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;**
- **Preconditioninings: 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:

Find: $\tilde{\lambda} \in \mathbb{C}, \tilde{u} \in K$ **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:

$$Q^H(A - \tilde{\lambda}I)\tilde{u} = 0 \;\rightarrow$$

- $Q^H AQy = \tilde{\lambda}y$ where $\tilde{u} = Qy$
- Known as Rayleigh Ritz process

Procedure:

1. Obtain an orthonormal basis of X
2. Compute $C = Q^H A Q$ (an $m \times m$ matrix)
3. Obtain Schur factorization of C, $C = Y R Y^H$
4. Compute $\tilde{U} = QY$

Property: if X is (exactly) invariant, then procedure will yield exact eigenvalues and eigenvectors.

Proof: Since X is invariant, $(A - \tilde{\lambda} I)u = Qz$ for a certain z. $Q^H Q z = 0$ implies $z = 0$ and therefore $(A - \tilde{\lambda} I)u = 0$.

➤ Can use this procedure in conjunction with the subspace obtained from subspace iteration algorithm

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 $\hat{Z} = C_k(A)X$.

   (b) Orthonormalize $\hat{Z}$: $[Z, R_Z] = qr(\hat{Z}, 0)$

   (c) Compute $B = Z^H A Z$

   (d) Compute the Schur factorization $B = Y R_B Y^H$ of B

   (e) Compute $X := ZY$.

   (f) Test for convergence. If satisfied stop. Else select a new polynomial $C'_{k'}$ and continue.

THEOREM: Let $S_0 = span\{x_1, x_2, \ldots, x_m\}$ and assume that S_0 is such that the vectors $\{Px_i\}_{i=1,\ldots,m}$ are linearly independent where P is the spectral projector associated with $\lambda_1, \ldots, \lambda_m$. Let $\mathcal{P}_k$ the orthogonal projector onto the subspace $S_k = span\{X_k\}$. 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

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

where ϵ_k tends to zero as k tends to infinity.

KRYLOV SUBSPACE METHODS

Krylov subspace methods

Principle: Projection methods on Krylov subspaces:

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

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

➤ Let μ = deg. of minimal polynom. of v_1. Then:

- $K_m = \{p(A)v_1 | p = \text{polynomial of degree} \le m - 1\}$
- $K_m = K_\mu$ for all $m \ge \mu$. Moreover, K_μ is invariant under A.
- $dim(K_m) = m$ iff $\mu \ge m$.

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

- **Based on Gram-Schmidt procedure**

Result of Arnoldi's algorithm

Let: $\overline{H}_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \\ & & & & x \end{pmatrix}$, $H_m = \begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{pmatrix}$

Results:

1. $V_m = [v_1, v_2, ..., v_m]$ **orthonormal basis of** K_m.
2. $AV_m = V_{m+1}\overline{H}_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T$
3. $V_m^T A V_m = H_m \equiv \overline{H}_m -$ **last row.**

Application to eigenvalue problems

- Write approximate eigenvector as $\tilde{u} = V_m y$
- Galerkin condition:

$$(A - \tilde{\lambda} I) V_m y \ \perp \ \mathcal{K}_m \quad \rightarrow \quad V_m^H (A - \tilde{\lambda} I) V_m y = 0$$

- Approximate eigenvalues are eigenvalues of H_m

$$H_m y_j = \tilde{\lambda}_j y_j$$

- Associated approximate eigenvectors are

$$\tilde{u}_j = V_m y_j$$

- Typically a few of the outermost eigenvalues will converge first.

