Welcome to Ma221! Lecture 25, Fall 24

Krylov Subspace Methods (KSMs) for $Ax=b$ and $Ax=\lambda x$

Intro: Arnoldi and Lanczos

Many KSMs for $Ax=b$, depending on structure of A

1) General A: GMRES
   generalized minimum residual
2) SPD A (eg Poisson): CG
   conjugate gradients

Later: Decision tree to choose KSM depending on A (Fig 6.8 in text)

Unlike Splitting Methods, KSMs only need "black box" for $A \cdot x$,
Eg don't need diag(A), as for Jacobi

1) can write algorithms that are very general, leave details of $A \cdot x$ to user
2) can solve problems when A not explicitly available, eg only $A \cdot x$ from doing a complicated simulation, or from automatic differentiation of a program for $f(x)$ to multiply $\nabla f \cdot x$

eg PyTorch

If you have access to A itself, can use it to avoid communication. Dominant cost of KSM is usually $A \cdot x$. Cost of doing k $A \cdot x$'s is $O(k)$ if A too large to store in fast memory. Can reorganize KSMs to take k steps and only read A once from slow memory, assuming A has a "good" sparsity structure

How to extract info about A from $A \cdot x$:

Given starting vector y_1 (say $y_1 = b$ if solving $Ax = b$)

Compute $y_2 = Ay_1$, $y_3 = Ay_2$, $\cdots$ $y_n = A^{n-1} y_1$

$$K = [y_1, y_2, \cdots, y_n]^{n \times n}$$

$$A \cdot K = [Ay_1, Ay_2, \cdots Ay_n]$$

$$= [y_2, y_3, \cdots y_n, A^n y_1]$$

If K nonsingular write $c = -K^{-1} A^n y_1$

$$A \cdot K = K \underbrace{[e_2, e_3, e_4 \cdots, e_n, -c]}_{C} = K \cdot C$$

$$C = K^{-1}AK = \begin{bmatrix} 0 & 0 & & & 0 & -c_1 \\ 1 & 0 & & & \vdots & -c_2 \\ 0 & 1 & \ddots & & \vdots & \vdots \\ 0 & 0 & & & \vdots & \vdots \\ \vdots & \vdots & & & \vdots & \vdots \\ & & \ddots & & 1 & -c_n \end{bmatrix}$$

upper Hessenberg

companion matrix

$$p(x) = \det(xI - C) = x^n + \sum_{i=1}^{n} c_i x^{i-1}$$

= characteristic polynomial of A

Is this useful for solving $Ax = b$ or $Ax = \lambda x$?

Disadvantages:

(1) K likely dense if A sparse, so solving $Kx = b$ harder than $Ax = b$

(2) K likely very ill conditioned because y_i converging to evec for λ_{max}, $\Rightarrow$ nearly parallel

How KSMs fix these problems

(1) Instead of K, compute orthogonal Q, where leading k columns of Q span same space as leading k columns of K, via QR of K

(2) only compute a few leading columns of Q

Def: Krylov subspace:

$= \text{span}\{y_1, y_2, y_3, \cdots, y_k\}$

$= \text{span}\{y_1, Ay_1, \cdots, A^{k-1}y_1\}$

$= \mathcal{K}_k(A, y_1)$

Relationship between K and Q: $K = QR$

Find "best" solution to $Ax = b$ or $Ax = \lambda x$ inside $\mathcal{K}_k(A, y_1)$

many definitions of "best" $\Rightarrow$ many algorithms (see decision tree)

How to compute Q column by column

$$K^{-1} A K = C = \begin{bmatrix} 1 & & 0 & -c_1 \\ & \ddots & & \vdots \\ 0 & & 1 & -c_n \end{bmatrix}$$

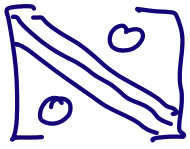
$$K = QR \Rightarrow R^{-1} Q^T A Q R = C$$

$$(*)\ Q^T A Q = RCR^{-1} = \square = H$$

upper Hessenberg

(Q6.11)

