

# AMS526: Numerical Analysis I (Numerical Linear Algebra)

## Lecture 22: Computing SVD; Overview of Iterative Methods

Xiangmin Jiao

SUNY Stony Brook

December 2, 2008

# Outline

1 Wrapping up Eigenvalue Computation

2 Overview of Iterative Methods

# Computing the SVD

- Intuitive idea for computing SVD of  $\mathbf{A} \in \mathbb{R}^{m \times n}$ :
  - ▶ Form  $\mathbf{A}^* \mathbf{A}$  and compute its eigenvalue decomposition  $\mathbf{A}^* \mathbf{A} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^*$
  - ▶ Let  $\mathbf{\Sigma} = \sqrt{\mathbf{\Lambda}}$ , i.e.,  $\text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots, \sqrt{\lambda_n})$
  - ▶ Solve system  $\mathbf{U} \mathbf{\Sigma} = \mathbf{A} \mathbf{V}$  to obtain  $\mathbf{U}$
- This method can be very efficient if  $m \gg n$ .
- However, it is not very stable, especially for smaller singular values because of the squaring of the condition number
  - ▶ For SVD of  $\mathbf{A}$ ,  $|\tilde{\sigma}_k - \sigma_k| = O(\epsilon_{\text{machine}} \|\mathbf{A}\|)$ , where  $\tilde{\sigma}_k$  and  $\sigma_k$  denote the computed and exact  $k$ th singular value
  - ▶ If computed from eigenvalue decomposition of  $\mathbf{A}^* \mathbf{A}$ ,  
 $|\tilde{\sigma}_k - \sigma_k| = O(\epsilon_{\text{machine}} \|\mathbf{A}\|^2 / \sigma_k)$ , which is problematic if  $\sigma_k \ll \|\mathbf{A}\|$
- If one is interested in only relatively large singular values, then using eigenvalue decomposition is not a problem. For general situations, a more stable algorithm is desired.

## Computing the SVD

- Typical algorithm for computing SVD are similar to computation of eigenvalues
- Consider  $\mathbf{A} \in \mathbb{R}^{m \times m}$ , then hermitian matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{0} & \mathbf{A}^* \\ \mathbf{A} & \mathbf{0} \end{bmatrix}$$

has eigenvalue decomposition

$$\mathbf{H} \begin{bmatrix} \mathbf{V} & \mathbf{V} \\ \mathbf{U} & -\mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{V} & \mathbf{V} \\ \mathbf{U} & -\mathbf{U} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{\Sigma} \end{bmatrix},$$

where  $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}$  gives the SVD. This approach is stable.

- In practice, such a reduction is done implicitly without forming the large matrix
- Typically done in two or more stages:
  - ▶ First, reduce to bidiagonal form by applying different orthogonal transformations on left and right,
  - ▶ Second, reduce to diagonal form using a variant of QR algorithm or divide-and-conquer algorithm

## Generalized Eigenvalue Problem

- Generalized eigenvalue problem has the form

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{B}\mathbf{x},$$

where  $\mathbf{A}$  and  $\mathbf{B}$  are  $m \times m$  matrices

- For example, in structural vibration problems,  $\mathbf{A}$  represents the *stiffness matrix*,  $\mathbf{B}$  the *mass matrix*, and eigenvalues and eigenvectors determine natural frequencies and modes of vibration of structures
- If  $\mathbf{A}$  or  $\mathbf{B}$  is nonsingular, then it can be converted into standard eigenvalue problem

$$(\mathbf{B}^{-1}\mathbf{A})\mathbf{x} = \lambda\mathbf{x} \quad \text{or} \quad (\mathbf{A}^{-1}\mathbf{B})\mathbf{x} = (1/\lambda)\mathbf{x}$$

- If  $\mathbf{A}$  and  $\mathbf{B}$  are both symmetric, preceding transformation loses symmetry and in turn may lose orthogonality of generalized eigenvectors. If  $\mathbf{B}$  is positive definite, alternative transformation is

$$(\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T})\mathbf{y} = \lambda\mathbf{y}, \quad \text{where } \mathbf{B} = \mathbf{L}\mathbf{L}^T \text{ and } \mathbf{y} = \mathbf{L}^T\mathbf{x}$$

- If  $\mathbf{A}$  and  $\mathbf{B}$  are both singular or indefinite, then use *QZ algorithm* to reduce  $\mathbf{A}$  and  $\mathbf{B}$  into triangular matrices simultaneously by orthogonal

# Outline

1 Wrapping up Eigenvalue Computation

2 Overview of Iterative Methods

## Direct vs. Iterative Methods

- *Direct methods*, or *noniterative methods*, compute the exact solution after a finite number of steps (in exact arithmetic)
  - ▶ Example: Gaussian elimination, QR factorization
- *Iterative methods* produce a sequence of approximations  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots$  that hopefully converge to the true solution
  - ▶ Example: Jacobi, Conjugate Gradient (CG), GMRES, BiCG, etc.
- Caution: The boundary between direct and iterative methods is vague sometimes
- Why use iterative methods (instead of direct methods)?
  - ▶ may be faster than direct methods
  - ▶ produce useful intermediate results
  - ▶ handle sparse matrices more easily (needs only matrix-vector product)
  - ▶ often are easier to implement on parallel computers
- Question: When not to use iterative methods?

