Ma 221 Lecture 16

Chebyshev Polynomials:

Convergence analysis of CG

Accelerating SOR

Recall Krylov Subspace Methods:

seek "best" \( x_k \) approximating \( Ax = b \)

in \( \mathcal{X}_k(A, b) = \text{span} \{ b, Ab, A^2b, \ldots, A^k b \} \)

= \text{span} \{ \rho_{k-1}(A)b, \rho_{k-1} \}

polynomial, deg \( \leq k-1 \)

CG, "best" \( x_k \) minimized

\[ \|Ax_k - b\|^2_A = (Ax_k - b)^T A^{-1} (Ax_k - b) \]

\[ = (A \rho_{k-1}(A)b - b)^T A^{-1} (A \rho_{k-1}(A)b - b) \]

\[ = (\rho_k(A)b)^T A^{-1} (\rho_k(A)b) \]

\( \rho_k(A) = I - A \cdot \rho_{k-1}(A) \) polynomial

of degree \( \leq k \) with \( \rho_k(0) = 1 \)

\[ = b^T \rho_k(A)^2 A^{-1} b \]
\[ A \text{ spd} \Rightarrow A = Q \Lambda Q^\top, \quad y = Q^\top b \]
\[ = y^\top g_k(C\Lambda)^2 \Lambda^{-1} y \]
\[ = \max_i g_k(\lambda_i)^2 \Lambda^{-1} y_i^2 \]

So "best" \( x_k \) corresponds to "best" \( g_k(\cdot) \) i.e. it minimizes weighted sum of squared values \( g_k(\lambda_i)^2 \)

\[ \leq \max_i |g_k(\lambda_i)|^2 \cdot y^\top \Lambda^{-1} y \]
\[ = \max_i |g_k(\lambda_i)|^2 \cdot \|b\|^2 \Lambda^{-1} \]

\((\star)\) \[
\frac{\|Ax_k-b\|_{\Lambda^{-1}}}{\|b\|_{\Lambda^{-1}}} \leq \max_i |g_k(\lambda_i)|
\]

So seek \( g_k \) where \( g_k(0) = 1 \) and \( |g_k(\lambda_i)| \) small for all evals of \( A \)

\((A \text{ spd} \Rightarrow 0 \text{ not eval})\)

Use Chebyshev polynomials to construct a good \( g_k \)

Also use Chebyshev polynomials to accelerate splitting methods.
\[ x_k = R x_{k-1} + c, \text{ where exact } x: k = R x + c \]
\[ x_k - x = R (x_{k-1} - x) = R^k (x_0 - x) \]

Seek linear combination

\[ y_k = \sum_{i=0}^{k} c_i \cdot x_i \]

that is a better approximate solution than \( x_k \).

If \( x_0 = x \), then \( x_k = x \) \( \Rightarrow \)
\[ x = \sum_{i=0}^{k} c_i x_i \] \( \Rightarrow \) \[ 1 = \sum_{i=0}^{k} c_i \]

(\#2) \[ y_k - x = \sum_{i=0}^{k} c_i \cdot x_i - x \]
\[ = \sum_{i=0}^{k} c_i (x_i - x) \]
\[ = \sum_{i=0}^{k} c_i R^i (x_0 - x) \]
\[ = p_k (R) (x_0 - x) \]

Seek polynomial \( p_k (R) \)
that is small at evals of \( R \)
and \( p_k (1) = 1 \); again use (scaled)
Chebyshev polynomials

We only need to keep \( y_{-2} \) and \( y_{-1} \)
to compute \( y_k \), not \( x_0, \ldots, x_k \)
\( \Rightarrow \) cheap
Def: The \( m \)-th Chebyshev polynomial is defined by 3-term recurrence
\[
T_0(z) = 1, \quad T_1(z) = z, \\
T_m(z) = 2zT_{m-1}(z) - T_{m-2}(z)
\]

