Welcome to Ma221! (Apr 28)

Preconditioning to accelerate iterative methods

How to change $Ax=b$ in a cheap way to converge faster

Simplest: $M^{-1}Ax = M^{-1}b$

where $M \approx A$

(1) M^{-1} should be cheap to multiply by

(2) $M^{-1}A$ better conditioned than A

Idea straightforward for GMRES, which works for any general A,

But trickier for CG: M s.p.d and A s.p.d. then $M^{-1}A$ generally not

M s.p.d $\Rightarrow$ $M = Q\Lambda Q^T \Rightarrow M^{1/2} = Q\Lambda^{1/2}Q^T$ (also spd.)

Imagine applying CG to

$$(*) \quad \underbrace{\left(M^{-1/2}AM^{-1/2}\right)}_{\text{spd.}}\left(M^{1/2}x\right) = M^{-1/2}b$$

$M^{-1}A$ and $M^{-1/2}AM^{-1/2}$ are similar

$M^{\frac{1}{2}}\left(M^{-1}A\right)M^{-\frac{1}{2}} =$ "

$\Rightarrow$ same condition number

Preconditioned CG (equivalent to CG on (*))

$k=0$, $x(0)=0$, $r(0)=b$, $p(1)=M^{-1}b$, $y(0)=M^{-1}r(0)$

repeat

$k=k+1$

$z=Ap(k)$

$\nu(k)=(y(k-1)^T r(k-1))/(p(k)^T z)$

$x(k)=x(k-1)+\nu(k)\,p(k)$

$r(k)=r(k-1)-\nu(k)\cdot z$

$y(k)=M^{-1}r(k)$

$\mu(k+1)=(y(k)^T r(k))/(y(k-1)^T r(k-1))$

$p(k+1)=y(k)+\mu(k+1)\cdot p(k)$

until $\|r(k)\|_2$ small enough

Thm 6.9: equivalent to CG on (*)

(HW Q 6.14)

(see errata on page 317)

Where to get a good M?

Goals: (1) cheap to multiply b by M^{-1}

(2) $\text{cond}(M^{-1}A) \ll \text{cond}(A)$

lots of preconditioners, depending on A

(1) If A s.p.d. $M=\text{diag}(A_{11}, A_{22}, \ldots, A_{nn})$

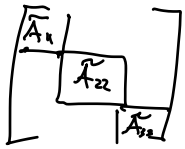
"Jacobi preconditioning"

$$M^{-1}A = I - R_J$$

$\rho(R_J) \ll 1$ ($\Leftrightarrow$ $M^{-1}A$ well conditioned)

Thm (van der Sluis, 1969): if M diagonal, then $\text{diag}(A_{11}\ldots A_{nn})$ within n of optimal

(2) $M = \mathrm{diag}(\tilde{A}_{11}, \tilde{A}_{22}, \ldots, \tilde{A}_{kk})$
where each $\tilde{A}_{ii}$ is a square block on diagonal



more expensive than diagonal M, but can converge faster.
"block Jacobi preconditioning"
each $\tilde{A}_{ii}$ could correspond to a different component of a physical model

Note: can apply to PAP^T, P permutation

(3) "Incomplete Cholesky"
compute a partial Cholesky decomp $LL^T \approx A$, by limiting fill-in in L, i.e. # new non zeros

Easiest: no new non zeros
$L_{ij} \neq 0 \Rightarrow A_{ij} \neq 0$
$\Rightarrow$ triangular solve with L costs $O(\mathrm{nnz}(L)) = O(\mathrm{nnz}(A)) =$ cost of matrix vector multiply by A

(4) a few steps of multigrid (perhaps just applied to $\tilde{A}_{ii}$)

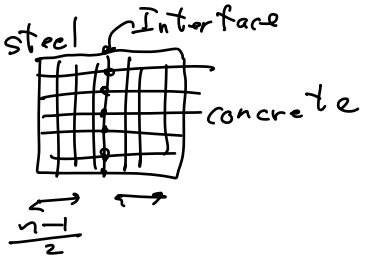
(5) Domain Decomposition (sec 6.10 text)

