Iterative Methods for Sparse Linear Systems

Luca Bergamaschi

e-mail: berga@dmsa.unipd.it - http://www.dmsa.unipd.it/~berga

Department of Mathematical Methods and Models

for Scientific Applications

University of Padova

Discretized diffusion equation

After discretization in the space, the PDE is converted in a system of ODEs like

$$P\frac{\partial \boldsymbol{u}}{\partial t} = H\boldsymbol{u} + \boldsymbol{b}$$

where $P = I$ in FD discretization. Stiffness matrix H is symmetric definite negative, capacity or mass matrix P is symmetric positive definite.

Finally discretization in time transforms the system of ODEs to a linear system of size equal to the number of gridpoints to be solved at each time step.
Time discretization divides the time interval $(0, T]$ into subintervals of length Δt_i so that solution in time is computed at $t_k = \sum_{i=1}^{k} \Delta_i$ and uses Finite Differences to approximate the time derivative.

For example applying the implicit Euler method ($\boldsymbol{u} \approx \frac{\boldsymbol{u}(t_{k+1}) - \boldsymbol{u}(t_k)}{\Delta t_k}$).

$$(I + \Delta t_k H)\boldsymbol{u}(t_{k+1}) = \Delta t_k P\boldsymbol{u}(t_k) + \Delta t_k \boldsymbol{b}$$

Eigenvalues and condition number of the Laplacian

The 2D FD discretization of the Laplace equation takes a general form (in the case $\Delta x = \Delta y = h$):

$$H = \frac{1}{h^2} \begin{pmatrix} S & I & & & \\ I & S & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & S & I \\ & & & I & S \end{pmatrix} \quad \text{where} \quad S = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix}$$

Eigenvalues of S and hence those of H are explicitly known:

$$\lambda_{j,k}(H) = \frac{4}{h^2} \sin^2\left(\frac{j\pi h}{2}\right) + \frac{4}{h^2} \sin^2\left(\frac{k\pi h}{2}\right), \quad j,k = 1,\ldots,n_x = \frac{1}{h} - 1$$

Eigenvalues and condition number of the Laplacian

Theorem. The smallest and largest eigenvalues of H behave like:
$\lambda_n = 2\pi^2 + O(h^2),\ \lambda_1 = 8h^{-2} + O(1)$.
Proof.

$$\begin{aligned}
\lambda_n &= \frac{8}{h^2}\sin^2\left(\frac{\pi h}{2}\right) = \frac{8}{h^2}(h\pi/2 + O(h^3))^2 = 2\pi^2 + O(h^2). \\
\lambda_1 &= \frac{8}{h^2}\sin^2\left(\frac{n_x \pi h}{2}\right) = \frac{8}{h^2}\sin^2\left(\frac{\pi}{2} - \frac{\pi h}{2}\right) = \\
&= \frac{8}{h^2}\cos^2\left(\frac{\pi h}{2}\right) = \frac{8}{h^2}(1 - O(h^2))^2 = 8h^{-2} + O(1).
\end{aligned}$$

Corollary. The condition number of H behaves like

$$\kappa(H) = \frac{4}{\pi^2}h^{-2} + O(1) \qquad (\kappa(H) = \frac{3d}{\pi^d}h^{-2} \text{ when } \Omega \subset \mathbb{R}^d)$$

Corollary. The number of iteration of the CG for solving $H\boldsymbol{x} = \boldsymbol{b}$ is proportional to

$$h^{-1} = \begin{cases} n & \text{1D discretizations} \\ \sqrt{n} & \text{2D discretizations} \\ \sqrt[3]{n} & \text{3D discretizations} \end{cases}.$$

Iterative methods on the diffusion equations

Again stationary iterative methods

1. Jacobi iteration. $\boldsymbol{x}_{k+1} = -D^{-1}(L+U)\boldsymbol{x}_k + D^{-1}\boldsymbol{b} = E_J\boldsymbol{x}_k + \boldsymbol{q}_J$.
2. Gauss-Seidel iteration.

$$\boldsymbol{x}_{k+1} = -(D+L)^{-1}U\boldsymbol{x}_k + (D+L)^{-1}\boldsymbol{b} = E_{GS}\boldsymbol{x}_k + \boldsymbol{q}_{GS}$$

