LARGE SPARSE EIGENVALUE PROBLEMS

- Projection methods
- The subspace iteration
- Krylov subspace methods: Arnoldi and Lanczos
- Golub-Kahan-Lanczos bidiagonalization

General Tools for Solving Large Eigen-Problems

- Projection techniques – Arnoldi, Lanczos, Subspace Iteration;
- Preconditionings: shift-and-invert, Polynomials, ...
- Deflation and restarting techniques
- Computational codes often combine these three ingredients

A few popular solution Methods

- Subspace Iteration [Now less popular – sometimes used for validation]
- Arnoldi’s method (or Lanczos) with polynomial acceleration
- Shift-and-invert and other preconditioners. [Use Arnoldi or Lanczos for \((A - \sigma I)^{-1}\).]
- Davidson’s method and variants, Jacobi-Davidson
- Specialized method: Automatic Multilevel Substructuring (AMLS).

Projection Methods for Eigenvalue Problems

Projection method onto \(K\) orthogonal to \(L\)

- Given: Two subspaces \(K\) and \(L\) of same dimension.
- Approximate eigenpairs \(\tilde{\lambda}, \tilde{u}\), obtained by solving:
  \[
  \text{Find: } \tilde{\lambda} \in \mathbb{C}, \tilde{u} \in K \text{ such that } (\tilde{\lambda} I - A)\tilde{u} \perp L
  \]
- Two types of methods:
  Orthogonal projection methods: Situation when \(L = K\).
  Oblique projection methods: When \(L \neq K\).
- First situation leads to Rayleigh-Ritz procedure
Rayleigh-Ritz projection

Given: a subspace $X$ known to contain good approximations to eigenvectors of $A$.

Question: How to extract ‘best’ approximations to eigenvalues/eigenvectors from this subspace?

Answer: Orthogonal projection method
- Let $Q = [q_1, \ldots, q_m] = \text{orthonormal basis of } X$
- Orthogonal projection method onto $X$ yields:
  \[ Q^H (A - \tilde{\lambda}I) \tilde{u} = 0 \]
- Known as Rayleigh Ritz process

Subspace Iteration

Original idea: projection technique onto a subspace of the form $Y = A^k X$
Practically: $A^k$ replaced by suitable polynomial
Advantages: • Easy to implement (in symmetric case);	• Easy to analyze;
Disadvantage: Slow.
- Often used with polynomial acceleration: $A^k X$ replaced by $C_k(A)X$. Typically $C_k = \text{Chebyshev polynomial}$.

Algorithm: Subspace Iteration with Projection
1. Start: Choose an initial system of vectors $X = [x_0, \ldots, x_m]$ and an initial polynomial $C_k$.
2. Iterate: Until convergence do:
   (a) Compute $\tilde{Z} = C_k(A)X$. [Simplest case: $\tilde{Z} = AX$.]
   (b) Orthonormalize $\tilde{Z}$: $[Z, R_Z] = qr(\tilde{Z}, 0)$
   (c) Compute $B = Z^H AZ$
   (d) Compute the Schur factorization $B = Y R_B Y^H$ of $B$
   (e) Compute $X := ZY$.
   (f) Test for convergence. If satisfied stop. Else select a new polynomial $C'_k$ and continue.
THEOREM: Let $S_0 = \text{span}\{x_1, x_2, \ldots, x_m\}$ and assume that $S_0$ is such that the vectors $\{P^i\} = \{P^i x_i\} : i = 1, \ldots, m$ are linearly independent, where $P$ is the spectral projector associated with $\lambda_1, \ldots, \lambda_m$. Let $P_k$ the orthogonal projector onto the subspace $S_k = \text{span}\{X_k\}$. Then for each eigenvector $u_i$ of $A$, $i = 1, \ldots, m$, there exists a unique vector $s_i$ in the subspace $S_0$ such that $Ps_i = u_i$. Moreover, the following inequality is satisfied

$$\| (I - P_k)u_i \|_2 \leq \| u_i - s_i \|_2 \left( \frac{\lambda_{m+1}}{\lambda_i} + \epsilon_k \right)^k,$$

where $\epsilon_k$ tends to zero as $k$ tends to infinity.