Ex: discretize FE problem on 2D mesh

Left is discretization of steel

Right " " " concrete

Interface in between



$n \times n$ mesh

n odd

$$A = \begin{array}{c|c|c} \frac{n(n-1)}{2} & \frac{n(n-1)}{2} & n \\ \hline A_{ss} & 0 & A_{si} \\ \hline 0 & A_{cc} & A_{ci} \\ \hline A_{is} & A_{ic} & A_{ii} \end{array}$$

$$A = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \hline A_{is}A_{ss}^{-1} & A_{ic}A_{cc}^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \hline 0 & 0 & S \end{bmatrix} \begin{bmatrix} A_{ss} & 0 & A_{si} \\ \hline 0 & A_{cc} & A_{ci} \\ \hline 0 & 0 & I \end{bmatrix}$$

Schur complement: $S = A_{ii} - A_{is}A_{ss}^{-1}A_{si} - A_{ic}A_{cc}^{-1}A_{ci}$

$$A^{-1} = \begin{bmatrix} A_{ss}^{-1} & 0 & -A_{ss}^{-1}A_{si} \\ 0 & A_{cc}^{-1} & -A_{cc}^{-1}A_{ci} \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & S^{-1} \end{bmatrix} \cdot$$

$$\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -A_{is}A_{ss}^{-1} & -A_{ic}A_{cc}^{-1} & I \end{bmatrix}$$

Need to multiply by approximation of A^{-1}
ie. M^{-1}

Useful when there are good preconditioners for A_{ss} and A_{cc} separately

Need to multiply by

$$(-A_{is}A_{ss}^{-1})x = -A_{is}(A_{ss}^{-1}x)$$

$\Rightarrow$ use preconditioner for A_{ss} to approximate $A_{ss}^{-1}x$

ditto for $A_{ic}\cdot A_{cc}^{-1}x$

Multiplying by S^{-1}:

Can multiply by S same way!

$$Sx = (A_{ii}x) - A_{is}(A_{ss}^{-1}(A_{si}x)) - A_{ic}(A_{cc}^{-1}(A_{ci}x))$$

uses preconditioners for A_{ss} and A_{cc}

Given Sx, use CG to multiply by S^{-1} ie. need all previous tricks

why is it faster to use CG on S versus original problem?

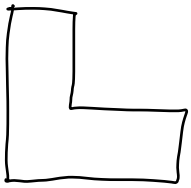
Often S much better conditioned than A

Thm: $\text{cond}(S) \leq \text{cond}(A)$ if A s.p.d.

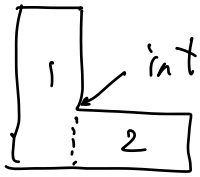
Eg: for Poisson $\kappa(S) = \sqrt{\kappa(A)}$

Called "nonoverlapping domain decomposition"

Also: "overlapping domain decomposition"

Ex: 

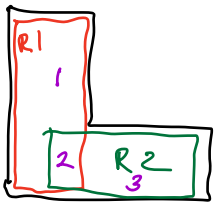
solve Poisson on L-shaped domain

nonoverlapping: 

use fast Poisson solver on each domain

disadvantage: data moves slowly across interface (same problem that multigrid addressed)

Solution: let domains overlap



$$A = \begin{bmatrix} A_{11} & A_{12} & \\ A_{21} & A_{22} & A_{23} \\ & A_{32} & A_{33} \end{bmatrix}$$

R1 over the first two rows; R2 over the last two rows

$R_1 = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ for rectangle 1

$R_2 = \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix}$ for rectangle 2

$$M^{-1} = \begin{bmatrix} R_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & R_2^{-1} \end{bmatrix}$$

"additive Schwartz"

"overlapping block Jacobi"

Apply idea from Gauss Seidel to use most recent data:

: Solve with R_1^{-1}, then use updated solution to solve with R_2^{-1}

"multiplicative Schwartz"