3. Acceleration of Gauss-Seidel $\Longrightarrow$ SOR method depending on $\omega \in \mathbb{R}$.

$$\boldsymbol{x}_{k+1} = -(D+\omega L)^{-1}\left((1-\omega)D - \omega U\right)\boldsymbol{x}_k + \omega(D+\omega L)^{-1}\boldsymbol{b} = E_{SOR}\boldsymbol{x}_k + \boldsymbol{q}_{SOR}$$

Theorem. (Young & Varga). If all the eigenvalues of the Jacobi iteration matrix are real then

1. $\lambda(E_{GS}) = \lambda(E_J)^2$
2. If in addition $\rho(E_J) < 1$ then there is an optimal value of ω:
$$\omega_{opt} = \frac{2}{1+\sqrt{1-\rho(E_J)^2}}.$$
3. $\rho(E_{SOR}(\omega_{opt})) = \omega_{opt} - 1$

Iterative method for the diffusion equation

Recall $H = \frac{1}{h^2}\begin{pmatrix} S & I & & & \\ I & S & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & S & I \\ & & & I & S \end{pmatrix}$ where $S = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix}$

Since $D = \text{diag}(H) = -4h^2 I$, then eigenvalues of $E_J = I - D^{-1}A$ are

$$\lambda_{j,k}(E) = 1 - \sin^2\left(\frac{j\pi h}{2}\right) - \sin^2\left(\frac{k\pi h}{2}\right)$$

$$\begin{aligned} \lambda_{\max}(E_J) &\approx 1 - \frac{\pi^2 h^2}{2} \\ \lambda_{\max}(E_{GS}) &\approx \left(1 - \frac{\pi^2 h^2}{2}\right)^2 \approx 1 - \pi^2 h^2 \\ \lambda_{\max}(E_{SOR}) &\approx \frac{2}{1+\pi h} - 1 \approx 1 - 2\pi h \end{aligned}$$

Iterative methods on the diffusion equations

What about CG? Asymptotically the reduction factor is $\mu = \dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa+1}}$ where $\approx \kappa(H) = \dfrac{4}{\pi^2}h^{-2}$. Hence

$$\mu \approx \frac{\frac{2}{h\pi}-1}{\frac{2}{h\pi}+1} = 1 - \frac{2h\pi}{2+h\pi} \approx 1 - \pi h$$

Asymptotic behaviour:

method	error reduction factor	type
Jacobi	$1-5h^2$	sharp
Gauss-Seidel	$1-10h^2$	sharp
SOR with ω_{opt}	$1-6h$	sharp
SD (no precond)	$1-5h^2$	tight
CG (no precond)	$1-3h$	tight

Some results

- 2D discretization of the diffusion equation by FD, with $h = 0.005$, tol $= 10^{-8}$
- Results:

Method	iterations	CPU time	comments
Jacobi	100000	262.01	$\|\boldsymbol{r}_k\| > 10^{-6}$
Gauss Seidel	62207	186.36	
SOR	817	2.68	$\omega = 1.969 \approx \omega_{opt}$
SOR	1819	5.98	$\omega = 1.95$
SOR	1207	3.91	$\omega = 1.98$
CG	357	1.61	no prec
CG	146	1.38	IC(0) prec

- 2D discretization of the diffusion equation by FD, with $h = 0.0025$, tol $= 10^{-8}$
- Results

Method	iterations	CPU time	comments
SOR	1614	73.16	$\omega = 1.984 \approx \omega_{opt}$
CG	702	34.48	no prec
CG	244	22.84	IC(0) prec

Results and Comments

- Results with $h = 0.00125$, tol $= 10^{-8}$

Method	iterations	CPU time	comments
SOR	3572	605.40	$\omega = 1.992 \approx \omega_{opt}$
CG	1380	345.04	no prec
CG	451	199.35	IC(0) prec

Comments

- Theoretically $SOR(\omega_{opt})$ is the fastest method with CG very close.
- ω_{opt} is always very difficult to assess.
- Bounds on SOR is sharp while that on CG is tight.
- CG can be preconditioned while SOR can not.
- CG better than the estimates also when eigenvalues are spread into the spectral interval.
- Dependence of condition number on h^{-2} (and the number of iterations on h^{-1}) only alleviated by preconditioning.
- SOR is dramatically dependent on ω. Solving the FD-discretized diffusion equation, it takes advantage on the a priori knowledge of the spectral interval.

