Forgot to say last lecture:

each iteration of Chebyshev only requires

- 1 multiplication of a vector by A, and
- a constant number of vector operations.

It produces x_t from Ax_{t-1} and x_{t-2}.

Today: The matrix norm and conjugate gradient.

Recall $\|x\|_A = \sqrt{x^T A x} = \|A^{1/2} x\|$, for $A \succcurlyeq 0$

$\tilde{x}$ is an ε-approx solution to $Ax=b$ if

$$\|\tilde{x} - x\|_A \leq \varepsilon \|x\|_A.$$

Richardson and Chebyshev produce these.

let p be a polynomial s.t. $\|p(A)A - I\| \leq \varepsilon$.

Then $\|p(A)b - x\|_A = \|A^{1/2} p(A) A x - A^{1/2} x\|$

$= \|(p(A)A - I) A^{1/2} x\|$ because $A^{1/2}$ commutes with $p(A)$

$\leq \|p(A)A - I\| \cdot \|A^{1/2} x\|$

$\leq \varepsilon \|x\|_A$

A general solver might not commute. For these $\| \ \|_A$ is right measure of approximation.

<u>Thm</u> For $A, Z \succcurlyeq 0$

$$\| ZAx - x \|_A \leq \varepsilon \|x\|_A, \quad \text{for all } x \qquad (*)$$

iff $$(1-\varepsilon) A^{-1} \preccurlyeq Z \preccurlyeq (1+\varepsilon) A^{-1}$$

<u>proof</u>

$$(*) \iff \forall x, \ \|A^{1/2}(ZA - I)x\| \leq \varepsilon \|A^{1/2} x\|$$

$$\iff \forall y \ (= A^{1/2} x) \ \|(A^{1/2} Z A^{1/2} - I) y\| \leq \varepsilon \|y\|$$

$$\iff \|A^{1/2} Z A^{1/2} - I\| \leq \varepsilon$$

$$\iff -\varepsilon I \preccurlyeq A^{1/2} Z A^{1/2} - I \preccurlyeq \varepsilon I \quad (\text{by symmetric})$$

$$\iff (1-\varepsilon) I \preccurlyeq A^{1/2} Z A^{1/2} \preccurlyeq (1+\varepsilon) I$$

$$\iff (1-\varepsilon) A^{-1} \preccurlyeq Z \preccurlyeq (1+\varepsilon) A^{-1}$$

Say want to approximate ψ_1 eigvec of λ_1 of A

Use power method on A^{-1}

This tells us could use power method on Z

If $\lambda_1 = \frac{1}{n}$, $\lambda_2 = \frac{2}{n}$, small gap, power method on $uI - A$ is slow. But, gap in A^{-1} is big.

CG Iter methods find solutions in span of $\{b, Ab, A^2b, \dots, A^t b\} = S_t(A, b)$

called $(t+1)$st Krylov subspace.

CG will solve

$$\arg\min_{x_t \in S_t} \|x_t - x\|_A$$

using t multiplies by A and $O(t)$ vector operations

First, $\|x_t - x\|_A^2 = x_t^T A x_t - 2x^T A x_t + x^T A x$

$$= x_t^T A x_t - 2b^T x_t + x^T A x$$

we do not know $x^T A x$.

But, minimizing $x_t^T A x_t - 2b^T x_t$ is the same.

Let $p_0, \dots, p_t$ be a basis of S_t, $x_t = \sum_{i=1}^{t} c_i p_i$

$$x_t^T A x_t - 2b^T x_t = \left(\sum_i c_i p_i\right)^T A \left(\sum_j c_j p_j\right) - 2b^T\left(\sum_i c_i p_i\right)$$

$$= \sum_i c_i^2 p_i^T A p_i - 2\sum_i c_i b^T p_i + \underbrace{\sum_{i \neq j} c_i c_j p_i^T A p_j}$$

CG constructs $p_0 \dots p_t$ s.t. this is zero

That is $p_i^T A p_j = 0$ for $i \neq j$

Thus, only need to minimize

$$\sum_i \left(c_i^2 p_i^T A p_i - 2 c_i b^T p_i \right)$$

Do by setting $c_i = \frac{b^T p_i}{p_i^T A p_i}$

That is, $x_t = \sum_{i=0}^{t} p_i \frac{b^T p_i}{p_i^T A p_i}$

so, add one vector to go from x_{t-1} to x_t

$p_0, \ldots, p_t$ is an A-orthogonal basis.

$p_0 = b$

$p_1 = A p_0$ ... but want $p_0^T A p_1 = 0$

so, set $p_1 = A p_0 - \frac{p_0^T A^2 p_0}{p_0^T A p_0} p_0$

Because need to compensate for $p_0^T A^2 p_0$

Note: is like running Gram-Schmidt $(A^{1/2} \cdot A^i b)_i$

General formula is

$$p_{t+1} = A p_t - \sum_{i=0}^{t} p_i \frac{p_i^T A^2 p_t}{p_i^T A p_i}$$

because $p_i^T A^2 p_t$ on left, $- p_i^T A p_i \frac{p_i^T A^2 p_t}{p_i^T A p_i}$ on right, and $p_i^T A p_j = 0$ for $i \neq j$

We can compute p_{t+1} quickly because

$$p_i^T A^2 p_t = 0 \quad \text{for } i < t-1$$

because $A p_i \in \text{Span}(p_0, \ldots, p_{i+1})$, which are A-orthogonal to p_t for $i+1 < t$

$$p_{t+1} = A p_t - p_t \frac{p_t A^2 p_t}{p_t A p_t} - p_{t-1} \frac{p_{t-1} A^2 p_t}{p_{t-1} A p_{t-1}}$$

So, we can compute p_{t+1} by a const ~~#~~ of multiples by A and vector operations.

Are ways to arrange these computations to make them very fast.

How good is CG? Never needs $> n$ iterations

In fact, never more than # distinct eigenvalues.

Let $\lambda_1 \dots \lambda_k$ be the eigvals of A, without repetition.

$$q(x) = \frac{\prod_{i=1}^{k} (\lambda_i - x)}{\prod_i \lambda_i} \qquad \begin{array}{l} q(\lambda_i) = 0 \\ q(0) = 1 \end{array}$$

So, should get exact solution after k steps.

For sparse matrices, with $O(n)$ nonzeros,
this takes time $O(n^2)$ and space $O(n^2)$.

In contrast, writing A^{-1} takes space $O(n^2)$, in general,
and longer to compute.

For Laplacian of Hypercube,
has only $\log n$ distinct eigenvalues.
So, $\log n$ iterations.