

# CS 598 EVS: Tensor Computations

## Matrix Computations Background

Edgar Solomonik

University of Illinois at Urbana-Champaign

# Conditioning

- ▶ **Absolute Condition Number:**

$$K(f, x) = f'(x)$$

- ▶ **(Relative) Condition Number:**

## Posedness and Conditioning

- ▶ **What is the condition number of an ill-posed problem?**

## Matrix Condition Number

- ▶ The matrix condition number  $\kappa(A)$  is the ratio between the max and min distance from the surface to the center of the unit ball (norm-1 vectors) transformed by  $A$ :

$$f(x) = Ax$$
$$\kappa(A) = \|A\| \cdot \underline{\|A^{-1}\|}$$

- ▶ The matrix condition number bounds the worst-case amplification of error in a matrix-vector product:

$$\|A^{-1}x\| \leq \|A^{-1}\| \cdot \|x\|$$
$$Ax = y \quad x = A^{-1}y$$
$$\|Ax\| \geq \frac{\|x\|}{\|A^{-1}\|}$$
$$\|x\| \leq \|A^{-1}\| \cdot \|y\|$$
$$\|y\| \geq \frac{\|x\|}{\|A^{-1}\|}$$

# Singular Value Decomposition

- ▶ The singular value decomposition (SVD)

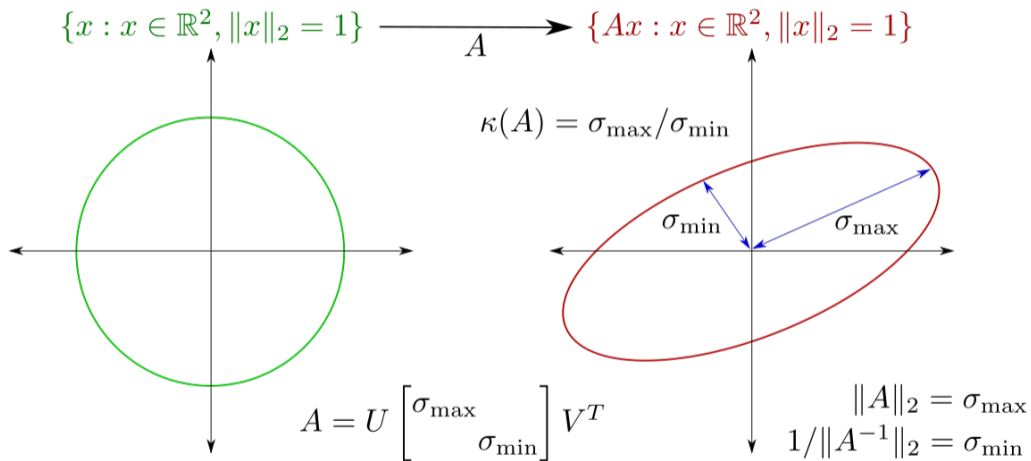
$$A = U \left[ \begin{array}{c} \sigma_{\max} \\ \vdots \\ \sigma_{\min} \end{array} \right] \left[ \begin{array}{c} V^T \\ \vdots \\ V_n^T \end{array} \right]$$

$Ax$

- ▶ Condition number in terms of singular values

$$\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}}$$

# Visualization of Matrix Conditioning



Ax



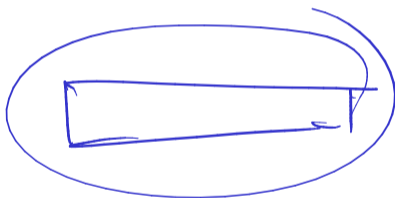
$$A = U \Sigma V^T$$

↓

$$F(A) = \|A\| \cdot \|A^{-1}\|$$



$$= \|A\| \cdot \|A^+ \|$$



f(A, x)



## Linear Least Squares

- ▶ Find  $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$  where  $\mathbf{A} \in \mathbb{R}^{m \times n}$ :
  
