Welcome back to Ma 221! Lecture 35, Nov 15

Splitting Methods for $Ax=b$

Goal: Given initial guess $x_0$
for solution of $Ax=b$, cheaply
compute sequence $x_0 \rightarrow A^{-1}b$

Def: Splitting of $A = M - K$, $M$ nonsingular

$Ax = b \iff Mx = Kx + b$

compute $x_{i+1}$ by solving $Mx_{i+1} = Kx_i + b$

(*) $x_{i+1} = M^{-1}Kx_i + M^{-1}b = Rx_i + c$

For (*) to work well we need:

1. $x_i$ should converge to $A^{-1}b$, for any $x_0$
2. Solving $Mx_{i+1} = Kx_i + b$ for $x_{i+1}$
   should be much cheaper than solving
   with $A$

Lemma: Let $\| \cdot \|$ be any operator norm

$\| R \| = \max_{x \neq 0} \frac{\| R x \|}{\| x \|}$

then if $\| R \| < 1$ (*) converges to $A^{-1}b$
for any $x_0$

Proof: Subtract $x = Rx + c$ from (*)

$x_{i+1} - x = R (x_0 - x) = R x_i (X_0 - x)$

$\| x_{i+1} - x \| \leq \| R \| \| x_i \| \| (X_0 - x) \|

\implies i \leq \frac{\ln \| R \|}{\ln \| x \|}$
Def: The spectral radius \( \rho(R) \) of \( R \) is
\[
\rho(R) = \max_{\lambda \text{ an eval of } R} |\lambda|
\]

Thm: \( x_iu_i = Rx_i + c \) converges to \( A^{-1}b \)
for all \( x_0 \) iff \( \rho(R) < 1 \)

Lemma: For all operator norms, \( \rho(R) \leq \|R\| \)
For any \( R \) and \( \epsilon > 0 \), there exist \( ||R||^* \)
such that \( ||R||^* \leq \rho(R) + \epsilon \)

Proof: To show \( \rho(R) \leq \|R\| \)
\[
\|R\| \leq \frac{\|Rx_i\|}{\|x_i\|} = \frac{\|\lambda x_i\|}{\|x_i\|} = |\lambda| = \rho(R)
\]
To construct \( ||R||^* \), use Jordan Form
\[
S^*RS = J = \begin{bmatrix}
\lambda_1 & 0 & \cdots & 0 \\
0 & \lambda_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_n
\end{bmatrix}
\]
\[
D = \begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_n
\end{bmatrix}
\]
\(D^tSD \rightarrow \) all 1's turn into \( \epsilon \)’s
\[
J_\epsilon = D^tS^*RSDB = \begin{bmatrix}
\epsilon & \epsilon & \cdots & \epsilon \\
\epsilon & \epsilon & \cdots & \epsilon \\
\vdots & \vdots & \ddots & \vdots \\
\epsilon & \epsilon & \cdots & \epsilon
\end{bmatrix}
\]
to define \( \| R \|_* \) define \( \| x \|_* = \| (S D^{-1}) x \|_1 \)

\[
\| R \|_* = \max_{x \neq 0} \frac{\| R x \|_*}{\| x \|_*} = \max_{x \neq 0} \frac{\| (S D)^{-1} R x \|_1}{\| (S D)^{-1} x \|_1} = \max_{y \neq 0} \frac{\| (S D)^{-1} R (C D) y \|_1}{\| y \|_1} = \max_{y \neq 0} \frac{\| J x \|_1}{\| y \|_1} = \| J \|_1 \leq p(R) + \varepsilon \quad \text{QED}
\]

Proof of Thm: if \( p(R) \geq 1 \) choose \( x_0 \) so that \( x_0 - x = \lambda e \) for \( \lambda \) where \( |\lambda| = p(R) \geq 1 \)

\[ x_i - x = R^i (x_0 - x) = \lambda^i (x_0 - x) \rightarrow \text{does not converge to 0} \]

if \( p(R) < 1 \), use Lemma to construct \( \| R \|_* \)

such that \( \| R \|_* = p(R) + \varepsilon \), choose \( \varepsilon \)

so \( \| R \|_* \leq 1 \)

\( \Rightarrow \) convergence \( \forall x_0 \) by first Lemma

Goal for \( A = M - K \): \( p(R) \) as small as possible

but still cheap to solve \( M x_i = K x_i + b \)

Ex: \( M = I \), \( K = I - A \) \( \Rightarrow \) solving for \( x_i \) very cheap, but no guarantees on \( p(R) = p(M^T K) = p(I - A) \)

