NMPDE/ATSC 2025
Lecture 10

Unfortunately, the zoom recording of this lecture did not work as, for some reason, only the audio track was saved.

(Q. 4.5.2)

$$AU = F$$

Assuming A is SPD associated to the

$$(A\underline{v}, \underline{v}) = \mathcal{A}(v_h, v_h)$$

where $v_h = \sum_i v_i \varphi_i$; $\underline{v} = (v_i)_i$

v_h ↑ FE function; $\underline{v}$ ↑ coeffs

$$\chi_{sp}(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

Lemma: let $\mathcal{T}_h$ quasi-uniform, shape-regular. Then

$\exists\, C_1, C_2 > 0$:

$$C_1 h^d |\underline{v}|^2 \le \|v_h\|_0^2 \le C_2 h^d |\underline{v}|^2 \qquad \forall v_h \in V_h^k$$

Theorem: Same assumptions $\Rightarrow \chi_{sp}(A) = O(h^{-2})$

Proof:

$$\underbrace{R_A(\underline{v}) = \frac{(A\underline{v}, \underline{v})}{|\underline{v}|^2}}_{\text{Reileigh quotient}} = \frac{\mathcal{A}(v_h, v_h)}{|\underline{v}|^2} \underset{\text{continuity}}{\le} \gamma \frac{\|v_h\|_1^2}{|\underline{v}|^2}$$

$$\|v_h\|_1^2 = \|v_h\|_0^2 + |v_h|_1^2$$

Proposition (H^1-L^2 Inverse ineq.): Some ass. $\Rightarrow$ $\exists\, C_{inv} > 0$:

$$|v_h|_1^2 \leq C_{inv}\, h^{-2} \|v_h\|_0^2 \qquad \forall v_h \in V_h^k$$

$$\leq \left(1 + C_{inv} h^{-2}\right) \|v_h\|_0^2$$

$$R_A(\underline{v}) \leq \gamma \left(1 + C_{inv} h^{-2}\right) \frac{\|v_h\|_0^2}{|\underline{v}|^2} \leq C_2 \gamma h^d \left(1 + C_{inv} h^{-2}\right)$$

$$R_A(\underline{v}) = \frac{a(v_h, v_h)}{|\underline{v}|^2} \underset{\text{coersivity}}{\geq} \alpha_0 \frac{\|v_h\|_1^2}{|\underline{v}|^2} \geq \alpha_0 \frac{\|v_h\|_0^2}{|\underline{v}|^2} \geq C_1 \alpha_0 h^d$$

$$\underline{C_1 \alpha_0 h^d} \leq R_A(\underline{v}) = \frac{(A v, v)}{|v|^2} \leq C_2 \gamma h^d \left(1 + C_{inv} h^{-2}\right)$$

In particular $\leq \lambda_{min}(A) \leq \lambda_{max}(A) \leq$

$$\Rightarrow \chi_{sp}(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \leq \frac{C_2}{C_1} \frac{\gamma}{\alpha} \left(1 + C_{inv} h^{-2}\right) = O(h^{-2})$$

Remark: - independent from space dim, conditioning linked to discretization parameter h (how fine is the mesh)

- As the space dim grows, expect larger systems ...

(Q. Ch7)

SOLUTION OF $AU=F$

Let m = dim. of the system (= # DoF of FEM)

COST

① DIRECT SOLVERS,

e.g LU decomposition (Gauss elimination) $O(2/3\, m^3)$

Advantages:

- once the LU decomposition is computed, can solve for different F
- incomplete LU versions available
- if A Symmetric ⇒ Cholesky method $O(1/3\, m^3)$
- for Banded A, e.g stemming from FE discretisations cost reduces to $O(m^2)$
- tridiagonal A ⇒ Thomas algorithm $O(m)$

Rule of Thumb: use direct methods for $d \leq 2$.

② Iterative solvers: Produce a sequence of vectors $\{U^{(n)}\}_n \xrightarrow{n} U$ sol. of $AU = F$

- decompose $A = P - N$
- given a initial guess $U^{(0)}$, iterate

$$P U^{(n+1)} = N U^{(n)} + F$$

requiring many solutions w.r.t. matrix P

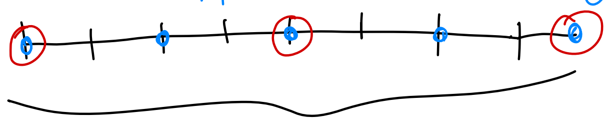
$\Rightarrow$ need
- P invertible
- P easy to invert

Preconditioner

Examples:
- A symetric : Conjugate gradient
- A general : Krylov subspaces (GMRES, biCG, biCGStab) for FEM sys
- Geometric Multigrid (GMG) $\rightsquigarrow O(m)$
  - To solve over given mesh $\mathcal{T}_h$,

GMG exploits a sequence of hyerarchical meshes coarsenings of $\mathcal{T}_h$

$H = 2h$ Coarsening

e.g. 

$\mathcal{T}_h$

$$\boxed{V_H^k \subset V_h^k}$$

The idea is to stop iteration for the solution of $A_h U_h = F_h$ before convergence and correct it by a (cheaper!) solve of $A_H \delta_H = F_H$.

An idea of the GMG algorithm steps

case two meshes $\mathcal{T}_h, \mathcal{T}_H$

V-cycle

1. $U_h^{(\ell)} = S_h(U_h^{(\ell-1)}, F_h)$ starting from some $U_h^{(0)}$, $\ell = 1, \dots, m_1$

   S_h: an iterative method

2. $r_h = F_h - A_h U_h^{(m_1)}$

3. $r_H = I_h^H r_h$ restrict residual

   I_h^H: some fine-to-coarse operator

4. $A_H \delta_H = r_H$ coarse solve

(5) $U_h^{(m_1+1)} = U_h^{(m_1)} + I_H^h \delta_H$ coarse-grid correction

coarse-to-fine op.

(6) $U_h^{(\ell)} = S_h\left(U_h^{(\ell-1)}, F_h\right)$ $\ell = m_1+1, \dots, m_1+m_2+1$

Iterative methods require many matrix vector multiplications ⇒ crucial to do such multiplications efficiently

One way to do this is by

MATRIX FREE approaches