Lemma: (see Lemma 6.7 in text)
(a) \( T_m(1) = 1 \)
(b) \( T_m(2) = 2^m \cdot 2^m + O(2^{m-1}) \)
(c) \( T_m(\cos y) = \cos(m \cdot y) \), applies to \( T_m(z) \) if \( |z| \leq 1 \) \( \Rightarrow |T_m(z)| \leq 1 \) if \( |z| \leq 1 \)
(d) \( T_m(\cosh y) = \cosh(m \cdot y) \), applies to \( T_m(z) \) if \( |z| \leq \cosh(1) \) \( \Rightarrow |T_m(z)| \leq \cosh(1) \) if \( |z| \leq 1 \)
(e) If \( m \) is even (resp. odd) then \( T_m(z) \) is even (resp. odd) polynomial
(f) If \( |z| < 1 \), \( T_m(z) = \frac{1}{2} \left[ (z + \sqrt{1-z^2})^m + \frac{1}{(z - \sqrt{1-z^2})^m} \right] \)
(g) \( T_m(1+z) \geq 2.5(1+m \sqrt{2\varepsilon}) \) if \( \varepsilon > 0 \)

Property (e) extends property (d) to \( z \leq -1 \) since \( T_m(z) = (-1)^m T_m(-z) \)

Prop. (g) follows from prop(f)
\( \Rightarrow |T_m(z)| \) grows rapidly for \( |z| \geq 1 \) (Fig. 6.6 in text)
Apply to convergence of CG using Eq. 

\((*) \quad \frac{\|A x_k - b\|_{\infty}}{\|b\|_{\infty}} \leq \max_i |q_k(\lambda_i)|, \quad q_k(0) = 1\)

\(0 < \delta_{\min} \leq \delta_{\max}\) be range of evals of \(A\)

\(\delta_{\min} \leq \delta \leq \delta_{\max}\) 

\(-1 \leq \frac{\delta_{\max} + \delta_{\min} - 2\delta}{\delta_{\max} - \delta_{\min}} \leq 1\)

Def \(q_k(\delta) = \frac{\delta_{\max} + \delta_{\min} - 2\delta}{\delta_{\max} - \delta_{\min}}\)

\(q_k(0) = 1\) and for \(\delta_{\min} \leq \delta \leq \delta_{\max}\)

\(|q_k(\delta)| \leq \frac{1}{\delta_{\max} - \delta_{\min}}\), by prop(c)

\(= \frac{1}{\delta_{\max} - \delta_{\min}}\), \(\kappa = \frac{\delta_{\max}}{\delta_{\min}}\)

\(= \frac{1}{\kappa - 1}\)

\(= \frac{1}{\kappa (1 + \frac{2}{\kappa - 1})}\)

\(\leq \frac{2}{1 + \frac{2}{\kappa - 1}}\) by prop(g)
by (\ref{eq:cg-convergence}), need $k=O(\sqrt{k})$ steps of CG to reduce the error by a constant factor.

Ma221 Lecture 16 Segment 2

Use polynomials to accelerate splitting methods like SOR(w)

Need to pick polynomial $p_k(z)$:

$p_k(z)$ small for $z$ an eval of $R$ and $p_k(1) = 1$

If splitting method converges, $\Rightarrow p(R) < 1$

Chebyshev properties apply to real $z \Rightarrow$ assume $R$ real evens

Let $p$ satisfy $-1 < p \leq \min(R) \leq \max(R) \leq p < 1$

Need $\min$ and $\max$ to implement algorithm, or a bound $p$
This limits applicability, but sometimes, eg Model Problem, we do know $p$

$$p_k(z) = \frac{T_k(z \rho)}{T_k(1 \rho)} = 1 \text{ at } z = 1$$

if $2k \rho \Rightarrow |p_k(z)| \leq \frac{1}{T_k(1 \rho)}$

How much faster can this converge than original $x_{k+1} = R x_k + \epsilon$?