**Krylov subspace methods**

*Principle:* Projection methods on Krylov subspaces:

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}$$

- The most important class of projection methods [for linear systems and for eigenvalue problems]
- Variants depend on the subspace $L$

- Let $\mu = \text{deg. of minimal polynom. of } v_1$. Then:
  - $K_m = \{p(A)v_1 | p = \text{polynomial of degree } \leq m - 1\}$
  - $K_m = K_\mu$ for all $m \geq \mu$. Moreover, $K_\mu$ is invariant under $A$.
  - $\dim(K_m) = m$ iff $\mu \geq m$.

**Arnoldi’s algorithm**

- Goal: to compute an orthogonal basis of $K_m$.
- Input: Initial vector $v_1$, with $\| v_1 \|_2 = 1$ and $m$.

**Algorithm:**

1. **Arnoldi’s procedure**

   For $j = 1, \ldots, m$ do
   
   Compute $w := Av_j$
   
   For $i = 1, \ldots, j$, do
   
   \[ h_{i,j} := (w, v_i) \]
   
   \[ w := w - h_{i,j}v_i \]
   
   \[ h_{j+1,j} := \|w\|_2; \]
   
   \[ v_{j+1} := w/h_{j+1,j} \]
   
   End

- Based on Gram-Schmidt procedure
Result of Arnoldi’s algorithm

Let: \( \mathbf{H}_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \), \( \mathbf{H}_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \)

Results:
1. \( \mathbf{V}_m = [v_1, v_2, \ldots, v_m] \) orthonormal basis of \( \mathbf{K}_m \).
2. \( \mathbf{A} \mathbf{V}_m = \mathbf{V}_{m+1} \mathbf{H}_m = \mathbf{V}_m \mathbf{H}_m + h_{m+1,m} v_{m+1} e_m^T \)
3. \( \mathbf{V}_m^T \mathbf{A} \mathbf{V}_m = \mathbf{H}_m \equiv \mathbf{H}_m - \text{last row.} \)

Application to eigenvalue problems

- Write approximate eigenvector as \( \tilde{\mathbf{u}} = \mathbf{V}_m \mathbf{y} \)
- Galerkin condition:
  \( (\mathbf{A} - \tilde{\lambda} \mathbf{I}) \mathbf{V}_m \mathbf{y} \perp \mathbf{K}_m \rightarrow \mathbf{V}_m^H (\mathbf{A} - \tilde{\lambda} \mathbf{I}) \mathbf{V}_m \mathbf{y} = 0 \)
- Approximate eigenvalues are eigenvalues of \( \mathbf{H}_m \)
  \( \mathbf{H}_m \mathbf{y}_j = \tilde{\lambda}_j \mathbf{y}_j \)
- Associated approximate eigenvectors are \( \tilde{\mathbf{u}}_j = \mathbf{V}_m \mathbf{y}_j \)
- Typically a few of the outermost eigenvalues will converge first.

Hermitian case: The Lanczos Algorithm

- The Hessenberg matrix becomes tridiagonal:
  \( \mathbf{A} = \mathbf{A}^H \) and \( \mathbf{V}_m^H \mathbf{A} \mathbf{V}_m = \mathbf{H}_m \rightarrow \mathbf{H}_m = \mathbf{H}_m^H \)
- Denote \( \mathbf{H}_m \) by \( \mathbf{T}_m \) and \( \tilde{\mathbf{H}}_m \) by \( \tilde{\mathbf{T}}_m \). We can write
  \( \mathbf{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 \\ \beta_2 & \alpha_2 & \beta_3 \\ \beta_3 & \alpha_3 & \beta_4 \\ \vdots & \ddots & \ddots \\ \beta_m & \alpha_m \end{pmatrix} \)
- Relation \( \mathbf{A} \mathbf{V}_m = \mathbf{V}_{m+1} \mathbf{T}_m \)

Consequence: three term recurrence
\( \beta_{j+1} \mathbf{v}_{j+1} = \mathbf{A} \mathbf{v}_j - \alpha_j \mathbf{v}_j - \beta_j \mathbf{v}_{j-1} \)

