Welcome to Ma221! Lecture 27, Fall24

Chebyshev Polynomials

Explain Convergence of CG
Accelerating Splitting Methods

Recall Krylov Subspace Methods
seek the "best" approximation $x_k \approx A^{-1}b$
in $\mathcal{K}_k(A,b) = \text{span}\{b, Ab, \ldots, A^{k-1}b\}$
$= \{p_{k-1}(A)b, \text{ deg of } p_{k-1} \le k-1\}$

CG: "best" x_k minimizes $r_k = Ax_k - b$

$$\|r_k\|_{A^{-1}}^2 = \|Ax_k - b\|_{A^{-1}}^2$$
$$= (Ax_k - b)^T A^{-1}(Ax_k - b)$$
$$= (Ap_{k-1}(A)b - b)^T A^{-1}(Ap_{k-1}(A)b - b)$$
$$= (q_k(A)b)^T A^{-1}(q_k(A)b)$$

$q_k(A) = I - A\cdot p_{k-1}(A)$
$=$ polynomial of deg k
with $q_k(0) = 1$

$$= b^T q_k(A) A^{-1} q_k(A) b$$
$$= b^T (q_k(A))^2 A^{-1} b$$

A s.p.d $\Rightarrow A = Q \Lambda Q^T \quad y = Q^T b$

$$= y^T q_k^2(\Lambda) \Lambda^{-1} y$$

$$= \sum_{i=1}^{n} q_k^2(\lambda_i) \cdot \lambda_i^{-1} \cdot y_i^2$$

so "best" x_k corresponds to best $q_k(x)$ i.e minimizes weighted sum of squares $q_k^2(\lambda_i)$ over evals of A

$$\leq \max_i q_k^2(\lambda_i) \cdot \sum_{i=1}^{n} \lambda_i^{-1} y_i^2$$

$$= \max_i q_k^2(\lambda_i) \cdot y^T \Lambda^{-1} y$$

$$= \max_i q_k^2(\lambda_i) \cdot \|b\|_{A^{-1}}^2$$

$$\frac{\|Ax_k - b\|_{A^{-1}}}{\|b\|_{A^{-1}}} \leq \max_i q_k(\lambda_i)$$

so seek q_k where $q_k(0) = 1$ and $|q_k(\lambda_i)|$ small for all evals λ_i of A

use Chebyshev polynomials to find good $q_k(x)$

Also use Chebyshev polynomials to accelerate splitting methods

$Ax = b$

$A = M - K$

$\rightarrow \quad Mx = Kx + b$

$x = M^{-1} K x + M^{-1} b$

$= R x + c$

$\rightarrow \quad x_{k+1} = R x_k + c$

converges if $\rho(R) < 1$, or $\|R\| < 1$

$x_k - x = R(x_{k-1} - x)$

$= R^k (x_0 - x)$

Seek $y_k = \sum_{i=0}^{k} c_{ki} x_i$ that converges to x faster than x_k

If $x_0 = x \Rightarrow x_i = x \Rightarrow$

$x = \sum_{i=0}^{k} c_{ki} x \quad \Rightarrow \quad 1 = \sum_{i=0}^{k} c_{ki}$

$y_k - x = \sum_{i=0}^{k} c_{ki} (x_i - x)$

$= \sum_{i=0}^{k} c_{ki} R^i (x_0 - x)$

(**) $y_k - x = p_k(R)(x_0 - x)$

Seek polynomial $p_k(R)$ that is small at evals of R

and $p_k(1) = 1 = \sum_{i=0}^{k} c_{ki}$

again, will use (scaled) Chebyshev polynomials to get $p_k(x)$

Don't need to keep $x_0, x_1, \ldots, x_k$ to compute y_k, just y_{k-1} and y_{k-2}

Def: The m^{th} Chebyshev Polynomial is defined by 3-term recurrence

$$T_0(z) = 1, \quad T_1(z) = z$$

$$T_m(z) = 2 \cdot z \cdot T_{m-1}(z) - T_{m-2}(z)$$

eg $T_2(z) = 2z^2 - 1$ etc

Lemma (Lemma 6.7 in text)