Ex: \( K = 0 \) \( \Rightarrow R = 0 \) \( \Rightarrow \) converge in one step \( b \)

but need \( c = M^T b = A^T b \), no savings

Describe Jacobi's, Gauss-Seidel (GS)
Successive Over-relaxation (SOR)
\[ A = \begin{bmatrix} \mathbf{D} & -\mathbf{U}' \\ -\mathbf{L}' & \mathbf{D} \end{bmatrix} = \mathbf{D} \mathbf{L}' \mathbf{U}' = \mathbf{D} (\mathbf{I} - \mathbf{L} - \mathbf{U}) \]

**Jacobi:** In words: for \( j = 1 \) to \( n \), pick \( x_{i_1}(j) \) to exactly solve equation \( j \)

As a Loop: for \( j = 1 \) to \( n \)
\[
x_{i_1}(j) = \left( \mathbf{b}_i - \sum_{k \neq i} A_{ij} x_i(k) \right) / A_{ii}
\]

As a Splitting: \( \mathbf{A}_{i+1} = (\mathbf{L}' + \mathbf{U}') \mathbf{x}_i + \mathbf{b} \)
\[
\mathbf{A} = \mathbf{M} - \mathbf{K} = \mathbf{D} - (\mathbf{L}' + \mathbf{U}')
\]
\[
\mathbf{R}_j = \mathbf{M}^{-1} \mathbf{K} = \mathbf{D}^{-1} (\mathbf{L}' + \mathbf{U}') = \mathbf{L} + \mathbf{U}
\]

For 2D Poisson:
\[
\mathbf{T}_x \mathbf{V} + \mathbf{V} \mathbf{T}_y = \mathbf{h}^2 \mathbf{F}
\]
\( \mathbf{V}^{n \times n} \) unknowns
\( \mathbf{T}^n \) 2D Poisson

to get from \( \mathbf{V_i} \) to \( \mathbf{V}_{i+1} \)
for \( j = 1 : N \), for \( k = 1 : N \)
\[
\mathbf{V}_{i+1}(j,k) = (\mathbf{V}_i(j-1,k) + \mathbf{V}_i(j+1,k) + \mathbf{V}_i(j,k-1) + \mathbf{V}_i(j,k+1) + \mathbf{h}^2 \mathbf{F}(j,k)) / 4
\]

= “average” of 4 nearest neighbors and right hand side
Gauss-Seidel

In words: improve on Jacobi by using most recently updated values of \( x \)

As a loop for \( j = 1: n \)

\[
    x_{i+1}(j) = (b_j - \sum_{k \neq j} A_{jk} x_{i+1}(k)) \frac{\sum_{k \neq j} A_{kj}}{A_{jj}}
\]

As a Splitting \( A = (D - L^\top)U' = M - K \)

each step of G-S is a triangular solve

\[
    R_{GS} = M^{-1}K = (D - L^\top)^{-1}U'
    = (D(I-L)^{-1})U'
    = (I-L)^{-1}U
\]

In contrast to Jacobi, order of updating \( x_{i+1}(j) \) matters

for 2D Poisson

Natural Order (rowwise or columnwise), updating of \( U(j, k) \)

Red-Black Ordering

Red \( \equiv j + k \text{ even} \)
Black \( \equiv j + k \text{ odd} \)
Number all Red node nodes before Black node nodes. Red nodes only have Black neighbors (vice versa).

When updating Red nodes, can update them in any order (Parallelism), all Black nodes have “old data”. Then update all Black nodes (parallel) using updated Red data.

forall Red $(j,k) \; \hat{j}+k \text{ even}
\begin{align*}
V_{i+1}(j,k) &= (V_i(j-1,k) + V_i(j+1,k) \\
&\quad + V_i(j,k-1) + V_i(j,k+1) \\
&\quad + h^2 F(j,k))/4
\end{align*}

forall Black $(j,k) \; \hat{j}+k \text{ odd}
\begin{align*}
V_{i+1}(j,k) &= (V_{i+1}(j-1,k) + V_{i+1}(j+1,k) \\
&\quad + V_{i+1}(j,k-1) + V_{i+1}(j,k+1) \\
&\quad + h^2 F(j,k))/4
\end{align*}

$sQR$:
In words: Depends on parameter $w$.
Result of $sQR$ = weighted comb of old $k$ and result of GS

$x_{w,i+1}(j) = (1-w)x_{i}(j) + w x_{GS,i+1}(j)$

$w=1 \Rightarrow$ same as GS
$w<1 \Rightarrow$ “under relaxation”, not useful
$w>1 \Rightarrow$ “over relaxation”: good idea
since \( x_{i+1} \) is better than \( x_i \), go farther in same direction

Later will pick \( \mu \) optimally for Poisson

As a loop:

\[
\text{for } j = 1 \to n \\
\quad x_{i+1}^j = (1-\mu) x_i^j + \\
\quad \mu \left( b_j - \sum_{k \neq j} A_{jk} x_{i+1}^k \right) - \sum_{k \neq j} A_{jk} x_i^k \right) / A_{jj}
\]