ALGORITHM : 2. Lanczos
1. Choose an initial \( \mathbf{v}_1 \) with \( \| \mathbf{v}_1 \|_2 = 1 \);
   Set \( \beta_1 \equiv 0, v_0 \equiv 0 \)
2. For \( j = 1, 2, \ldots, m \) Do:
3. \( \mathbf{w}_j := \mathbf{A} \mathbf{v}_j - \beta_j \mathbf{v}_{j-1} \)
4. \( \alpha_j := (\mathbf{w}_j, \mathbf{v}_j) \)
5. \( \mathbf{w}_j := \mathbf{w}_j - \alpha_j \mathbf{v}_j \)
6. \( \beta_{j+1} := \| \mathbf{w}_j \|_2. \) If \( \beta_{j+1} = 0 \) then Stop
7. \( \mathbf{v}_{j+1} := \mathbf{w}_j / \beta_{j+1} \)
8. EndDo

Hermitian matrix + Arnoldi → Hermitian Lanczos
In theory $v_i$’s defined by 3-term recurrence are orthogonal.

However: in practice severe loss of orthogonality;

Observation [Paige, 1981]: Loss of orthogonality starts suddenly, when the first eigenpair has converged. It is a sign of loss of linear independence of the computed eigenvectors. When orthogonality is lost, then several the copies of the same eigenvalue start appearing.

Reorthogonalization

- Full reorthogonalization – reorthogonalize $v_{j+1}$ against all previous $v_i$’s every time.
- Partial reorthogonalization – reorthogonalize $v_{j+1}$ against all previous $v_i$’s only when needed [Parlett & Simon]
- Selective reorthogonalization – reorthogonalize $v_{j+1}$ against computed eigenvectors [Parlett & Scott]
- No reorthogonalization – Do not reorthogonalize - but take measures to deal with ‘spurious’ eigenvalues. [Cullum & Willoughby]

Lanczos Bidiagonalization

We now deal with rectangular matrices. Let $A \in \mathbb{R}^{m \times n}$.

Algorithm: 3. Golub-Kahan-Lanczos

1. Choose an initial $v_1$ with $\|v_1\|_2 = 1$; Set $p \equiv v_1$, $\beta_0 \equiv 1$, $u_0 \equiv 0$
2. For $k = 1, \ldots, p$ Do:
3. $r := Av_k - \beta_{k-1}u_{k-1}$
4. $\alpha_k = \|r\|_2$; $u_k = r/\alpha_k$;
5. $p = A^T u_k - \alpha_k v_k$
6. $\beta_k = \|p\|_2$; $v_{k+1} := p/\beta_k$
7. EndDo

Let:

$V_p = [v_1, v_2, \ldots, v_p] \in \mathbb{R}^{n \times p}$
$U_p = [u_1, u_2, \ldots, u_p] \in \mathbb{R}^{m \times p}$

Result:

$B_p = \begin{bmatrix} \alpha_1 & \beta_2 & \alpha_2 & \beta_3 & \cdots & \cdots \\ \alpha_2 & \beta_3 & \alpha_3 & \beta_4 & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \ddots & \alpha_p \\ \cdots & \cdots & \cdots & \cdots & \cdots & \beta_{p+1} \end{bmatrix}$

$\hat{B}_p = B_p(:,1:p)$

$V_p = [v_1, v_2, \ldots, v_p] \in \mathbb{R}^{n \times p}$

$V_{p+1}^T V_{p+1} = I$
$U_{p+1}^T U_{p+1} = I$
$A V_p = U_p \hat{B}_p$
$A^T U_p = V_{p+1} \hat{B}_p^T$
Observe that:

\[ A^T(AV_p) = A^T(U_p \hat{B}_p) \]
\[ = V_{p+1}B_p^T \hat{B}_p \]

\[ B_p^T \hat{B}_p \] is a (symmetric) tridiagonal matrix of size \((p + 1) \times p\)

Call this matrix \( T_k \). Then:

\[ (A^T A)V_p = V_{p+1} \overline{T}_p \]

Standard Lanczos relation!

Algorithm is equivalent to standard Lanczos applied to \( A^T A \).

Similar result for the \( u_i \)'s [involves \( AA^T \)]

Work out the details: What are the entries of \( \overline{T}_p \) relative to those of \( B_p \)?