$A = A^T \Rightarrow H = H^T \Rightarrow H$ tridiagonal



$(*)\ Q^T A Q = H \Rightarrow AQ = QH$

equate j^{th} columns:

$$A q_j = \sum_{i=1}^{j+1} q_i H_{ij}$$

q_j orthog to $q_i \Rightarrow$ multiply both sides by q_m^T

$$q_m^T A q_j = \sum_{i=1}^{j+1} q_m^T q_i H_{ij} = H_{mj} \qquad 1 \le m \le j$$

$$H_{j+1,j} \cdot q_{j+1} = A q_j - \sum_{i=1}^{j} q_i H_{ij}$$

Arnoldi Algorithm for (partial) reduction to upper Hessenberg form

$q_1 = y_1 / \|y_1\|_2$

for $j = 1$ to k

  $z = A \cdot q_j$

  for $i = 1$ to j

    $H_{ij} = q_i^T z$ } MGS on z

    $z = z - H_{ij} \cdot q_i$ }

  end for

  $H_{j+1,j} = \|z\|_2$

  $q_{j+1} = z / H_{j+1,j}$

end for

q_j are called Arnoldi vectors

cost = k multiplications by A

+ $O(kn^2)$ flops for MGS

What have we learned about A after k steps?

$$Q = \left[\begin{array}{c|c} \overset{k}{Q_k} & \overset{n-k}{Q_U} \end{array}\right]$$

$Q_k = [q_1, q_2, \cdots, q_k]$ known

q_{k+1} known too

$$H = Q^T A Q = [Q_k \mid Q_U]^T A [Q_k \mid Q_U]$$

$$= \left[\begin{array}{c|c} Q_k^T A Q_k & Q_k^T A Q_U \\ \hline Q_U^T A Q_k & Q_U^T A Q_U \end{array}\right]$$

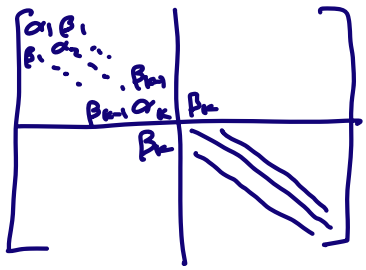
$$= \left[\begin{array}{c|c} H_k & H_{uk} \\ \hline H_{ku} & H_u \end{array}\right]$$

H upper Hessenberg $\Rightarrow$ H_k and H_u upper Hessenberg

$$H_{ku} = \left[\begin{array}{c} 0 \\ \hline \end{array}\right] \quad H(k+1,k)$$

H_k and H_{ku} known
H_u and H_{uk} unknown

if $A = A^T \Rightarrow H = T =$



Equate column j of both sides of $AQ = QT$

(***) $Aq_j = \beta_{j-1} q_{j-1} + \alpha_j \cdot q_j + \beta_j q_{j+1}$

Multiply both sides by $q_j^T \Rightarrow$

$q_j^T A q_j = \alpha_j$

Lanczos Algorithm for (partial) reduction of $A = A^T$ to tridiagonal form

$q_1 = y_1 / \|y_1\|_2$, $\beta_0 = 0$, $q_0 = 0$

for $j = 1$ to k

$z = Aq_j$

$\alpha_j = q_j^T z$

$z = z - \alpha_j \cdot q_j - \beta_{j-1} \cdot q_{j-1}$ ... MGS

$\beta_j = \|z\|_2$

$q_{j+1} = z / \beta_j$

How do we use Arnoldi or Lanczos to solve $Ax=b$ or $Ax=\lambda x$?

Consider $Ax=\lambda x$: use evals of H_k or T_k as approximate evals of A

To estimate error:

$$H_k y = \lambda y$$

$$H\begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} H_k & H_{uk} \\ H_{ku} & H_u \end{bmatrix}\begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} H_k y \\ H_{ku} y \end{bmatrix} = \begin{bmatrix} \lambda y \\ H_{k+1,k} \cdot y_k \cdot e_1 \end{bmatrix}$$

$$AQ = QH$$

$$A\left(Q\begin{bmatrix} y \\ 0 \end{bmatrix}\right) = QH\begin{bmatrix} y \\ 0 \end{bmatrix} = \lambda Q\begin{bmatrix} y \\ 0 \end{bmatrix} + \underbrace{q_{k+1} H_{k+1,k} \cdot y_k}_{\text{error}}$$

$$\|\text{error}\|_2 = |H_{k+1,k} \cdot y_k|$$

so if $|H_{k+1,k} \cdot y_k|$ small $\rightarrow$ approx. evec/eval pair $\left(Q\begin{bmatrix} y \\ 0 \end{bmatrix}, \lambda\right)$ has small residual