Iterative methods for general systems

- PCG is expected to converge only on spd matrices. Why? One reason is that for non spd matrices it is impossible to have at the same time orthogonality and a minimization property by means of a short term recurrence. PCG at iteration k minimizes the error on a subspace of size k and constructs an orthogonal basis using a short-term recurrence.
- Extensions of PCG for general nonsymmetric matrices:
  1. PCG method applied to the (spd) system $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$ (Normal equations). This system is often ill-conditioned. It is the case of symmetric matrices where $\kappa(A^T A) = \kappa(A^2) = (\kappa(A))^2$. Also difficult to find a preconditioner if $A^T A$ could not be explicitly formed.
  2. Methods that provide orthogonality + minimization by using a long-term recurrence (GMRES)
  3. Methods that provide (bi) orthogonality. Examples: BiCG, BiCGstab
  4. Methods that provide some minimization properties. Examples: QMR, TFQMR.

The GMRES method

Given an arbitrary nonzero vector $\boldsymbol{v}$, a Krylov subspace of dimension m is defined as

$$K_m(\boldsymbol{v}) = \text{span}\left(\boldsymbol{v}, A\boldsymbol{v}, A^2\boldsymbol{v}, \ldots, A^{m-1}\boldsymbol{v}\right)$$

The GMRES (Generalized Minimal RESidual) method finds the solution of the linear system

$$A\boldsymbol{x} = \boldsymbol{b}$$

by minimizing the norm of the residual $\boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m$ over all the vectors $\boldsymbol{x}_m$ written as

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{y}, \qquad \boldsymbol{y} \in K_m(\boldsymbol{r}_0)$$

where $\boldsymbol{x}_0$ is an arbitrary initial vector and K_m is the Krylov subspace generated by the initial residual.
First note that the basis

$$\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots, A^{m-1}\boldsymbol{r}_0\}$$

of K_m is “little linearly independent”.

Orthogonalization of the Krylov basis

Theorem. If the eigenvalues of A are such that $|\lambda_1| > |\lambda_2| \geq \cdots$ then $A^k \boldsymbol{r}_0 = \boldsymbol{v}_1 + O\left(\frac{|\lambda_2|}{|\lambda_1|}\right)^k$, with $\boldsymbol{v}_1$ the eigenvector corresponding to λ_1.

To compute a really independent basis for K_m we have to orthonormalize such vectors using the Gram-Schmidt procedure.

$\beta = \|\boldsymbol{r}_0\|, \ \boldsymbol{v}_1 = \dfrac{\boldsymbol{r}_0}{\beta}$

DO $k = 1, m$

1. $\boldsymbol{w}_{k+1} = A\boldsymbol{v}_k$
2. DO $j = 1, k-1$
3. $h_{jk} = \boldsymbol{w}_{k+1}^T \boldsymbol{v}_j$
4. $\boldsymbol{w}_{k+1} := \boldsymbol{w}_{k+1} - h_{jk}\boldsymbol{v}_j$
5. END DO
6. $h_{k+1,k} = \|\boldsymbol{w}_{k+1}\|; \quad \boldsymbol{v}_{k+1} = \boldsymbol{w}_{k+1}/h_{k+1,k}$

END DO

Once the Krylov subspace basis is computed, the GMRES method minimizes the norm of the residual onto

$$\boldsymbol{x}_0 + K_m = \boldsymbol{x}_0 + \sum_{j=1}^{m} z_i \boldsymbol{v}_i \tag{1}$$

Minimizing the residual

Theorem. The new vectors $\boldsymbol{v}_k$ satisfy:

$$A\boldsymbol{v}_k = \sum_{j=1}^{k+1} h_{jk}\boldsymbol{v}_j, \qquad k = 1, \ldots, m$$

Proof. In fact from step 4. of the Gram Schmidt algorithm we have

$$\boldsymbol{w}_{k+1} = A\boldsymbol{v}_k - \sum_{j=1}^{k} h_{jk}\boldsymbol{v}_j$$

Now substituting $\boldsymbol{w}_{k+1} = h_{k+1,k}\boldsymbol{v}_{k+1}$ we obtain

$$h_{k+1,k}\boldsymbol{v}_{k+1} = A\boldsymbol{v}_k - \sum_{j=1}^{k} h_{jk}\boldsymbol{v}_j$$

and the thesis holds.