(a) $T_m(1) = 1$

(b) $T_m(z) = 2^{m-1} z^m + O(z^{m-1})$

(c) $T_m(\cos y) = \cos(m \cdot y)$

applies when $|z| \le 1 \Rightarrow$

if $|z| \le 1$ then $|T_m(z)| \le 1$

(d) $T_m(\cosh y) = \cosh(m \cdot y)$

applies if $|z| \ge 1$, then $|T_m(z)| \ge 1$

(e) if m is even (resp odd) then

$T_m(z)$ is even (resp odd) polynomial

$T_m(-z) = (-1)^m T_m(z)$

(f) if $|z| \ge 1$

$$T_m(z) = \frac{1}{2}\left[(z + \sqrt{z^2-1})^m + \frac{1}{(z + \sqrt{z^2-1})^m}\right]$$

(g) $T_m(1+\varepsilon) \geq .5\left(1+m\sqrt{2\varepsilon}\right)$ if $\varepsilon > 0$

(see Fig 6.6 in text)

Apply to convergence analysis of CG

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

Let $0 < \lambda_{min} \leq \lambda_{max}$ be. evals of A

$$\lambda_{min} \leq z \leq \lambda_{max} \Rightarrow$$

$$-1 \leq \frac{\lambda_{max} + \lambda_{min} - 2z}{\lambda_{max} - \lambda_{min}} \leq 1$$

$$\Rightarrow \text{let } q_k(z) = \frac{T_k\left(\frac{\lambda_{max}+\lambda_{min}-2z}{\lambda_{max}-\lambda_{min}}\right)}{T_k\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)}$$

$q_k(0) = 1$ and for $\lambda_{min} \leq z \leq \lambda_{max}$

$$|q_k(z)| \leq \frac{1}{T_k\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)} \quad \text{by part c in Lemma}$$

$$= \frac{1}{T_k\left(\frac{\kappa+1}{\kappa-1}\right)} \qquad \kappa = \frac{\lambda_{max}}{\lambda_{min}}$$

$$= \frac{1}{T_k\left(1+\frac{2}{\kappa-1}\right)}$$

$$\leq \frac{2}{1 + \frac{2k}{\sqrt{\kappa - 1}}}$$ by part g of Lemma

$\Rightarrow$ need $k = O(\sqrt{\kappa})$ steps to converge

worst case bound, CG can do better if evals are clustered

Use polynomials to accelerate splitting methods like SOR

Need a polynomial $p_k(z)$

$p_k(z)$ small for z = eval of R

and $p_k(1) = 1$

If splitting method converges, need all $|\lambda_i| < 1$ i.e $\rho(A) < 1$ spectral radius

Chebyshev poly. properties assume z real,

$\Rightarrow$ assume R has real evals

let ρ satisfy

$$-1 < -\rho \leq \lambda_{min}(R) \leq \lambda_{max}(R) \leq \rho < 1$$

Need ρ to use this acceleration method,

use Model problem as example

limits applicability, sometimes works

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

if $|z| \le \rho \Rightarrow |\rho_k(z)| \le \frac{1}{|T_k(1/\rho)|}$

How much faster can this converge than $x_{k+1} = Rx_k + c$?

Let $\rho = 1-\varepsilon \quad 0<\varepsilon<<1$

$$\frac{1}{|T_k(1/\rho)|} = \frac{1}{|T_k(\frac{1}{1-\varepsilon})|}$$

$$= \frac{1}{|T_k(1+\frac{\varepsilon}{1-\varepsilon})|}$$

$$\le \frac{2}{1+k\sqrt{2\varepsilon/(1-\varepsilon)}} \quad \text{part g of Lemma}$$