- ▶ Given the SVD  $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$  we have  $\mathbf{x}^* = \underbrace{\mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T}_{\mathbf{A}^\dagger} \mathbf{b}$ , where  $\mathbf{\Sigma}^\dagger$  contains the reciprocal of all nonzeros in  $\mathbf{\Sigma}$ , and more generally  $\dagger$  denotes pseudoinverse:



# Normal Equations

*Demo: Normal equations vs Pseudoinverse*

*Demo: Issues with the normal equations*

- ▶ *Normal equations* are given by solving  $\underline{A^T A}x = A^T b$ :

- ▶ However, solving the normal equations is a more ill-conditioned problem than the original least squares algorithm

$$\kappa(A^T A) = \kappa(A)^2$$

## Solving the Normal Equations

- ▶ If  $A$  is full-rank, then  $A^T A$  is symmetric positive definite (SPD):
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- ▶ Since  $A^T A$  is SPD we can use Cholesky factorization, to factorize it and solve linear systems:

## QR Factorization


- ▶ If  $A$  is full-rank there exists an orthogonal matrix  $Q$  and a unique upper-triangular matrix  $R$  with a positive diagonal such that  $A = QR$
  
- ▶ A reduced QR factorization (unique part of general QR) is defined so that  $Q \in \mathbb{R}^{m \times n}$  has orthonormal columns and  $R$  is square and upper-triangular
  
- ▶ We can solve the normal equations (and consequently the linear least squares problem) via reduced QR as follows

# Computing the QR Factorization

- ▶ The Cholesky-QR algorithm uses the normal equations to obtain the QR factorization

$$\begin{array}{l}
 A = \begin{matrix} n \\ \boxed{\phantom{A}} \\ m \end{matrix} \\
 A^T A = \underbrace{\phantom{LL^T}}_{mn^2} = \underbrace{\phantom{LL^T}}_{\substack{\uparrow \\ R^T \\ \substack{2 \\ 3 \\ n^3}}} LL^T \\
 A L^{-T} = Q \\
 A R^{-1} = \overline{Q} \\
 (QR) \quad mn^2
 \end{array}
 \Rightarrow
 \begin{array}{l}
 L Q^T = A^T \\
 \Delta \boxed{\phantom{A}} = \boxed{\phantom{A}} \\
 \underline{2mn^2 + \frac{2}{3}n^3}
 \end{array}$$

- ▶ Orthogonalization-based methods are most efficient and stable for QR factorization of dense matrices Projector

$$\begin{array}{l}
 (Q_1 \dots Q_k)^T A = R \\
 \underbrace{\phantom{(Q_1 \dots Q_k)^T A}}_{QR} = R
 \end{array}
 \left|
 \begin{array}{l}
 \text{Householder QR} \quad Z = \frac{I - uu^T}{2x} \\
 Q_i = \frac{I - 2u_i u_i^T}{2x} \\
 \uparrow \\
 \text{reflector}
 \end{array}
 \right.$$


## Eigenvalue Decomposition

- ▶ If a matrix  $A$  is diagonalizable, it has an *eigenvalue decomposition*

$$A = XDX^{-1}$$

\

$$AX = XD$$

- ▶  $A$  and  $B$  are *similar*, if there exist  $Z$  such that  $A = \underline{Z} \underline{B} Z^{-1}$

# Similarity of Matrices

A and B are similar iff  $\exists Z$   
 $A = Z B Z^{-1} \Rightarrow A \& B$  have same eigs

$$\underbrace{(Q_1 \dots Q_k)^T}_{A^H = A} A (Q_1 \dots Q_k)$$

$$a_{ij}^* = a_{ji}$$

matrix	similarity	reduced form
SPD	orthogonal	positive diagonal
real symmetric	orthogonal	diagonal $Q(n^2)$
	orthogonal	<u>tridiagonal</u> $O(n^3)$
<u>Hermitian</u>	unitary $Q^H Q = I$	diagonal real
$AA^H = A^H A$ normal	unitary	diagonal
real	orthogonal	<u>upper-Hessenberg</u> (triangular)
diagonalizable		
arbitrary		