Minimizing the residual

Now define

$$V_m = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots \boldsymbol{v}_m]$$

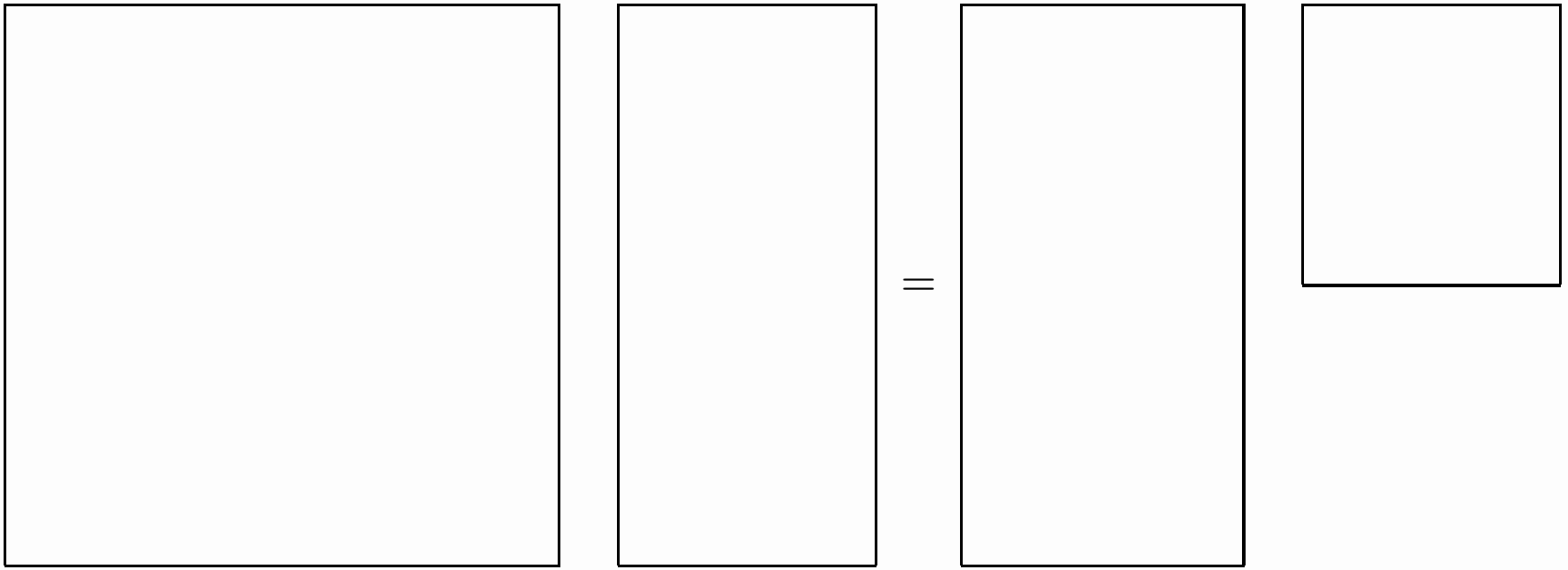
and $H_m = (h_{jk})$. The statement of the theorem reads

$$AV_m = V_{m+1} H_m$$

H_m is a rectangular $m+1 \times m$ matrix. It is a Hessenberg matrix since it has the following nonzero pattern:

$$H_m = \begin{pmatrix} h_{11} & h_{12} & \ldots & & \\ h_{21} & h_{22} & h_{23} & \ldots & \\ 0 & h_{32} & h_{33} & h_{34} & \\ 0 & 0 & \ldots & \ldots & \\ 0 & 0 & 0 & h_{m,m-1} & h_{mm} \\ 0 & 0 & 0 & 0 & h_{m+1,m} \end{pmatrix}$$

Minimizing the residual



A V_m $=$ V_{m+1} H_m

Minimizing the residual

Now we would like to minimize $\|\boldsymbol{r}_m\|$ among all the $\boldsymbol{x}_m$:

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \sum_{j=1}^{m} z_i \boldsymbol{v}_i$$

or

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m \boldsymbol{z}, \qquad \boldsymbol{z} = (z_1, \ldots, z_m)^T$$

Now

$$\begin{aligned} \boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m &= \boldsymbol{b} - A(\boldsymbol{x}_0 + V_m \boldsymbol{z}) = \\ &= \boldsymbol{r}_0 - AV_m \boldsymbol{z} = \\ &= \beta \boldsymbol{v}_1 - V_{m+1} H_m \boldsymbol{z} = \\ &= V_{m+1}(\beta \boldsymbol{e}_1 - H_m \boldsymbol{z}) \end{aligned} \tag{2}$$

where $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^T$.

Minimizing the residual

Recalling that $V_{m+1}^T V_{m+1} = I_{m+1}$:

$$\begin{aligned} \|\boldsymbol{r}_m\| &= \sqrt{\boldsymbol{r}_m^T \boldsymbol{r}_m} = \\ &= \sqrt{(\beta \boldsymbol{e}_1 - H_m \boldsymbol{z})^T V_{m+1}^T V_{m+1} (\beta \boldsymbol{e}_1 - H_m \boldsymbol{z})} = \\ &= \sqrt{(\beta \boldsymbol{e}_1 - H_m \boldsymbol{z})^T (\beta \boldsymbol{e}_1 - H_m \boldsymbol{z})} = \\ &= \| (\beta \boldsymbol{e}_1 - H_m \boldsymbol{z}) \| \end{aligned} \tag{3}$$

and hence $\boldsymbol{z} = \text{argmin} \|\beta \boldsymbol{e}_1 - H_m \boldsymbol{z}\|$.
Comments:

- H_m has more rows than columns then $H_m \boldsymbol{z} = \beta \boldsymbol{e}_1$ has (in general) no solutions.
- Minimization problem $\boldsymbol{z} = \text{argmin} \|\beta \boldsymbol{e}_1 - H_m \boldsymbol{z}\|$ is very small $m = 20, 50, 100$ as compared to the original size n. Computational solution of this problem will be very cheap.

Least square minimization

- Any (even rectangular) matrix H can be factorized as a product of an orthogonal matrix Q and an "upper triangular" matrix R (known as QR factorization.
- When H is not square the resulting R has as many final zero rows as the gap between rows and columns.
- Let us now factorize our H_m as: $H_m = QR$ (computational cost $O(m^3)$). Then, in view of the orthogonality of Q:

$$\min \|\beta \boldsymbol{e}_1 - H_m \boldsymbol{z}\| = \min \|Q \left(\beta Q^T \boldsymbol{e}_1 - R\boldsymbol{z}\right)\| = \min \|\boldsymbol{g} - R\boldsymbol{z}\|$$

where $\boldsymbol{g} = \beta Q^T \boldsymbol{e}_1$.

End of minimization

Matrix R takes on the form:

$$R = \begin{pmatrix} r_{11} & r_{12} & \dots & & \\ 0 & r_{22} & r_{23} & \dots & \\ & \dots & \dots & \dots & \\ & & & 0 & r_{mm} \\ & & & & 0 \end{pmatrix} \tag{4}$$

- The solution to $\min \|\boldsymbol{g} - R\boldsymbol{z}\|$ is simply accomplished by solving $\tilde{R}\boldsymbol{z} = \tilde{\boldsymbol{g}}$ where $\tilde{R}$ is obtained from R by dropping the last row and $\tilde{\boldsymbol{g}}$ the first m components of $\boldsymbol{g}$.
- This last system being square, small, and upper triangular, is easily and cheaply solved.
- Finally note that $\min \|\boldsymbol{r}_m\| = |g_{m+1}|$.

