledge available
  - approximations viable ($\rightarrow$ ARPACK)
- Analysis of general deflation subspaces $\mathcal{V}$ (cf. [3])

Eigenmode deflation suffers from scaling (i.e., $a \rightarrow 0$)

- In order to have constant number of iterations for $a \rightarrow 0$

$$\kappa_{\text{eff}} = \text{const} \quad \Longleftrightarrow \quad \lambda_{min}^{\text{eff}} > \sigma$$

- Often number N_σ of eigvalues below threshold σ fulfills

$$N_\sigma \sim \text{system size } n \longrightarrow \infty \quad (a \rightarrow 0)$$

$\Rightarrow$ More eigenmodes need to be computed as $a \rightarrow 0$

Summary

To find an efficient solver is hard, but there are guidelines

- Use as much information about your system as possible
  - In the choice of the Krylov subspace method
    - Short recurrence method available?
    - Optimal method available?
  - In the choice of the preconditioner
- Adjust parameters of your method w.r.t. hardware, e.g.,
  - Restart length in GMRES(m)
  - Dimension of the deflation subspace
  - Dimension of the subdomains in domain decomposition

Most often there is no obvious optimal choice for the solver!

Construction of optimal solvers is ongoing research!

A. Frommer, K. Kahl, S. Krieg, B. Leder and M. Rottmann.
Aggregation-based multilevel methods for lattice QCD.
arXiv:1202.2462 [hep-lat], 2012

A. Greenbaum.
Iterative Methods for Solving Linear Systems, volume 17 of *Frontiers in Applied Mathematics*.
Society for Industrial and Applied Mathematics, 1997.

K. Kahl and H. Rittich.
Analysis of the deflated conjugate gradient method based on symmetric multigrid theory.
Submitted (pre-print: http://arxiv.org/abs/1209.1963), 2012.

R. Morgan.
Gmres with deflated restarting.
SIAM J. Sci. Comput., 24, 2002.

R. A. Nicolaides.
Deflation of conjugate gradients with applications to boundary value problems.
SIAM J. Numer. Anal., 24, 1987.

Y. Notay.
Flexible conjugate gradients.
SIAM J. Sci. Comput., 22:1444–1460, 2000.

U. Trottenberg, C. Oosterlee, and A. Schüller.
Multigrid.
Academic Press, San Diego (CA), 2001.

Y. Saad.
Iterative Methods for Sparse Linear Systems.
Society for Industrial and Applied Mathematics, 2nd edition, 2003.

B. Smith, P. Bjørstadt, and W. Gropp.
Domain Decomposition: Parallel Methods for Elliptic Partial Differential Equations.
Cambridge University Press, New York, 1996.