$p = 1 - \varepsilon$, use prop (g) of Lemma

$$\frac{1}{T_k(1 \rho)} = \frac{1}{T_k(1 \rho)}$$

$$\leq 2/(1 + k \sqrt{2\varepsilon/(1-\varepsilon)})$$

$$\sim 2(1 - k \sqrt{2\varepsilon})$$

Contrast with original splitting method $p^k = (1 - \varepsilon)^k \sim 1 - k \varepsilon$

$\Rightarrow$ Chebyshev acceleration can reduce #iterations by square root, an asymptotic improvement
Implement cheaply, using 3-term recurrence

\[ T_{k+1}(z) = 2zT_k(z) - T_k(z) \]

scale it to get recurrence for \( P_{k+1}(z) \)

\[ N_k = 1 / T_k(1/\rho) \]

\[ P_{k+1}(z) = N_{k+1} T_{k+1}(z/\rho) \]
\[ = N_{k+1} \left( 2(\eta/\rho)T_k(z/\rho) - T_k(z/\rho) \right) \]
\[ = N_{k+1} \left( 2(\eta/\rho) \frac{P_k(z)}{N_k} - \frac{P_{k-1}(z)}{N_{k-1}} \right) \]
\[ = \left( \frac{2N_{k+1}}{N_k} \right) P_k(z) - \left( \frac{N_{k+1}}{N_{k-1}} \right) P_{k-1}(z) \]
\[ = \alpha_k P_k(z) + \beta_k P_{k-1}(z) \]

Note that \( \frac{1}{N_k} = T_k(1/\rho) \) can be computed by 3-term recurrence, so \( \alpha_k, \beta_k \) cheap

Apply 3-term recurrence to (**)

\[ y_{k+1} - x = P_{k+1}(R) \cdot (x_0 - x) \]
\[ = \alpha_k R P_k(R)(x_0 - x) + \beta_k P_{k-1}(R)(x_0 - x) \]
\[ = \alpha_k R (y_k - x) + \beta_k (y_{k-1} - x) \]
\[ = \alpha_k R y_k + \beta_k y_{k-1} - \alpha_k R x - \beta_k x \]
\[ y_{k+1} = \alpha_k R y_k + B_k y_{k-1} - \alpha_k R x - \beta_k x + x \]

simplify last 3 terms using \( x = R x + c \)
\[-\alpha_k R x - \beta_k x + x = -\alpha_k (x - c) - \beta_k x + x \]
\[= (\alpha_k - \beta_k + 1)x + \alpha_k c \]
\[= 0 \]
\[= \alpha_k c \]

Yields final algorithm:
\[ y_{k+1} = \alpha_k R y_k + B_k y_{k-1} + \alpha_k c \]

cost is same as original splitting method

All this assumed \( R \) has real evals. and tight bound on \( \rho(R) \).

Try on Jacobi for model problem:
\( R_j \) was symmetric,
\[ \rho(R_j) = \cos\left(\frac{\pi}{N+1}\right) - \frac{\pi^2}{2(N+1)^2} = 1 - O\left(\frac{1}{N^2}\right) \]
\[\Rightarrow \] Jacobi takes \( \Theta(N^2) \) iterations to converge vs \( O(N) \) for Chebyshev, a big improvement
But we know SOR(\omega_{opt}) does \( O(N) \) iterations.

Try to apply Chebyshev to SOR(\omega_{opt})

But \( R_{sor(w)} \) has complex evals

Clever fix: Apply SOR(\omega) twice per iteration, first in usual order of updates, second in reverse order. Effectively "symmetrizes" SOR(w), yielding SSOR(\omega), symmetric SOR, with \( R_{ssor(w)} \) having real evals.

\( \omega_{opt} \) differs for SSOR(\omega) from SOR(\omega), but still have a good estimate, with

\[
P(R_{ssor(\omega_{opt})}) = 1 - O(\frac{1}{N})
\]

Apply Chebyshev to converge in \( O(\sqrt{N}) \) iterations instead of \( O(N) \).

For \( N \times N \) Poisson, need \( O(N^2 \sqrt{N}) = O(n^{5/4}) \) flops, where \( n = N^2 \), versus \( O(n^{3/2}) \) for SOR(\omega_{opt}).