Algorithm

ALGORITHM: GMRES

Input: $x_0, A, b, k_{\max}$, toll

- $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0,\ k = 0,\ \rho_0 = \|\boldsymbol{r}_0\|,\ \beta = \rho_0,\ \boldsymbol{v}_1 = \dfrac{\boldsymbol{r}_0}{\beta}$
- WHILE $\rho_k >$ toll $\|b\|$ AND $k < k_{\max}$ DO
  1. $k = k + 1$
  2. $\boldsymbol{v}_{k+1} = A\boldsymbol{v}_k$
  3. DO $j = 1, k$
     $h_{jk} = \boldsymbol{v}_{k+1}^T \boldsymbol{v}_j$
     $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1} - h_{jk}\boldsymbol{v}_j$
     END DO
  4. $h_{k+1,k} = \|\boldsymbol{v}_{k+1}\|$
  5. $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1}/h_{k+1,k}$
  6. $\boldsymbol{z}_k = \text{argmin}\|\beta\boldsymbol{e}_1 - H_k\boldsymbol{z}\|$
  7. $\rho_k = \|\beta\boldsymbol{e}_1 - H_k\boldsymbol{z}_k\|$
- END WHILE
- $\boldsymbol{x}_k = \boldsymbol{x}_0 + V_k z_k$

Practical GMRES implementations.

- GMRES is optimal in the sense that it minimizes the residual over a subspace of increasing size (finite termination property).
- GMRES is a VERY computationally costly method.
  1. Storage: It needs to keep in memory all the vectors of Krylov basis. When the number of iterations becomes large (order of hundreds) the storage may be prohibitive.
  2. Computational cost. The cost of a Gram-Schmidt orthogonalization increases with the iteration number ($O(mn)$) so again problems arise when the number of iteration is high.
- Practical implementations fix a maximum number of vectors to be kept in memory, say p. After p iterations, $\boldsymbol{x}_p$ is computed and a new Krylov subspace is begin constructed starting from $\boldsymbol{r}_p$.
- In this way however, we loose optimality with consequent slowing down of the method.

Convergence

Assume for simplicity that A is diagonalizable that is

$$A = V \Lambda V^{-1}$$

where Λ is the diagonal matrix of eigenvalues and columns of V are normalized eigenvectors of A. Then

$$\|\boldsymbol{r}_k\| = \min_{P_k} \|V p_k(\Lambda) V^{-1}\| \cdot \|\boldsymbol{r}_0\| \leq \kappa(V) \min_{P_k} \|p_k(\Lambda)\| \cdot \|\boldsymbol{r}_0\|$$

or

$$\frac{\|\boldsymbol{r}_k\|}{\|\boldsymbol{r}_0\|} \leq \kappa(V) \min_{P_k} \max |p_k(\lambda_i)|$$

- Matrix V may be ill-conditioned
- We cannot simply relate convergence of GMRES with the eigenvalue distribution of A.