## Two Classes of iterative methods

- *Stationary methods* find a splitting  $\mathbf{A} = \mathbf{M} - \mathbf{K}$  and iterates

$$\mathbf{x}^{(k+1)} = \mathbf{M}^{-1}(\mathbf{K}\mathbf{x}^{(k)} + \mathbf{b})$$

- ▶ Examples: Jacobi (for linear systems, not the Jacobi iterations for eigenvalues), Gauss-Seidel, Successive Over-Relaxation (SOR) etc.
- ▶ Refer to textbook of AMS 505
- *Krylov subspace methods* find optimal solution in *Krylov subspace*  $\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \dots, \mathbf{A}^k\mathbf{b}\}$ 
  - ▶ Build subspace successively
  - ▶ Example: Conjugate Gradient (CG), Generalized Minimum Residual (GMRES), BiCG, etc.
  - ▶ We will focus on Krylov subspace methods

# Krylov Subspace Methods

- Given  $\mathbf{A}$  and  $\mathbf{b}$ , Krylov subspace

$$\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \dots, \mathbf{A}^{k-1}\mathbf{b}\}$$

	linear systems	eigenvalue problems
Hermitian	CG	Lanczos
Nonhermitian	GMRES, BiCG, etc.	Arnoldi



## Arnoldi Algorithm

- Start by picking a random  $\mathbf{q}_1$  and then determine  $\mathbf{q}_2$  and  $\tilde{\mathbf{H}}_1$
- The  $n$ th columns of  $\mathbf{A}\mathbf{Q}_n = \mathbf{Q}_{n+1}\tilde{\mathbf{H}}_n$  can be written as

$$\mathbf{A}\mathbf{q}_n = h_{1n}\mathbf{q}_1 + \cdots + h_{nn}\mathbf{q}_n + h_{n+1,n}\mathbf{q}_{n+1}$$

Algorithm: Arnoldi Iteration

given random nonzero  $\mathbf{b}$ , let  $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$

for  $n = 1$  to  $1, 2, 3, \dots$

$$\mathbf{v} = \mathbf{A}\mathbf{q}_n$$

for  $j = 1$  to  $n$

$$h_{jn} = \mathbf{q}_j^* \mathbf{v}$$

$$\mathbf{v} = \mathbf{v} - h_{jn}\mathbf{q}_j$$

$$h_{n+1,n} = \|\mathbf{v}\|$$

$$\mathbf{q}_{n+1} = \mathbf{v}/h_{n+1,n}$$

- Question: What if  $\mathbf{q}_1$  happens to be an eigenvector?

## QR Factorization of Krylov Matrix

- The vector  $\mathbf{q}_j$  from Arnoldi are orthonormal bases of successive Krylov subspaces

$$\mathcal{K}_n = \langle \mathbf{b}, \mathbf{A}\mathbf{b}, \dots, \mathbf{A}^{n-1}\mathbf{b} \rangle = \langle \mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n \rangle \subseteq \mathbb{C}^m$$

- $\mathbf{Q}_n$  is reduced QR factorization  $\mathbf{K}_n = \mathbf{Q}_n \mathbf{R}_n$  of Krylov matrix

$$\mathbf{K}_n = \left[ \begin{array}{c|c|c|c} \mathbf{b} & \mathbf{A}\mathbf{b} & \dots & \mathbf{A}^{n-1}\mathbf{b} \end{array} \right]$$

- The projection of  $\mathbf{A}$  onto this space gives  $n \times n$  Hessenberg matrix  $\mathbf{H}_n = \mathbf{Q}_n^* \mathbf{A} \mathbf{Q}_n = \tilde{\mathbf{H}}_{1:n,1:n}$
- Eigenvalues of  $\mathbf{H}_n$  (known as Ritz values) produce good approximations of those of  $\mathbf{A}$

## Lanczos Iteration for Symmetric Matrices

- For symmetric  $\mathbf{A}$ ,  $\tilde{\mathbf{H}}_n$  and  $\mathbf{H}_n$  are tridiagonal, denoted by  $\tilde{\mathbf{T}}_n$  and  $\mathbf{T}_n$ , respectively.  $\mathbf{A}\mathbf{Q}_n = \mathbf{Q}_{n+1}\tilde{\mathbf{H}}_n$  can be written as three-term recurrence

$$\mathbf{A}\mathbf{q}_n = \beta_{n-1}\mathbf{q}_{n-1} + \alpha_n\mathbf{q}_n + \beta_n\mathbf{q}_{n+1}$$

where  $\alpha_i$  are diagonal entries and  $\beta_i$  are sub-diagonal entries of  $\tilde{\mathbf{T}}_n$

Algorithm: Lanczos Iteration

$$\beta_0 = 0, \mathbf{q}_0 = \mathbf{0}$$

given random  $\mathbf{b}$ , let  $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$

for  $n = 1$  to  $1, 2, 3, \dots$

$$\mathbf{v} = \mathbf{A}\mathbf{q}_n$$

$$\alpha_n = \mathbf{q}_n^T \mathbf{v}$$

$$\mathbf{v} = \mathbf{v} - \beta_{n-1}\mathbf{q}_{n-1} - \alpha_n\mathbf{q}_n$$

$$\beta_n = \|\mathbf{v}\|$$

$$\mathbf{q}_{n+1} = \mathbf{v}/\beta_n$$

- Eigenvalues of  $\mathbf{T}_n$  (known as Ritz values) converge to eigenvalues of  $\mathbf{A}$ , and extreme eigenvalues converge faster