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] = $ orthonormal basis of $X$
- Orthogonal projection method onto $X$ yields:
  \[ Q^H(A - \lambda I) \tilde{u} = 0 \rightarrow \]
  \[ Q^H A Q y = \tilde{\lambda} y \text{ where } \tilde{u} = Q y \]

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 = $ 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 $B = Z^H A Z$
   (d) Compute the Schur factorization $B = Y R_B Y^H$ of $B$
   (e) Compute $X := Z Y$.
   (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 \( \{P_i x_i\}_{i=1}^{m} \) 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 \( A \).
  - \( \text{dim}(K_m) = m \) 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:** Arnoldi’s procedure

For \( j = 1, \ldots, m \) do

Compute \( w := Av_j \)

For \( i = 1, \ldots, j \), do

\[
\begin{align*}
  h_{i,j} &:= (w, v_i) \\
  w &:= w - h_{i,j}v_i \\
  h_{j+1,j} &:= \|w\|_2 \\
  v_{j+1} &:= w/h_{j+1,j}
\end{align*}
\]

End

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

Let: \( H_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x \\ x & x & x \\ x & x \end{pmatrix} \), \( H_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x \\ x & x & x \\ x & x \end{pmatrix} \)

**Results:**

1. \( V_m = [v_1, v_2, ..., v_m] \) orthonormal basis of \( K_m \).
2. \( AV_m = V_{m+1}H_m = V_mH_m + h_{m+1,m}v_{m+1}e_T^m \)
3. \( V_m^T AV_m = H_m \equiv \overline{H}_m \) last row.

**Application to eigenvalue problems**

1. Write approximate eigenvector as \( \tilde{u} = V_m y \)
2. Galerkin condition:
   \((A - \tilde{\lambda} I)V_m y \perp K_m \rightarrow V_m^H (A - \tilde{\lambda} I)V_m y = 0 \)
3. Approximate eigenvalues are eigenvalues of \( H_m \)
   \( H_m y_j = \tilde{\lambda}_j y_j \)
4. Associated approximate eigenvectors are \( \tilde{u}_j = V_m y_j \)
5. Typically a few of the outermost eigenvalues will converge first.

**Hermitian case: The Lanczos Algorithm**

- The Hessenberg matrix becomes tridiagonal:
  \( A = A^H \) and \( V_m^H AV_m = H_m \rightarrow H_m = H_m^H \)
- Denote \( H_m \) by \( T_m \) and \( \overline{H}_m \) by \( \overline{T}_m \). We can write
  \( T_m = \begin{pmatrix} \alpha_1 & \beta_2 \\ \beta_2 & \alpha_2 & \beta_3 \\ & \beta_3 & \alpha_3 & \beta_4 \\ & & \ddots & \ddots \end{pmatrix} \)
- Relation \( AV_m = V_{m+1}T_m \)

**Consequence: three term recurrence**

\( \beta_{j+1} v_{j+1} = Av_j - \alpha_j v_j - \beta_{j} v_{j-1} \)

**ALGORITHM 2. Lanczos**

1. Choose an initial \( v_1 \) with \( \|v_{-1}\|_2 = 1 \);
   Set \( \beta_1 \equiv 0, v_0 \equiv 0 \)
2. For \( j = 1, 2, ..., m \) Do:
3. \( w_j := Av_j - \beta_j v_{j-1} \)
4. \( \alpha_j := (w_j, v_j) \)
5. \( w_j := w_j - \alpha_j v_j \)
6. \( \beta_{j+1} := \|w_j\|_2. If \beta_{j+1} = 0 then Stop \)
7. \( v_{j+1} := w_j / \beta_{j+1} \)
8. EndDo

Hermitian matrix + Arnoldi \rightarrow 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 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 $\beta_0 \equiv 0$, $u_0 \equiv 0$
2. For $k = 1, \ldots, p$ Do:
   3. $\hat{u} := Av_k - \beta_{k-1} u_{k-1}$
   4. $\alpha_k = \|\hat{u}\|_2$; $u_k = \hat{u}/\alpha_k$
   5. $\hat{v} = A^T u_k - \alpha_k v_k$
   6. $\beta_k = \|\hat{v}\|_2$; $v_{k+1} := \hat{v}/\beta_k$
7. EndDo

Let:

- $V_{p+1} = [v_1, v_2, \cdots, v_{p+1}] \in \mathbb{R}^{n \times (p+1)}$
- $U_p = [u_1, u_2, \cdots, u_p] \in \mathbb{R}^{m \times p}$

**Result:**

- $V_p^T V_p + 1 = I$
- $U_p^T U_p = I$
- $AV_p = U_p B_p$
- $A^T U_p = V_{p+1} B_p^T$
Observe that: 
\[ A^T(AV_p) = A^T(U_p \hat{B}_p) = V_{p+1}B_p \]

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

Call this matrix \(T_k\). Then:

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

Standard Lanczos relation!

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

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 \(B_p\)?