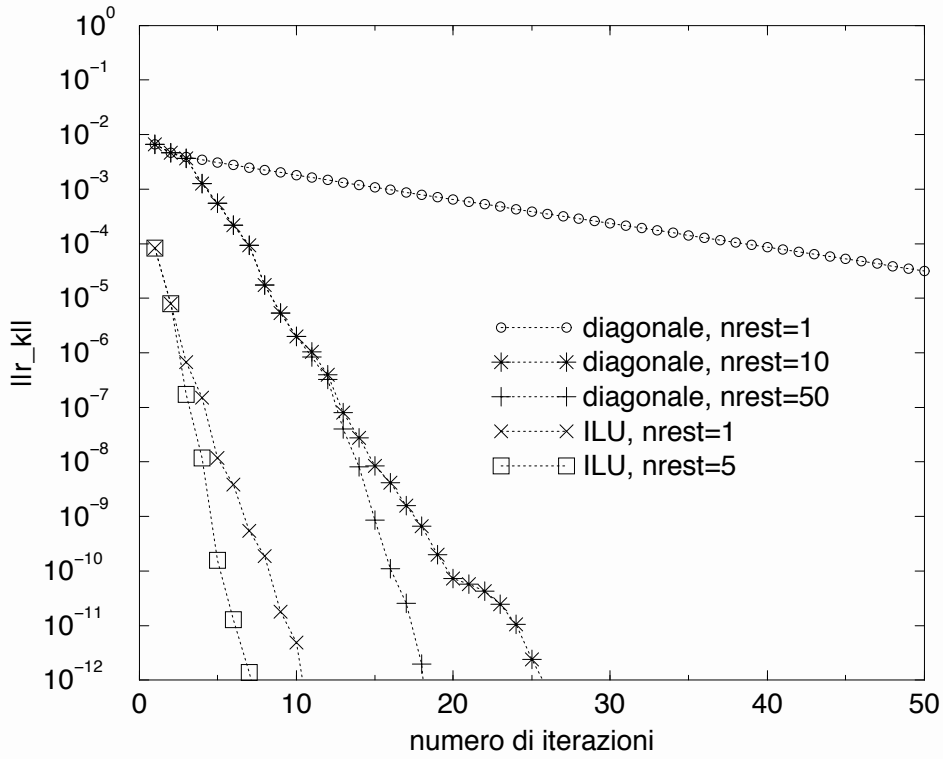
Preconditioning

- Also for general systems preconditioning is premultiplying the original system by a nonsingular matrix.
- Here preconditioning, more than reducing condition number of A is aimed at clustering eigenvalues away from the origin.
- Common general preconditioners.
  1. Diagonal: $M = \text{diag}(\|A^i\|)$, with A^i the ith row of A.
  2. ILU decomposition (based on pattern and/or on dropping tolerance).
  3. Approximate inverse preconditioners
- Computational cost. Instead of applying A one should apply $M^{-1}A$ to a vector (step 2. of algorithm) as

$$t = A\boldsymbol{v}_k, \qquad M\boldsymbol{v}_{k+1} = t$$

Numerical example

Convergence profile (semi-logarithmic plot) of GMRES as applied to a very small ($n = 27$) matrix with two preconditioners and some choices of the restart parameter.



Other iterative methods for nonsymmetric systems

- They are based on a fixed amount of work per iteration (short term recurrence)
- No a priori theoretical estimates on the error.
- Possibility of failure (breakdown) that can be alleviated through the use of *look-ahead* strategies.

The Lanczos method.

It is useful to underline connection between symmetric and nonsymmetric problems.

1. GMRES for spd matrices produces a tridiagonal matrix instead of an Hessenberg one (a symmetric Hessenberg is tridiagonal). This means that there is a three-term recurrence which implements Gram-Schmidt orthogonalization on a Krylov subspace generated by an spd matrix.
2. For A spd $V_m^T A V_m = T_m$ and it can be proved that the extremal eigenvalues of T_m converge quickly to the extremal eigenvalues of A. Moreover there is a strict connection between the Lanczos process and the CG method (entries of T_m can be computed from CG iterates). Knowing approximately the largest and the smallest eigenvalues is useful for an appropriate exit test for the CG method (recall 1st lecture on reliability of exit tests).
3. If A is nonsymmetric the two-sided Lanczos method can be employed to construct a biorthogonal basis for the Krylov subspaces corresponding to A and A^T by using a short term recurrence.

The Biconjugate Gradient method (BiCG).

Based on two-sided Lanczos process:

1. Given an arbitrary $\widehat{r_0}$ non orthogonal to $\boldsymbol{r}_0$, it constructs two sets of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in K(A, \boldsymbol{r}_0)$ and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n \in K(A^T, \widehat{r_0})$ s. t. $\boldsymbol{v}_i^T \boldsymbol{w}_j = 0,\ i \neq j$. In matrix form

$$AV_k = V_{k+1} T_{k+1,k}, \qquad A^T W_k = W_{k+1} \widehat{T}_{k+1,k}, \qquad V_k^T W_k = I_k$$

2. Write $\boldsymbol{x}_m = \boldsymbol{x}_0 + v_k \boldsymbol{y}_k$. From the several choices for vector $\boldsymbol{y}_k$ BiCG forces $\boldsymbol{r}_k = \boldsymbol{r}_0 - AV_k \boldsymbol{y}_k$ to be orthogonal to W_k. This leads to the equation

$$W_k^T \boldsymbol{r}_0 - W_k^T AV_k \boldsymbol{y}_k = 0$$

   By noticing that $W_k^T AV_k = T_k$ and that $W_k^T \boldsymbol{r}_0 = \beta \boldsymbol{e}_1, \beta = \|\boldsymbol{r}_0\|$, equation for $\boldsymbol{y}_k$ becomes

$$T_k \boldsymbol{y}_k = \beta \boldsymbol{e}_1$$

3. Practical implementation of BiCG algorithm implicitly performs an LDU factorization of matrix T_k (Also CG for spd matrices is equivalent to Lanczos process with LDL^T factorization of T_k),

