Welcome to Ma221! (Apr 19)

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

Intro: Arnoldi and Lanczos

Many KSMs for $Ax=b$ depending on structure

1) General A: GMRES
   Generalized Minimum Residual
2) SPD A (eg Poisson): CG
   Conjugate Gradients

Later: Decision tree to choose (Fig 6.8 in text)

Unlike Splitting Methods: KSMs only need "blackbox" for $A \cdot x$.
Eg Don't need diag(A) like Jacobi

1) can write algorithms that are very general, leave all details of $A \cdot x$ to user
2) can solve problems when A not explicitly available, eg only Ax from a complicated simulation, or from differentiating a

program that computes $f(x)$ to get ∇f. used in optimization, via Automatic differentiation (Tensorflow)

If you do have access to A itself, can use it to avoid communication. Dominant cost of KSM is multiple $A \cdot x$. Cost of k Ax's, if A too large to fit in cache, communication cost is $O(k)$ to take k steps. Can reorganize KSMs to take k steps, only read A once, if A has "right" sparsity pattern

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

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

Compute $y_2 = Ay_1$, $y_3 = Ay_2$, $\ldots$ $y_{i+1} = Ay_i$

$y_n = A^{n-1} y_1$

$K = [y_1, y_2, \ldots, y_n]$ $n \times n$

$AK = [Ay_1, \ldots, Ay_n] = [y_2, y_3, \ldots y_n, A^n y_1]$

If K nonsingular, write $c = -K^{-1}A^n y$.

$$AK = K\underbrace{[e_2, e_3, e_4, \ldots, -c]}_{C} = K \cdot C$$

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

upper Hessenberg companion matrix

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

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 d_{max}, $\Rightarrow$ nearly parallel

Krylov subspace methods address these problems:

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

(2) only compute a few leading columns of Q, not all of them

Def: Krylov subspace

$$\text{span}\{y_1, y_2, \cdots, y_k\}$$
$$= \text{span}\{y_1, Ay_1, \cdots A^{k-1}y_1\}$$
$$= \mathcal{K}_k(A, y_1)$$

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" ⇒ many algorithms

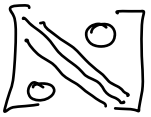
How to compute Q column by column:

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

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

(*) $Q^TAQ = RCR^{-1} =$  upper Hessenberg $= H$

(Q6.11)

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

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

equate j^{th} column of both sides

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

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

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

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

Arnoldi Algorithm for (partial) reduction to upper Hessenberg form

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

for $j = 1$ to k

  $z = Aq_j$

  for $i = 1$ to j

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

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

  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?

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

$Q_k = [q_1, \dots, q_k]$ known

q_{k+1} known

$$H = Q^T A Q = [Q_k | Q_u]^T A [Q_k | Q_u]$$

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

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

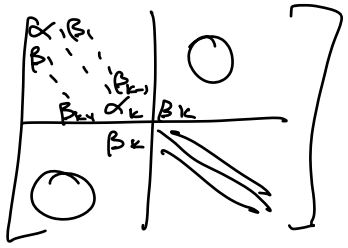
A upper Hessenberg $\Rightarrow$ H_k and H_u upper Hessenberg

$$H_{ku} = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \boxed{+} \end{array}\right] \quad H_{k+1,k}$$

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

$$AQ = QT$$

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

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

$z = A q_j$

$\alpha_j = q_j^T z$

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

$\beta_j = \|z\|$

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

end for

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

$$H_k y = \lambda y$$

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

$$AQ = QH$$

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

approx evec of A

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

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

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

Numerical Experiment: see Fig 7.2 in text