Ma 221 Lecture 16

Chebyshev Polynomials:

Convergence analysis of CG

Accelerating SOR

Recall Krylov Subspace Methods:

seek "best" x_k approximating $Ax=b$

in $\mathcal{K}_k(A,b) = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}$

$= \text{span}\{p_{k-1}(A)b,\ p_{k-1}$ polynomial, $\deg \le k-1\}$

CG, "best" x_k minimized

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

$$= (\underbrace{Ap_{k-1}(A)b - b}_{})^T A^{-1}(Ap_{k-1}(A)\cdot b - b)$$

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

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

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

A spd $\Rightarrow$ $A = Q \Lambda Q^T$, $y = Q^T b$

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

$$= \sum_i q_k(\lambda_i)^2 \lambda_i^{-1} y_i^2$$

So "best" x_k corresponds to "best" $q_k(\cdot)$ ie. it minimizes weighted sum of squared values $q_k(\lambda_i)^2$

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

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

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

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

(A spd $\Rightarrow$ 0 not eval)

Use Chebyshev polynomials to construct a good q_k

Also use Chebyshev polynomials to accelerate splitting methods.

$x_k = R x_{k-1} + c$, where exact x: $x = Rx + c$

$x_k - x = R(x_{k-1} - x) = R^k (x_0 - x)$

Seek linear combination

$$y_k = \sum_{i=0}^{k} c_{ki} \cdot x_i$$

that is a better approximate solution than x_k

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

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

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

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

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

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

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_{k-2} and y_{k-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

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

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

Lemma: (see Lemma 6.7 in text)

(a) $T_m(1)=1$

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

(c) $T_m(\cos y) = \cos(m\cdot y)$; applies to $T_m(z)$ if $|z|\le 1 \Rightarrow |T_m(z)|\le 1$ if $|z|\le 1$

(d) $T_m(\cosh y) = \cosh(m\cdot y)$, applies to $T_m(z)$ if $z\ge 1$, $\Rightarrow T_m(z)\ge 1$ if $z\ge 1$

(e) If m is even (resp odd) then $T_m(z)$ is even (resp odd) polynomial

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

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

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

Prop. (g) follows from prop (f)

$\Rightarrow |T_m(z)|$ grows rapidly for $|z|\ge 1$ (Fig 6.6 in text)

Apply to convergence of CG using (*)

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

$0 < \lambda_{min} \le \lambda_{max}$ be range of evals of A

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

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

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

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

$$|q_k(z)| \le \frac{1}{T_k\left(\frac{\lambda_{max} + \lambda_{min}}{\lambda_{max} - \lambda_{min}}\right)} \text{, by prop(c)}$$

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

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

$$\le \frac{2}{1 + \frac{2k}{\sqrt{\kappa - 1}}} \quad \text{by prop (g)}$$

by (*), need $k = O(\sqrt{\kappa})$ steps of CG to reduce the error by a constant factor

---

Ma221 Lecture 16 Segment 2

Use polynomials to accelerate splitting methods like SOR(ω)

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, $\Rightarrow \rho(R) < 1$,

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

Let ρ satisfy

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

Need λ_{min} and λ_{max} to implement algorithm, or a bound ρ

This limits applicability, but sometimes, eg Model Problem, we do know ρ

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

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

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

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

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

$$= 1/T_k\left(1 + \frac{\varepsilon}{1-\varepsilon}\right)$$

$$\le 2/\left(1 + k\sqrt{2\varepsilon/(1-\varepsilon)}\right)$$

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

Contrast with original splitting method

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

$\Rightarrow$ Chebyshev acceleration can reduce # iterations by square root, an asymptotic improvement

Implement cheaply, using 3-term recurrence

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

scale it to get recurrence for $p_{k+1}(z)$

$$\mu_k = 1/T_k(1/\rho)$$

$$\begin{aligned}
p_{k+1}(z) &= \mu_{k+1} T_{k+1}(z/\rho) \\
&= \mu_{k+1}\left(2(z/\rho)T_k(z/\rho) - T_{k-1}(z/\rho)\right) \\
&= \mu_{k+1}\left(2(z/\rho)\frac{p_k(z)}{\mu_k} - \frac{p_{k-1}(z)}{\mu_{k-1}}\right) \\
&= \underbrace{\left(\frac{2\mu_{k+1}}{\rho\mu_k}\right)}z p_k(z) - \underbrace{\left(\frac{\mu_{k+1}}{\mu_{k-1}}\right)}p_{k-1}(z) \\
&= \alpha_k \; z p_k(z) + \beta_k p_{k-1}(z)
\end{aligned}$$

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

Apply 3-term recurrence to (**)

$$\begin{aligned}
(**) \quad y_{k+1} - x &= p_{k+1}(R)\cdot(x_0 - x) \\
&= \alpha_k R\, \underline{p_k(R)(x_0 - x)} + \beta_k \underline{p_{k-1}(R)(x_0 - x)} \\
&= \alpha_k R\,(y_k - x) + \beta_k (y_{k-1} - x) \\
&= \alpha_k R\, y_k + \beta_k y_{k-1} - \alpha_k R x - \beta_k x
\end{aligned}$$

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

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

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

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

$$= \alpha_k \cdot c$$

Yields final algorithm:

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

cost $\sim$ 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,

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

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

But we know $SOR(\omega_{OPT})$ does $O(N)$ iterations.

Try to apply Chebyshev to $SOR(\omega_{OPT})$

But $R_{SOR(\omega)}$ has complex evals

Clever fix: Apply $SOR(\omega)$ twice per iteration, first in usual order of updates, second in reverse order.

Effectively "symmetrizes" $SOR(\omega)$, yielding $SSOR(\omega)$, symmetric SOR, with $R_{SSOR(\omega)}$ having real evals.

ω_{OPT} differs for $SSOR(\omega)$ from $SOR(\omega)$, but still have a good estimate, with

$$\rho(R_{SSOR(\omega_{OPT})}) = 1 - O\left(\frac{1}{N}\right)$$

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 $SOR(\omega_{OPT})$