BiCG Algorithm

ALGORITHM: BICONJUGATE GRADIENT

Input: $\boldsymbol{x}_0, \widehat{\boldsymbol{r}_0}, A, b, k_{\max}$, tol

- $\boldsymbol{r}_0 = \boldsymbol{p}_0 = b - A\boldsymbol{x}_0,\ k = 0, \widehat{\boldsymbol{p}_0} = \widehat{\boldsymbol{r}_0}$
- WHILE $\|\boldsymbol{r}_k\| > \text{tol}\ \|b\|$ AND $k < k_{\max}$ DO
  1. $\boldsymbol{z} = A\boldsymbol{p}_k,\ \boldsymbol{u} = A^T\widehat{\boldsymbol{p}_k}$
  2. $\alpha_k = \dfrac{\boldsymbol{r}_k^T\widehat{\boldsymbol{r}_k}}{\boldsymbol{z}^T\boldsymbol{p}_k}$
  3. $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$
  4. $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\boldsymbol{z}\ \widehat{\boldsymbol{r}_{k+1}} = \boldsymbol{r}_k - \alpha_k\boldsymbol{u}$
  5. $\beta_k = \dfrac{\boldsymbol{r}_{k+1}^T\boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T\boldsymbol{r}_k}$
  6. $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k\boldsymbol{p}_k\ ,\ \widehat{\boldsymbol{p}_{k+1}} = \widehat{\boldsymbol{r}_{k+1}} + \beta_k\widehat{\boldsymbol{p}_k}$
  7. $k = k + 1$
- END WHILE

BiCGstab

Drawbacks of the BiCG method:

1. Need to implement a matrix vector product also for the transpose of the matrix (this can be a problem: sometimes the matrix is not available explicitly but only the result of its application to a vector)
2. No local minimization property fulfilled in BiCG. This leads to sometimes large oscillations in the convergence profile.

First CGS (Conjugate Squared method) avoids using A^T then a subsequent improvement: the BiConjugate Gradient stabilized (BiCGstab) method is aimed at avoiding such an erratic behavior trying to produce a residual of the form

$$\boldsymbol{r}_k = \Psi_k(A)\phi_k(A)\boldsymbol{r}_0$$

where $\Psi_k(z) = \prod_{j=1}^{k}(1 - \omega_j z)$ and the ω's are chosen to minimize

$$\|\boldsymbol{r}_j\| = \|(1 - \omega_j A)\Psi_{j-1}(A)\phi_j(A)\boldsymbol{r}_0\|$$

BiCGstab

ALGORITHM: BICGSTAB

Input: $x_0, A, b, k_{\max}$, toll

- $r_0 = b - Ax_0,\ k = 0,$ choose $\hat{r}_0$ s. t. $\hat{r}_0^T \hat{r}_0 \neq 0$
- $\rho_0 = \alpha_0 = \eta_1 = 1;$
- WHILE $\|r_k\| >$ toll $\|b\|$ AND $k < k_{\max}$ DO
  1. $k = k + 1$
  2. $\rho_k = \hat{r}_0^T r_k$
  3. $\beta_k = (\rho_k / \rho_{k-1})(\alpha_{k-1} / \eta_k)$
  4. $p_k = r_k + \beta_k (p_{k-1} - \eta_k v_{k-1})$
  5. $v_k = A p_k$
  6. $\alpha_k = \rho_k / \hat{r}_0^T v_k$
  7. $s = r_k - \alpha_k v_k$
  8. $t = As$
  9. $\eta_{k+1} = s^T t / t^T t$
  10. $c_{k+1} = c_k + \alpha_k p_k + \eta_{k+1} s$
  11. $r_{k+1} = s - \eta_{k+1} t$
- END WHILE

Bibliography

Books on iterative methods

- Anne Greenbaum. Iterative Methods for Solving Large Linear systems.
- C. T. Kelley. Iterative Methods for Linear and Nonlinear Equations

Preconditioning

- M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey , Journal of Computational Physics, 182 (2002), pp. 418-477.

Conjugate gradient

- J. R. Shewchuk. An Introduction to the CG method without the agonizing pain.

Nonsymmetric linear systems

- Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, pages 856–869, (7) 1986.
- H. A. van der Vorst. Bi-CGstab: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, pages 631–644, (13) 1992.