If $A = A^T$, Thm 5.5 $\Rightarrow$ $|H_{k+1,k} \cdot y_k|$ bounds distance from λ to nearest eval of A

(see fig 7.2 in text for illustration)

GMRES and CG for $Ax=b$

Goal: find "best" approximation x_k to $A^{-1}b$ in $\mathcal{K}_k$, $x_k \in \mathcal{K}_k$

(1) Choose x_k to minimize $\|x_k - x\|_2$, $x = A^{-1}b$
but we don't have enough info,
all we have is $H_k = Q_k^T A Q_k$, or T_k if $A = A^T$

(2) Choose x_k to minimize
$\|r_k\|_2$ $r_k = b - Ax_k$

2 Algorithms:
A general: use GMRES
$A = A^T$, use MINRES

(3) Choose x_k so $r_k \perp \mathcal{K}_k$, $r_k^T Q_k = 0$
"orthogonal residual property"
or a Galerkin condition

$A = A^T \Rightarrow$ use SYMMLQ

A general $\Rightarrow$ variant of GMRES

(4) A s.p.d $\Rightarrow$ define norm
$\|r\|_{A^{-1}} = (r^T A^{-1} r)^{1/2}$

"best" solution minimizes
$$\|r_k\|_{A^{-1}}^2 = r_k^T A^{-1} r_k$$
$$= (b - Ax_k)^T A^{-1} (b - Ax_k)$$
$$= (Ax - Ax_k)^T A^{-1} (Ax - Ax_k)$$
$$= (x - x_k)^T A A^{-1} A (x - x_k)$$

$= (x - x_k)^T A (x - x_k)$

$= \| x - x_k \|_A^2$

alg: Conjugate Gradients (CG)

Thm: A s.p.d. $\Rightarrow$ defs (3) and (4) of "best" are equivalent

CG cost: one step costs 1 $A \cdot x$
+ few dot products and saxpys,
only need to store 3 vectors

GMRES: fewest assumptions on A
CG: (nearly) most " "

see decision tree in text
or link on webpage to
Templates for the Solution of Linear Systems

GMRES: choose x_k to minimize
$\| r_k \|_2 = \| b - A x_k \|_2$

$x_k = Q_k y_k \in \mathcal{K}_k$

$n \left[\; \right]^k$

$\| r_k \|_2 = \| b - A Q_k y_k \|_2$

$= \| b - A [Q_k, Q_u] \begin{bmatrix} y_k \\ 0 \end{bmatrix} \|_2$

Q square, orthogonal

$= \| Q^T (\;) \|_2$

$$= \| Q^T b - \underbrace{Q^T A Q}_{H} \begin{bmatrix} y_k \\ 0 \end{bmatrix} \|_2$$

$b = y_1$:

$$= \| \|b\|_2 e_1 - H \begin{bmatrix} y_k \\ 0 \end{bmatrix} \|_2$$

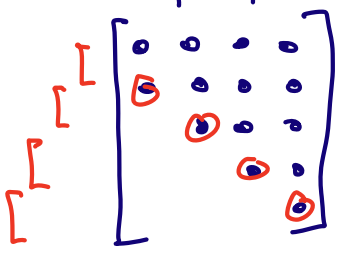
$$= \| \|b\|_2 e_1 - \left[\begin{array}{c|c} H_k & H_{ok} \\ \hline H_{ko} & H_o \end{array}\right] \begin{bmatrix} y_k \\ 0 \end{bmatrix} \|_2$$

$$= \| \|b\|_2 e_1 - \begin{matrix} \\ k+1 \end{matrix} \overset{k}{\begin{bmatrix} H_k \\ 0 \cdots 0 H_{k+1,k} \end{bmatrix}} y_k \|_2$$

$= k+1$ by k least squares problem

cheap to solve using Givens rotations

$k=4$



cost (QR) $= O(k^2)$

reuse work from step k to $k+1 \Rightarrow$

cost $= O(k)$ per iteration