# Rayleigh Quotient

- ▶ For any vector  $x$  that is close to an eigenvector, the *Rayleigh quotient* provides an estimate of the associated eigenvalue of  $A$ :

$$\rho(A, x) = \frac{x^T A x}{x^T x} \quad \text{if } x \text{ is an eigenvector}$$

$$x \approx \text{eigvec} \\ \downarrow \\ \rho(A, x) \approx \text{eigval}$$

$$\rho(A, x) = \frac{x^T \lambda x}{x^T x} = \lambda \frac{\min \| \lambda x - Ax \|_2}{\lambda} \quad \lambda = \frac{x^T A x}{x^T x}$$

$$Ax = \underbrace{A(\alpha_1 v_1 + \dots + \alpha_n v_n)}_{x}$$

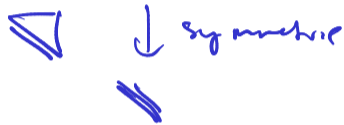
# Eigenvector & Eigenvalue computation

## Dense matrix

• start by reducing  $A$  to upper-Hessenberg

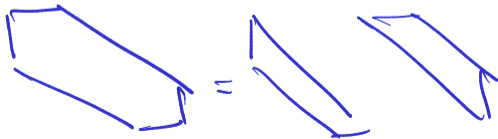
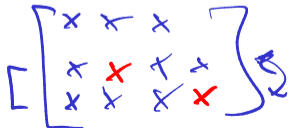
via Householder QR

in  $O(n^3)$



$f(x) = Ax$   
(implicit)  
Sparse

$f(x) = A^T Ax$   
matrix



• iterative methods often used, use  $A$  only for  $Ax$



## Power iteration

$$x \rightarrow Ax$$

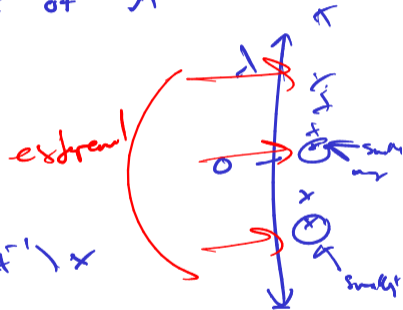
converge to largest eigenvector of  $A$

## Inverse iteration

$$x \rightarrow A^{-1}x$$

to smallest eigenvector of  $A$   
(in magnitude)

$$x \rightarrow (\sigma I - A^{-1})x$$



# Introduction to Krylov Subspace Methods

► *Krylov subspace methods* work with information contained in the  $n \times k$  matrix  $\text{poly}(A)$

$$\text{poly}(x) = 1 + x^2 + 2x^4$$

$$\text{poly}(A) = I + A^2 + 2A^4$$

$$K_k = [x_0 \quad Ax_0 \quad \dots \quad A^{k-1}x_0]$$

$$A^2 = AA$$

$$\min_{x \in \text{span}(K_k)} p(A, x)$$

$$\|Ax - b\| \quad \underbrace{A \dots A}_{k \text{ times}}$$

$$\lambda(A^2) = \lambda(A)^2$$

$$\text{span}(K_k)$$

$$\text{poly-deg-up-to-}(k)(A)x$$

►  $A$  is similar to companion matrix  $C = K_n^{-1}AK_n$ :

$$AK_n = \begin{bmatrix} Ax_0 & \dots & A^{n-1}x_0 \end{bmatrix}$$

$$K_n^{-1}AK_n =$$

$$\begin{bmatrix} A^n x_0 \\ x \\ x \\ x \\ x \\ x \\ 1 \end{bmatrix}$$



# Krylov Subspaces

- Given  $Q_k R_k = K_k$ , we obtain an orthonormal basis for the Krylov subspace,

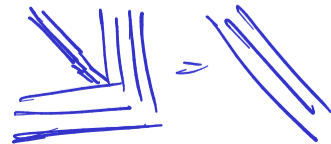
$$\mathcal{K}_k(\mathbf{A}, \mathbf{x}_0) = \text{span}(\mathbf{K}_k) = \{p(\mathbf{A})\mathbf{x}_0 : \deg(p) < k\},$$

where  $p$  is any polynomial of degree less than  $k$ .

