Ma221 Lecture 17

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

1. $M^{-1}$ cheap to multiply by
2. $M^{-1}A$ better conditioned than $A$

Straightforward for GMRES
But not CG: $M^{-1}A$ not spd in general
If $M$ spd $\Rightarrow M=QLQ^T \Rightarrow M^{\frac{1}{2}}=QL^{\frac{1}{2}}Q^T$

Imagine applying CG to $M^{-\frac{1}{2}}AM^{-\frac{1}{2}}$ where

\[ (M^{-1/2}AM^{-1/2})(M^{-1/2}x) = M^{-1/2}b \]

\[ \underbrace{\text{I}}_{\text{spd}} \]

$M^{-1}A$ and $M^{-1/2}AM^{-1/2}$ similar
equal condition numbers $\frac{d_{\max}}{d_{\min}}$

Preconditioned CG:

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

repeat

$k = k + 1$

$z = A p(k)$

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

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

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

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

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

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

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

see Thm 6.9 to see why this equivalent to CG with $M^{-1/2} A M^{-1/2}$
(see textbook errata on page 317)

Recall goals:

1. cheap to multiply by $M^T$
2. $M^{-1} A$ better conditioned than $A$

lots of preconditioners available,
best one depends on $A$
(1) If $A$ spd. has widely varying $\lambda_i$, $M = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$ good choice for preconditioning. $M^{-1}A = I - R_A$ so if $\rho(R_A)$ small $\Rightarrow M^{-1}A$ well-conditioned.

(2) More generally, $M = \text{diag}(\hat{A}_{11}, \ldots, \hat{A}_{zz})$ where $\hat{A}_{ii}$ is a diagonal block of $A$. More expensive than diagonal $M$, need to invert $\hat{A}_{ii}$, called block Jacobi preconditioning. $\hat{A}_{ii}$ can be chosen corresponding to different "domains" in a simulation (see domain decomposition later). By replacing $A$ by $P A P^T$ permutation so that $\hat{A}_{ii}$ can be formed by any group of rows and columns.

(3) "Incomplete Cholesky" $\Rightarrow$ compute partial Cholesky factorization, limiting nonzero pattern to
reduce cost, e.g. nonzeros of $L$
are only allowed where $A$ nonzero
$\Rightarrow$ solving with $L, L^T$ same
#flops as multiplying by $A$:
Same idea for non-spd $A$,
"incomplete LU" or ILU

(4) One or more $V$ cycles of Multigrid
as a preconditioner, given a
suitable $A$ or $\hat{A}$.

Ma221 Lecture 17 Segment 2
5) Domain Decomposition (see 6.10 of text)

Ex: Sparse on 2D mesh

Left is discretization of steel
Right " " concrete
Interface in between
\[ A = \begin{bmatrix}
\text{A}_{\text{ss}} & 0 & \text{A}_{\text{si}} \\
0 & \text{A}_{\text{cc}} & \text{A}_{\text{ci}} \\
\text{A}_{\text{is}} & \text{A}_{\text{ic}} & \text{A}_{\text{ic}}
\end{bmatrix} \]

\[ A = \begin{bmatrix}
I & 0 & 0 \\
0 & I & 0 \\
\text{A}_{\text{is}} & \text{A}_{\text{ss}}^{-1} & \text{A}_{\text{ic}}
\end{bmatrix} \begin{bmatrix}
I & 0 & 0 \\
0 & I & 0 \\
0 & 0 & I
\end{bmatrix}
\]

Schur complement

\[ S = \text{A}_{\text{ci}} - \text{A}_{\text{is}} \text{A}_{\text{ss}}^{-1} \text{A}_{\text{si}} - \text{A}_{\text{ic}} \text{A}_{\text{cc}}^{-1} \text{A}_{\text{ci}}^{-1}
\]

\[ A^{-1} = \begin{bmatrix}
\text{A}_{\text{ss}}^{-1} & \text{A}_{\text{ss}}^{-1} \text{A}_{\text{si}} \\
\text{A}_{\text{cc}}^{-1} & \text{A}_{\text{cc}}^{-1} \text{A}_{\text{ci}} \\
0 & I
\end{bmatrix} \begin{bmatrix}
I & 0 & 0 \\
0 & I & 0 \\
0 & 0 & S^{-1}
\end{bmatrix}
\]

use this factorization with available preconditioners for \text{A}_{\text{ss}} and \text{A}_{\text{cc}} to approximate \( A^{-1} \)

Need to multiply \((-\text{A}_{\text{is}} \text{A}_{\text{ss}}) \times = -\text{A}_{\text{is}} (A_{\text{ss}}^{-1} x)\)

\[ \Rightarrow \text{use preconditioner for } \text{A}_{\text{ss}} \text{ to multiply } A_{\text{ss}}^{-1} \text{ approximately} \]
ditto for $A^{-1} x$

Multiplying by $S^{-1}$:

Can multiply by $S$ same way

$$S x = A_{ii} x = A_{ii}^{-1} (A_{ii} x)$$
- $A_{ii}^{-1} (A_{ii} x)$

Given $S x$, can use eg CG to solve, ie. multiply by $S^{-1}$

Ie. use all previous methods to invert blocks as needed

Why is solving $S x = b$ with CG easier than solving $A x = b$?

Soften better conditioned than $A$
- (and much smaller, $n$ vs $n(n-1)/2$)

Poisson: $k(S) = O(N)$ vs $k(A) = O(N^2)$

This example called
- "non overlapping" domain decomposition

Overlapping domain decomposition also possible
Example: solve Poisson on “L-shaped” domain

Interface  use nonoverlapping DD as above

Disadvantage: data moves across interface “slowly”, same disadvantage that motivated Multigrid

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

\[
R_1 = \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\text{ for Rect 1}
\]

\[
R_2 = \begin{bmatrix}
A_{22} & A_{23} \\
A_{32} & A_{33}
\end{bmatrix}
\text{ for Rect 2}
\]

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

“additive Schwarz”, “overlapping block” Jacobi
Preconditioner solves each domain separately, adds solutions.

Same idea that led to Gauss-Seidel from Jacobi also works:

Solve with $R_1$, then use updated solution to solve with $R_2$

"multiplicative Schwartz"

(also using coarse grid with MG—has been proposed.)