Hermitian case: The Lanczos Algorithm

- **The Hessenberg matrix becomes tridiagonal :**

$$A = A^H \quad \text{and} \quad V_m^H A V_m = H_m \quad \rightarrow H_m = H_m^H$$

- **Denote H_m by T_m and $\bar{H}_m$ by $\bar{T}_m$. We can write**

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & & \beta_m & \alpha_m \end{pmatrix}$$

- **Consequence: three term recurrence**

$$\beta_{j+1} v_{j+1} = A v_j - \alpha_j v_j - \beta_j v_{j-1}$$

- **Relation $AV_m = V_{m+1} \overline{T_m}$**

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 $j = 1, 2, \ldots, m$ Do:
3. $\quad w_j := Av_j - \beta_j v_{j-1}$
4. $\quad \alpha_j := (w_j, v_j)$
5. $\quad w_j := w_j - \alpha_j v_j$
6. $\quad \beta_{j+1} := \|w_j\|_2$. If $\beta_{j+1} = 0$ then Stop
7. $\quad v_{j+1} := w_j / \beta_{j+1}$
8. EndDo

$\boxed{\text{Hermitian matrix + Arnoldi} \rightarrow \text{Hermitian Lanczos}}$

➤ 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 indedependence 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. $\quad r := Av_k - \beta_{k-1}u_{k-1}$
4. $\quad \alpha_k = \|r\|_2$; $\qquad u_k = r/\alpha_k$
5. $\quad p = A^T u_k - \alpha_k v_k$
6. $\quad \beta_k = \|p\|_2$; $\qquad v_{k+1} := p/\beta_k$
7. EndDo

Let:

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

Let:

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

- $\hat{B}_p = B_p(:, 1:p)$
- $V_p = [v_1, v_2, \cdots, v_p] \in \mathbb{R}^{n \times p}$

Result:

- $V_{p+1}^T V_{p+1} = I$
- $U_p^T U_p = I$
- $AV_p = U_p \hat{B}_p$
- $A^T U_p = V_{p+1} B_p^T$

Observe that

$$\begin{aligned} A^T(AV_p) &= A^T(U_p\hat{B}_p) \\ &= V_{p+1}B_p^T\hat{B}_p \end{aligned}$$

➤ $B_p^T\hat{B}_p$ is a (symmetric) tridiagonal matrix of size $(p+1) \times p$ – Call it $\overline{T_k}$. Then

$$\boldsymbol{(A^TA)V_p = V_{p+1}\overline{T_p}}$$

➤ Standard Lanczos relation !

➤ Therefore the algorithm is equivalent to the standard Lanczos algorithm applied to A^TA.

➤ Similar result for the u_i's [involves AA^T]

✍ Work out the details: What are the entries of $\bar{T}_p$ relative to those of B_p?

APPLICATION: GRAPH PARTITIONING

Graph Laplaceans - Definition

➤ "Laplace-type" matrices associated with general undirected graphs – useful in many applications

➤ Given a graph $G = (V, E)$ define

- A matrix W of weights w_{ij} for each edge
- Assume $w_{ij} \geq 0,$, $w_{ii} = 0$, and $w_{ij} = w_{ji} \; \forall (i, j)$
- The diagonal matrix $D = diag(d_i)$ with $d_i = \sum_{j \neq i} w_{ij}$

➤ Corresponding graph Laplacean of G is:

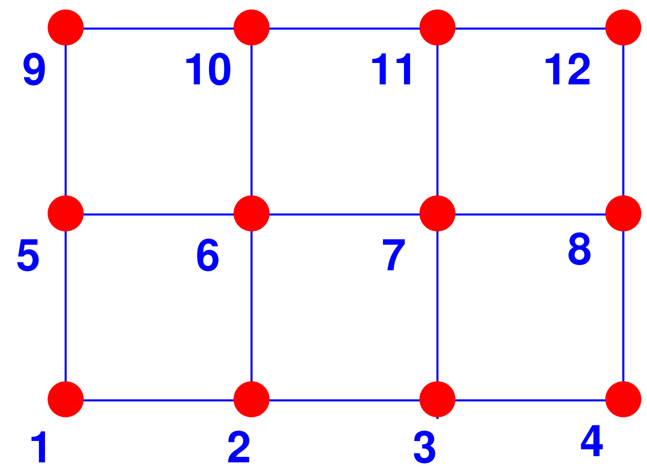
$$L = D - W$$

➤ Gershgorin's theorem → L is positive semidefinite

➤ **Simplest case:**

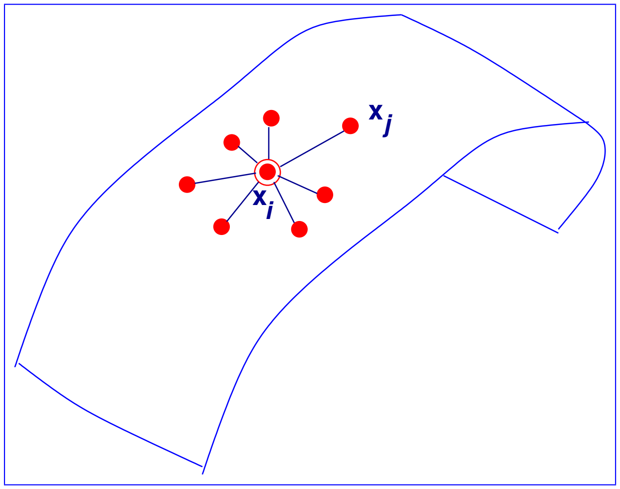
$$w_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \& i \neq j \\ 0 \quad \text{else} \end{cases} \qquad D = \mathrm{diag}\left[d_i = \sum_{j \neq i} w_{ij} \right]$$

✍ **Define the graph Laplacean for the graph associated with the simple mesh shown next. [use the simple weights of 0 or 1]**



✍ **What is the difference with the discretization of the Laplace operator in 2-D for case when mesh is the same as this graph?**

A few properties of graph Laplaceans



Strong relation between $x^T L x$ and local distances between entries of x

➤ **Let L = any matrix s.t. $L = D - W$, with $D = diag(d_i)$ and**

$$w_{ij} \geq 0, \qquad d_i = \sum_{j \neq i} w_{ij}$$

Property 1: **for any $x \in \mathbb{R}^n$:**

$$x^\top L x = \frac{1}{2} \sum_{i,j} w_{ij} |x_i - x_j|^2$$

Property 2: **(generalization) for any $Y \in \mathbb{R}^{d \times n}$:**

$$\mathrm{Tr}\,[YLY^\top] = \frac{1}{2}\sum_{i,j} w_{ij}\|y_i - y_j\|^2$$

Property 3: For the particular $L = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$

$$XLX^\top = \bar{X}\bar{X}^\top == n \times \text{Covariance matrix}$$

Property 4: L is singular and admits the null vector $e =$`ones(n,1)`

Property 5: (Graph partitioning) Consider situation when $w_{ij} \in \{0, 1\}$. If x is a vector of signs (± 1) then

$$x^\top L x = 4 \times \text{('number of edge cuts')}$$

edge-cut = pair (i, j) with $x_i \neq x_j$

- Would like to minimize (Lx, x) subject to $x \in \{-1, 1\}^n$ and $e^T x = 0$ [balanced sets]

- Wll solve a relaxed form of this problem

- Consider any symmetric (real) matrix A with eigenvalues $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ and eigenvectors $u_1, \cdots, u_n$

- Recall that:
(Min reached for $x = u_1$)

$$\min_{x \in \mathbb{R}^n} \frac{(Ax, x)}{(x, x)} = \lambda_1$$

- In addition:
(Min reached for $x = u_2$)

$$\min_{x \perp u_1} \frac{(Ax, x)}{(x, x)} = \lambda_2$$

- For a graph Laplacean $u_1 = e =$ vector of all ones and

- ...vector u_2 is called the Fiedler vector. It solves a relaxed form of the problem -

$$\min_{x \in \{-1,1\}^n;\ e^T x = 0} \frac{(Lx, x)}{(x, x)} \quad \rightarrow \quad \min_{x \in \mathbb{R}^n;\ e^T x = 0} \frac{(Lx, x)}{(x, x)}$$

- Define $v = u_2$ then $lab = sign(v - med(v))$

Spectral Graph Partitioning

Idea:

➤ Partition graph in two using fiedler vectors

➤ Cut largest in two ..

➤ Repeat until number of desired partitions is reached

➤ Use the Lanczos algorithm to compute the Fiedler vector at each step