- The Krylov subspace includes the  $k - 1$  approximate dominant eigenvectors generated by  $k - 1$  steps of power iteration:

$\mathbf{A}$   
Ritz vals  
eigval of  $H_k$

$$H_k = Q_k^T \mathbf{A} Q_k$$



$$\mathbf{A} = \mathbf{K}_n \mathbf{C} \mathbf{K}_n^{-1}$$

$$\mathbf{A} = \mathbf{Q}_n \mathbf{R}_n \mathbf{C} \mathbf{R}_n^{-1} \mathbf{Q}_n^T$$

$$H_k = Q_k^T A Q_k = Q_k^T Q_n R_n C R_n^{-1} Q_n^T Q_k$$

$\begin{bmatrix} \pm & 0 \end{bmatrix}$ 
 $\begin{bmatrix} \pm & 0 \end{bmatrix}$

## Krylov Subspace Methods

- ▶ The  $k \times k$  matrix  $\mathbf{H}_k = \mathbf{Q}_k^T \mathbf{A} \mathbf{Q}_k$  minimizes  $\|\mathbf{A} \mathbf{Q}_k - \mathbf{Q}_k \mathbf{H}_k\|_2$ :
  
  
  
  
  
  
  
  
  
  
- ▶  $\mathbf{H}_k$  is upper-Hessenberg, because the companion matrix  $\mathbf{C}_n$  is upper-Hessenberg:

## Rayleigh-Ritz Procedure

- ▶ The eigenvalues/eigenvectors of  $\mathbf{H}_k$  are the *Ritz values/vectors*:
- ▶ The Ritz vectors and values are the *ideal approximations* of the actual eigenvalues and eigenvectors based on only  $\mathbf{H}_k$  and  $\mathbf{Q}_k$ :







## Orthogonal Iteration

- ▶ For sparse matrices, QR factorization creates fill, so must revert to iterative methods
- ▶ Orthogonal iteration interleaves deflation and power iteration

## Randomized SVD

- ▶ Orthogonal iteration for SVD can also be viewed as a randomized algorithm

## Generalized Nyström Algorithm

- ▶ The generalized Nyström algorithm provides an efficient way of computing a sketched low-rank factorization

# Multidimensional Optimization

- ▶ Minimize  $f(\mathbf{x})$

- ▶ Quadratic optimization  $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T \mathbf{x}$

## Basic Multidimensional Optimization Methods

- ▶ Steepest descent: minimize  $f$  in the direction of the negative gradient:
  
  
  
  
  
  
  
  
  
  
- ▶ Given quadratic optimization problem  $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}^T \mathbf{x}$  where  $\mathbf{A}$  is symmetric positive definite, the error  $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$  satisfies

$$\|\mathbf{e}_{k+1}\|_{\mathbf{A}} =$$

- ▶ When sufficiently close to a local minima, general nonlinear optimization problems are described by such an SPD quadratic problem.
- ▶ Convergence rate depends on the conditioning of  $\mathbf{A}$ , since

## Gradient Methods with Extrapolation

- ▶ We can improve the constant in the linear rate of convergence of steepest descent by leveraging *extrapolation methods*, which consider two previous iterates (maintain *momentum* in the direction  $\mathbf{x}_k - \mathbf{x}_{k-1}$ ):
  
- ▶ The *heavy ball method*, which uses constant  $\alpha_k = \alpha$  and  $\beta_k = \beta$ , achieves better convergence than steepest descent:

# Conjugate Gradient Method

- ▶ The *conjugate gradient method* is capable of making the optimal (for a quadratic objective) choice of  $\alpha_k$  and  $\beta_k$  at each iteration of an extrapolation method:
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- ▶ *Parallel tangents* implementation of the method proceeds as follows