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

control to original splitting method

$$\rho^k = (1-\varepsilon)^k \sim 1-k\varepsilon$$

i.e. need $\sqrt{\ }$ as many steps to converge vs original splitting method

— an asymptotic improvement

Need 3-term recurrence to compute y_k:

$$T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z)$$

scale to get recurrence for $\rho_k(z)$

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

$$\begin{aligned}
\rho_{k+1}(z) &= \nu_{k+1} T_{k+1}(z/\rho) \\
&= \nu_{k+1}\left(2\left(\frac{z}{\rho}\right)T_k\left(\frac{z}{\rho}\right) - T_{k-1}\left(\frac{z}{\rho}\right)\right) \\
&= \nu_{k+1}\left(2\left(\frac{z}{\rho}\right)\frac{\rho_k(z)}{\nu_k} - \frac{\rho_{k-1}(z)}{\nu_{k-1}}\right) \\
&= \underbrace{\frac{2\nu_{k+1}}{\rho\cdot\nu_k}}\cdot z\cdot\rho_k(z) - \underbrace{\left(\frac{\nu_{k+1}}{\nu_{k-1}}\right)}\rho_{k-1}(z) \\
&= \alpha_k \cdot z\cdot\rho_k(z) + \beta_k\cdot\rho_{k-1}(z)
\end{aligned}$$

Note that $\frac{1}{\nu_k} = T_k(1/\rho)$ can be computed cheaply by 3-term recurrence for $T_k(\frac{1}{\rho})$, so α_k and β_k cheap

(cont) $$\begin{aligned}
y_{k+1} - x &= \rho_{k+1}(R)(x_0 - x) \\
&= \alpha_k R\rho_k(R)(x_0-x) + \beta_k \rho_{k-1}(R)(x_0-x) \\
&= \alpha_k R(y_k - x) + \beta_k(y_{k-1}-x) \\
&= \alpha_k R y_k + \beta_k y_{k-1} - \alpha_k R x - \beta_k x
\end{aligned}$$

$$y_{k+1} = \alpha_k R y_k + \beta_k y_{k-1} - \alpha_k R x - \beta_k x + x$$

simplify last 3 terms using $x = Rx + c$

$$\Rightarrow \quad -\alpha_k R x - \beta_k x + x = -\alpha_k(x-c) - \beta_k x + x$$

$$= (\underbrace{-\alpha_k - \beta_k + 1}_{=0})x + \alpha_k c = \alpha_k c$$

plug in $z=1$ into $\rho_{k+1}(z) = \alpha_k z \rho_k(z) + \beta_k \rho_{k-1}(z)$

$$\rightarrow \quad 1 = \alpha_k + \beta_k$$

Final algorithm:

$$y_{k+1} = \alpha_k R y_k + \beta_k y_{k-1} + \alpha_k c$$

cost ~ same as original splitting method

Apply to Model Problem: 2D Poisson

Jacobi: R_J has all real evals, symmetric

$$\rho(R_J) = \cos\left(\frac{\pi}{N+1}\right) \approx 1 - \frac{\pi^2}{2(N+1)^2}$$

$$= 1 - O\left(\frac{1}{N^2}\right)$$

$\Rightarrow$ Jacobi takes $O(N^2)$ iterations to converge vs $O(N)$ for Chebyshev, an asymptotic improvement

But we know SOR(ω_{opt}) does $O(N)$ iterations

$\Rightarrow$ apply Chebyshev to SOR(ω_{opt})

But $R_{SOR(\omega_{opt})}$ has some complex evals

Clever fix: run SOR(ω_{opt}) twice

forward (Red then Black) then backward (Black then Red)

called Symmetric SOR(ω), or SSOR(ω)

makes $R_{SSOR(\omega)}$ symmetric

How to pick ω? still a good estimate is known for Poisson: $\rho(R_{SSOR(\omega_{opt})}) = 1 - O(\frac{1}{N})$

$\Rightarrow$ can apply Chebyshev to accelerate $SSOR(\omega_{opt}) \Rightarrow$ only need $\sqrt{N}$ iterations to converge

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

For 3D Poisson on $N \times N \times N$ mesh, $n = N^3$
need $O(N^3 \cdot \sqrt{N}) = O(n^{7/6})$ flops, closer to lower bound $O(n)$

Finished explaining all rows in table of algorithms for Poisson (in Lecture 22)

Figure 6.8 in text: Decision Tree for choosing a KSM depending on matrix properties