## Krylov Optimization

- ▶ Conjugate gradient (CG) finds the minimizer of  $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T \mathbf{x}$  (which satisfies optimality condition  $\mathbf{A}\mathbf{x} = \mathbf{b}$ ) within the Krylov subspace of  $\mathbf{A}$ :



## CG and Krylov Optimization

The solution at the  $k$ th step,  $\mathbf{y}_k = \frac{\|\mathbf{b}\|_2}{\|\mathbf{T}_k\|_2} \mathbf{T}_k^{-1} \mathbf{e}_1$  is obtained by CG from  $\mathbf{y}_{k+1}$  with a single matrix-vector product with  $\mathbf{A}$  and vector operations with  $O(n)$  cost

## Preconditioning

- ▶ Convergence of iterative methods for  $\mathbf{Ax} = \mathbf{b}$  depends on  $\kappa(\mathbf{A})$ , the goal of a preconditioner  $\mathbf{M}$  is to obtain  $\mathbf{x}$  by solving

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$$

with  $\kappa(\mathbf{M}^{-1}\mathbf{A}) < \kappa(\mathbf{A})$

- ▶ Common preconditioners select parts of  $\mathbf{A}$  or perform inexact factorization

## Conjugate Gradient Convergence Analysis

- ▶ In previous discussion, we assumed  $\mathbf{K}_n$  is invertible, which may not be the case if  $\mathbf{A}$  has  $m < n$  distinct eigenvalues, however, in exact arithmetic CG converges in  $m - 1$  iterations<sup>1</sup>

---

<sup>1</sup>This derivation follows *Applied Numerical Linear Algebra* by James Demmel, Section 6.6.4

## Conjugate Gradient Convergence Analysis (II)

- ▶ Using  $z = \rho_{k-1}(\mathbf{A})\mathbf{A}\mathbf{x}$ , we can simplify  $\phi(z) = (\mathbf{x} - z)^T \mathbf{A}(\mathbf{x} - z)$  as
  
  
  
  
  
  
  
  
  
  
- ▶ We can bound the objective based on the eigenvalues of  $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$  using the identity  $p(\mathbf{A}) = \mathbf{Q}p(\mathbf{\Lambda})\mathbf{Q}^T$ ,

## Conjugate Gradient Convergence Analysis (III)

- ▶ Using our bound on the square of the residual norm  $\phi(\mathbf{z})$ , we can see why CG converges after  $m - 1$  iterations if there are only  $m < n$  distinct eigenvalues
  
- ▶ To see that the residual goes to 0, we find a suitable polynomial in  $\mathcal{Q}_m$  (the set of polynomials  $q_m$  of degree  $m$  with  $q_m(0) = 1$ )

## Round-off Error in Conjugate Gradient

- ▶ CG provides strong convergence guarantees for SPD matrices in exact arithmetic
  
  
  
  
  
  
  
  
  
  
- ▶ Due to round-off CG may stagnate / have plateaus in convergence

## Graph and Matrix Duality

- ▶ graphs have have a natural correspondence with sparse matrices
  
- ▶ matrix-based representations of graphs can be used to devise algorithms

## Graph Partitioning from Eigenvectors

- ▶ The Laplacian matrix provides a model of interactions on a graph that is useful in many contexts
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- ▶ The second-smallest-eigenvalue eigenvector of the Laplacian (the Fiedler vector), gives a good partitioning of the graph



## Newton's Method

- ▶ Newton's method in  $n$  dimensions is given by finding minima of  $n$ -dimensional quadratic approximation using the gradient and Hessian of  $f$ :

## Nonlinear Least Squares

- ▶ An important special case of multidimensional optimization is *nonlinear least squares*, the problem of fitting a nonlinear function  $f_{\mathbf{x}}(t)$  so that  $f_{\mathbf{x}}(t_i) \approx y_i$ :
  
- ▶ We can cast nonlinear least squares as an optimization problem to minimize residual error and solve it by Newton's method:

