Many applications require the computation of a few eigenvalues + associated eigenvectors of a matrix $A$

- Structural Engineering – (Goal: frequency response)
- Electronic structure calculations [Schrödinger equation..] – Quantum chemistry
- Stability analysis [e.g., electrical networks, mechanical system,..]
- ...

Introduction & Background
What is really needed is an invariant subspace of some large matrix $A$, i.e., a subspace $\mathcal{X}$ such that:

$$A\mathcal{X} \subseteq \mathcal{X} \quad \text{or} \quad AY = YC$$

$Y = \text{basis of subspace } \mathcal{X} \text{ of dim } m, \ C \in \mathbb{R}^{m \times m}$

Often ‘dominant’ invariant subspace needed ['dimension reduction']

Smallest eigenvalues needed in, e.g., electronic structure
Problems:

- Approximate the subspace
- Update it, e.g., when data changes
- Estimate its dimension (inexpensively)
- Exploit the subspace for certain calculations [e.g., model reduction]
- Track subspace of a sequence of matrices
- Find approximate common invariant subspace to a set of matrices
**Rayleigh-Ritz projection**

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

Question: How to extract good approximations to eigenvalues/ eigenvectors from this subspace?

**Answer:** Projection method

- Let $Q = [q_1, \ldots, q_m]$ an orthonormal basis of $X$.
- Express approximation as $\tilde{u} = Qy$ and obtain $y$ by writing

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

- Called *Rayleigh Ritz process* – Abbrev.: RR
**Subspace Iteration**

**Original idea:** projection technique onto a subspace of the form $Y = A^k X$ - Also called just the: “Power method”

- In practice: Replace $A^k$ by suitable polynomial [Chebyshev]

**ALGORITHM : 1** $[X_{new}, D] = \text{SubsIt}(A, X)$

1. **Start:** Select an initial system $X = [x_1, \ldots, x_m]$ and an initial polynomial $C_k$.

2. **Until convergence** Do:

3. Compute $\hat{X} = C_k(A)X$. [Original: $\hat{X} = A^kX$]

4. $[X_{new}, D] = \text{Rayleigh-Ritz}(A, \hat{X})$

5. If convergence satisfied Return.
   Else $X := X_{new}$ & select a new polynomial $C'_k$.

6. **EndDo**
Assumptions:

- $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_m| > |\lambda_{m+1}| \geq \cdots$
- $P = \text{eigenprojector (associated with } \lambda_1, \cdots, \lambda_m)$
- $\mathcal{L}_0 = \text{span}\{x_1, x_2, \ldots, x_m\}$. Assume:
- $\{Px_i\}_{i=1,\ldots,m}$ linearly independent.
- $\mathcal{P}_k = \perp$ projector onto $\mathcal{L}_k = \text{span}\{X_k\}$.

THEOREM: For each eigenvector $u_i$ of $A$, $i = 1, \ldots, m$, there exists a unique vector $s_i$ in the subspace $\mathcal{L}_0$ such that $Ps_i = u_i$. Moreover, the following inequality is satisfied

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

where $\epsilon_k$ tends to zero as $k$ tends to infinity.
Q: What Chebychev polynomial?

Typical scenario →

**Common thinking:** shift and scale $A$ to $B = (A - cI)/h$:

$$c = \frac{\lambda_{m+1} + \lambda_n}{2}, \quad h = \frac{\lambda_{m+1} - \lambda_n}{2}$$

Then: $p_k(t) = C_k(t)/C_k(\lambda_1)$

- Eigs of $B$ in $[-1, 1]$ are now the ‘unwanted’ eigenvalues

- Polynomial ‘optimal’ in some sense for each $\lambda_i, i \leq m$ individually - but not for the invariant subspace as a whole.
Krylov vs. subspace iteration

From the perspective of computing invariant subspaces

Krylov-type methods
+ Fast
+ Optimal in a certain sense
+ Requires one starting vector
  - Not easy to update
  - Changes in $A$ not allowed

Subspace iteration methods
+ Updates are easy
+ Geared toward subspaces [vs individual eigenvalues]
+ Tolerates changes in $A$
  – Slower

Important note: both types of methods require only matrix-vector products. Can get superior convergence with shift-and-invert [replace $A$ with $(A - \sigma I)^{-1}$ in Algorithms]. Issue: cost
Example: subspace iteration for Kohm-Sham equation

\[
\left[ -\frac{\nabla^2}{2} + V_{ion} + V_H + V_{xc} \right] \Psi(r) = E \Psi(r)
\]

With:

• Hartree potential (local)

\[ \nabla^2 V_H = -4\pi \rho(r) \]

• \( V_{xc} \) depends on functional. For LDA:

\[ V_{xc} = f(\rho(r)) \]

• \( V_{ion} \) = nonlocal – does not explicitly depend on \( \rho \)

\[ V_{ion} = V_{loc} + \sum_a P_a \]

• \( V_H \) and \( V_{xc} \) depend nonlinearly on eigenvectors:

\[ \rho(r) = \sum_{i=1}^{\text{occup}} |\psi_i(r)|^2 \]
Self-Consistent Iteration

Initial Guess for \( V \), \( V = V_{at} \)

Solve \((-\frac{1}{2} \nabla^2 + V) \psi_i = \epsilon_i \psi_i\)

Calculate new \( \rho(r) = \sum_i^{occ} |\psi_i|^2 \)

Find new \( V_H: -\nabla^2 V_H = 4\pi \rho(r) \)

Find new \( V_{xc} = f[\rho(r)] \)

\( V_{new} = V_{ion} + V_H + V_{xc} + \text{‘Mixing’} \)

If \( |V_{new} - V| < tol \) stop
The subspace filtering viewpoint

Given a basis \([v_1, \ldots, v_m]\), 'filter' each vector as

\[ \hat{v}_i = P_k(A)v_i \]

\( p_k = \) Low deg. polynomial [Chebyshev]

- Filtering step not used to compute eigenvectors accurately
- SCF & diagonalization loops merged
- Another viewpoint: nonlinear form of subspace iteration
Select initial $V = V_{at}$

Get initial basis $\{\psi_i\}$ (diag)

Calculate new $\rho(r) = \sum_i^{\text{occ}} |\psi_i|^2$

Find new $V_H$: $-\nabla^2 V_H = 4\pi\rho(r)$

Find new $V_{xc} = f[\rho(r)]$

$V_{new} = V_{ion} + V_H + V_{xc} + \text{‘Mixing’}$

If $|V_{new} - V| < tol$ stop

Filter basis $\{\psi_i\}$ (with $H_{new}$)+orth.

$V = V_{new}$

\[ Si_{525}H_{276} \], Polynomial deg. == 8. Single proc.

\[ Si_{9041}H_{1860} \] # PEs = 48; \( n_H = 2,992,832 \). Degree \( m = 8 \)

<table>
<thead>
<tr>
<th>method</th>
<th># ( A \times x )</th>
<th>SCF</th>
<th>CPU(s.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChebSI</td>
<td>124761</td>
<td>11</td>
<td>5946.69</td>
</tr>
<tr>
<td>ARPACk</td>
<td>142047</td>
<td>10</td>
<td>62026.37</td>
</tr>
<tr>
<td>TRLan</td>
<td>145909</td>
<td>10</td>
<td>26852.84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n_{state} )</th>
<th># ( A \times x )</th>
<th># SCF</th>
<th>total ( eV )</th>
<th>1st CPU</th>
<th>total CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>19015</td>
<td>4804488</td>
<td>18</td>
<td>-92.00412</td>
<td>102.12 h.</td>
<td>294.36 h.</td>
</tr>
</tbody>
</table>
The Grassmannian perspective

- Recall: Stiefel manifold (‘compact’ Stiefel manifold):
  \[ St(p, n) = \{ Y \in \mathbb{R}^{n \times p} : Y^T Y = I \}. \]

- Set of matrices with \( p \) orthonormal columns

- Grassmann manifold is the quotient manifold
  \[ G(p, n) = S(p, n)/O(p) \]

where \( O(p) \equiv \text{orthogonal group of unitary } p \times p \text{ matrices.} \)

- Each point on \( G(p, n) \equiv \text{a subspace of dimension } p \text{ of } \mathbb{R}^n \)

- Can be represented by a basis \( V \in St(p, n) \).

**Notation**: \([V]\), [it does not matter which basis \( V \) of is used]

**Tangent space of the Grassmann manifold at** \([\mathcal{Y}]\) **is the set of matrices** \(\Delta \in \mathbb{R}^{n \times p}\) **s.t.:**

\[
\mathcal{Y}^T \Delta = 0
\]

**The EAS paper (above) considers minimizing**

\[
F(\mathcal{Y}) = \frac{1}{2} \text{Tr} [\mathcal{Y}^T \mathcal{A} \mathcal{Y}]
\]

where \(\mathcal{Y}^T \mathcal{Y} = I\) **by a Newton approach**

**The gradient of** \(F(\mathcal{Y})\) **on the manifold at point** \([\mathcal{Y}]\) **is**

\[
G = (I - \mathcal{Y} \mathcal{Y}^T) \mathcal{A} \mathcal{Y}
\]
For Newton: We need to solve $\text{Hess}\Delta = -G$ on manifold

Notation: $\Pi = I - YY^T$, $C_Y = Y^TAY$

Newton leads to Sylvester equation:

$$\Pi[A\Delta - \Delta C_Y] = -\Pi AY$$

Solution: $\Delta = -Y + Z(Y^T Z)^{-1}$ where $Z$ solves

$$AZ - ZC_Y = Y$$
A few other well-known references

The Grassmannian perspective (continued)

Problem with these 2nd-order methods: Need to solve multiple systems of equations or a Sylvester equation at each step

Can we use Grassmannian perspective without inversion?

Idea: Use a gradient - or conjugate gradient - approach

Recall: On $G(p, n)$, gradient of objective function $\phi$ at $[Y]$ is

$$G = \nabla \phi_Y = (I - YY^T)AY \equiv AY - YC_Y$$

with $C_Y = Y^TAY$. 
**Gradient approach**

- Next iterate is of the following form ($\mu$ to be determined)

\[
\tilde{Y} = Y + \mu G
\]

- Direction of gradient will increase $\phi$ locally but new iterate must stay on manifold.

- Could follow a geodesic (EAS paper) ..

- Or follow a path along $G$ but implicitly re-project each $Y + \mu G$ on manifold, i.e., consider $[Y + \mu G]$
Can show

\[
\phi(\tilde{Y}) = \phi(Y) + \mu \|G\|_F^2 + \frac{\mu^2}{2} \text{Tr} [AY]^T \Pi A \Pi [AY]
\]

... and because \(Y^T G = 0\) we have:

\[
\tilde{Y}^T \tilde{Y} = [Y + \mu G]^T [Y + \mu G] = I + \mu^2 G^T G.
\]

Let: \(G^T G = UD_\beta U^T \equiv \text{spectral decomposition of } G^T G\)

Want: To orthonormalize \(\tilde{Y}\) without changing its span

Sol: Right-multiply \(\tilde{Y}\) by \(UD_\mu^{-1}\), i.e., define new \(Y\) as:

\[
Y(\mu) = \tilde{Y} UD_\mu^{-1} = (Y + \mu G) UD_\mu^{-1}.
\]

where:

\[
D_\mu \equiv [I + \mu^2 D_\beta]^{1/2}
\]
Set:

\[
Y_u = YU \\
\alpha_i = (Y_u^T A Y_u)_{ii} \\
D_{\alpha} = \text{Diag}(\alpha_i);
\]

\[
G_u = GU \\
\gamma_i = (G_u^T A G_u)_{ii} \\
D_{\gamma} = \text{Diag}(\gamma_i);
\]

Then:

\[
\phi(Y(\mu)) = \frac{1}{2} \text{Tr} \left[ I + \mu^2 D_\beta \right]^{-1} \left[ D_{\alpha} + 2\mu D_\beta + \mu^2 D_\gamma \right]
\]

This is a rational function \( \rightarrow \)

\[
\phi(Y(\mu)) = \frac{1}{2} \sum_{i=1}^{m} \frac{\alpha_i + 2\beta_i \mu + \gamma_i \mu^2}{1 + \beta_i \mu^2}
\]

Derivative of \( Y(\mu) \) \( \rightarrow \)

\[
\frac{dY(\mu)}{d\mu} = \sum_{i=1}^{m} \frac{\beta_i + (\gamma_i - \alpha_i \beta_i) \mu - \beta_i^2 \mu^2}{(1 + \beta_i \mu^2)^2}
\]
Each numerator is an inverted parabola: \[ \rightarrow \] then \[ \leftarrow \]

Easy to devise procedures to optimize \( \phi(Y(\mu)) \)

Careful in case \( \beta_i \)'s are small!

ALGORITHM : 2 Gradient Ascent algorithm

0. **Start:** Select initial \( Y \) such that \( Y^T Y = I \).
1. Compute \( G = AY - YC_Y \)
2. **While** \( \| G \|_F > \text{tol} \)
3. Compute and Diagonalize \( G^T G \) as \( G^T G = UD_\beta U^T \)
4. Compute \( D_\alpha, D_\gamma \)
5. Call `get_mu` to approximately maximize \( \phi(Y(\mu)) \)
6. Set \( Y := (Y + \mu G)U[I + \mu^2 D_\beta]^{-1/2} \)
7. Compute \( G = AY - YC_Y \)
8. **EndWhile**
Use of Conjugate Gradients [work in progress (!)]

- Can’t use perspective of linear CG [obj. function not quadratic]
- Also we are maximizing a function \([\phi(Y)]\)
- An approach based on a Polak-Ribiere formulation works quite well. New Conj. Direction \(P:\)

\[
P_{\text{new}} = P + \beta G_{\text{new}} \quad \text{where} \quad \beta = \frac{\langle G_{\text{new}} - G, G_{\text{new}} \rangle}{\langle G, G \rangle}
\]

- But we will also project new \(P\) on tangent space:

\[
P_{\text{new}} \leftarrow (I - YY^T)P_{\text{new}}
\]

- Since \(Y_{\text{new}}^T P = 0\) formulas similar to Grad. case available [Slightly more expensive]
Conjugate Gradients – Polak-Ribiere

ALGORITHM : 3 Conjugate Gradient Ascent algorithm

0. **Start:** Select initial $Y$ such that $Y^TY = I$.
1. Compute $G = AY - YC_Y$; Set $P := G$
2. **While** $\|G\|_F > tol$
3. Call `get_mu` to approximately maximize $\phi(Y(\mu))$
4. Set $[Y, R] = qr(Y + \mu P, 0)$ [Matlab]
5. Compute $G_{new} = AY - YC_Y$
6. Compute $\beta = \frac{\langle G_{new} - G, G_{new} \rangle}{\langle G, G \rangle}$ and set:
7. $P_{new} := G_{new} + \beta P$ and $G := G_{new}$
8. $P_{new} := (I - YY^T)P_{new}$
9. **EndWhile**

Luminy, 11-12-2021 p. 25
A few numerical tests

Test cases:

1) Finite Difference Laplacean on $35 \times 40$ grid ($n = 1, 400$)
2) Matrix $Ukerbe1$ from SuiteSparse collection →

- All tests: $m = \text{Subsp. dim. } \equiv 8$

- For Standard Subspace iteration – we apply optimal shift so $A \rightarrow A - \sigma I$ [where $\sigma = (\lambda_n + \lambda_9)/2$]

Small Laplacean \([35 \times 40 \text{ grid}, n = 1400, \text{nnz} = 6850]\)

Trace of \(C_Y\) vs. its

Invariance Meas. vs. its

Performance measures: 1) Trace; 2) Invariance \(\|AY - YC_Y\|_1\)
Matrix \texttt{ukerbe1} \([n = 5,981, \text{nnz} = 15704]\)

In the context of solving linear systems, we analyze the matrix \texttt{ukerbe1} with dimension \(n = 5,981\) and \(\text{nnz} = 15704\). The plots illustrate the trace of \(C_Y\) and its invariance measurements against iterations. The traces show the evolution of the trace over iterations for different methods, indicating convergence behavior. The invariance measurements provide insights into the stability of the solution across iterations for various algorithms.
JOINT DIAGONALIZATION
Application: Joint Diagonalization

- Current joint work with Karim Seghouane

Standard **Orthogonal Joint Diagonalization (OJD)**: given \( p \) matrices \( A_1, \ldots, A_p \) find a unitary matrix \( Q \) such that each \( Q^T A_i Q \) is close to a diagonal.

- Main applications: Blind Source Separation, ICA, ...

Typical formal formulation:

\[
(\text{Off}(X) \equiv X - \text{Diag}(X))
\]

\[
\min_{Q \in O_n} \sum_{i=1}^{p} \|\text{Off}(Q^T A_i Q)\|_F^2
\]

- Deals with the case where each \( A_i \) is dense.
- Well-known algorithm: A Jacobi-like method [Cardoso & Souloumiac, ’96]. Cost: \( O(pn^3) \)
Large matrices: Use a subspace approach

- Previous criterion and obj. function do not work
- Roughly: Seek an $n \times k$ matrix ($k \ll n$) such that

1) $A_i Q - Q D_i$ small for some diagonal $D_i$ [Invariance]
2) $Q$ near dominant invariant subspace for each $A_i$

New objective function:

$$f(Q, D_1, \ldots, D_p) = \sum_{i=1}^{p} \| A_i Q - Q D_i \|_F^2.$$

- Does not specify which invariant subspace is selected [we let algorithm take care of this]
ALGORITHM : 4 Subspace iteration for partial JOD

Start : select initial $Q$ such $Q^TQ = I$

While { Not converged }

For $j = 1, \cdots, p$

Compute $X_j = A_jQ$

EndFor

Let $X = [X_1, \cdots, X_p]$

Compute $X = Q\Sigma V^T$ the SVD of $X$

Define $Q := Q(:, 1:k)$ [Matlab notation used]

EndWhile

Alternative: Similar algorithm to Grassmann gradient ascent
- but uses combined objective function (to maximize)

$$\psi(Y) = \frac{1}{2} \sum_{i=1}^p \text{Tr} [Y^T A_i Y] - \eta \sum_{i=1}^p \| A_i Q - QC_{Q,i} \|^2_F$$
Updating the SVD (E. Vecharynski and YS’13)

**Problem**
Given partial SVD of $X$, to get partial SVD of $X_{new}$

- Example: In information retrieval, updates of the form $X_{new} = [X, D]$ (documents added) where $D \in \mathbb{R}^{n \times p}$

- Assume $X \approx X_k \equiv U_k \Sigma_k V_k^T$

- Compute $D_k = (I - U_k U_k^T)D$ and its QR factorization:
$$[\hat{U}_p, R] = qr(D_k, 0), \ R \in \mathbb{R}^{p \times p}, \ \hat{U}_p \in \mathbb{R}^{n \times p} \quad \rightarrow$$

$$[X_k, D] = [U_k, \hat{U}_p]H_D \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}^T; \ H_D \equiv \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & R \end{bmatrix}$$

**Zha–Simon (’99):** Compute SVD of $H_D$ & get approximate SVD from above $\rightarrow$ This is a Rayleigh-Ritz projection method for the SVD [E. Vecharynski & YS 2013]
When the number of updates is large ZS becomes costly.

Idea: Replace $\hat{U}_p$ by a low dimensional approximation:

Use $\bar{U}$ of the form $\bar{U} = [U_k, Z_l]$ instead of $\bar{U} = [U_k, \hat{U}_p]$

$Z_l == \text{rank-}l \text{ approximation of } D_k = (I - U_k U_k^T) D$

Details of Experiments skipped but: we get slightly improved precision at a much lower cost.
RANK ESTIMATION
What dimension to use in dimension reduction?

- Important problem in signal processing applications, machine learning, ...
- Often: a certain rank is selected ad-hoc. Dimension reduction is application with this “guessed” rank.
- \( k \) = intrinsic rank of data. Can we estimate it?
- Recall: Numerical rank:

\[
\epsilon\text{-rank} = \text{number } k \text{ of sing. values } > \epsilon
\]
Determining rank by eigenvalue counts

- Idea: count eigenvalues of $A^T A$ (or $AA^T$) that are $> \epsilon^2$.
- Let $A$ be a Hermitian matrix with eigenpairs $(\lambda_i, u_i)$, where

\[
\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n
\]

- Given: $a, b$ such that $\lambda_1 \leq a \leq b \leq \lambda_n$.
- Want: $\mu_{[a,b]} = \text{number of } \lambda_i \text{'s} \in [a, b]$.

- Standard method: Use Sylvester inertia theorem. Requires two $LDL^T$ factorizations $\rightarrow$ expensive!
Alternative: Exploit trace of the eigen-projector:

\[ P = \sum_{\lambda_i \in [a \ b]} u_i u_i^T. \]

We know that:

\[ \text{Tr} (P) = \mu_{[a,b]} \]

Goal now: approximate \( \text{Tr} (P) \)

\( P \) not available but:

\[ P = h(A) \text{ where } h(t) = \begin{cases} 1 & \text{if } t \in [a \ b] \\ 0 & \text{otherwise} \end{cases} \]

Can approximate \( h(t) \) by a polynomial \( \psi \)

Then use statistical estimator for approximating \( \text{Tr} (\psi(A)) \)

Details: [E. Di Napoli, E. Polizzi, and Y.S., 2013]
**Alternative: ‘Density of States’ (DOS)**

Formally, the Density Of States (DOS) of a matrix $A$ is

$$
\phi(t) = \frac{1}{n} \sum_{j=1}^{n} \delta(t - \lambda_j),
$$

where: $\delta$ is the Dirac $\delta$-function or Dirac distribution

- Term used by mathematicians: Spectral Density

- $\phi(t)$ == a probability distribution function == probability of finding eigenvalues of $A$ in a given infinitesimal interval near $t$.

- Many uses in Solid-State physics

The Kernel Polynomial Method

- Used by Chemists to calculate the DOS – see Silver and Röder’94, Wang ’94, Drabold-Sankey’93, + others
- Basic idea: expand DOS into Chebyshev polynomials
- Use trace estimators [discovered independently] to get traces needed in calculations
- Assume change of variable done so eigenvalues lie in $[-1, 1]$.
- Include the weight function in the expansion so expand:

$$
\hat{\phi}(t) = \sqrt{1 - t^2} \phi(t) = \sqrt{1 - t^2} \times \frac{1}{n} \sum_{j=1}^{n} \delta(t - \lambda_j).
$$

Then, (full) expansion is: 

$$
\hat{\phi}(t) = \sum_{k=0}^{\infty} \mu_k T_k(t).
$$
An example: The Benzene matrix

```
>> TestKpmDos
  Matrix Benzene n = 8219  nnz = 242669
  Degree = 40  # sample vectors = 10
  Elapsed time is 0.235189 seconds.
```
Note: number of eigenvalues in an interval \([a, b]\) is
\[
\mu_{[a,b]} = \int_a^b \sum_j \delta(t - \lambda_j) \, dt \equiv \int_a^b n\phi(t) \, dt.
\]

If we use KPM to approximate \(\phi(t) = \hat{\phi}(t) / \sqrt{1 - t^2}\) then
\[
\mu_{[a,b]} \approx \sum_{k=0}^m \mu_k \int_a^b T_k(t) \sqrt{1 - t^2} \, dt
\]

A little calculation shows that the result obtained in this way is identical with that of the eigenvalue count by Cheb expansion.
Use of the Lanczos Algorithm

- Lanczos process builds orthogonal polynomials wrt to:
  \[ \langle p, q \rangle = \int p(t)q(t)dt \equiv (p(A)v_1, q(A)v_1) \]

- Let \( \theta_i, y_i \; i = 1 \cdots m \) be the eigenvalues / eigenvectors of tridiagonal matrix \( T_m \) [Ritz values]

**Idea:** exploit relation of Lanczos with (discrete) orthogonal polynomials and related Gaussian quadrature:

\[
\int p(t)dt \approx \sum_{i=1}^{m} a_i p(\theta_i) \quad a_i = [e_1^T y_i]^2
\]

- Formula exact when \( p \) is a polynomial of degree \( \leq 2m + 1 \)
Consider now \( \int p(t) \, dt = \langle p, 1 \rangle \equiv (\text{Stieltjes}) \text{ integral } \equiv \sum \beta_i^2 p(\lambda_i) \equiv \langle \phi_v, p \rangle \)

where \( v = \sum \beta_i u_i \) = eigen-expansion of \( v \),

\[ \phi_v = \sum \beta_i^2 \delta \lambda_i \]

Note: Ideal case \( \beta_i = 1/\sqrt{n} \) yields \( \phi_v \equiv \phi \)

Then \( \langle \phi_v, p \rangle \approx \sum a_i p(\theta_i) = \sum a_i \langle \delta_{\theta_i}, p \rangle \rightarrow \)

\[ \phi_v \approx \sum a_i \delta_{\theta_i} \]

Statistically produce choice \( \beta_i \equiv 1/\sqrt{n}, \forall i \), average results over several vectors \( v \) with \( \|v\|_2 = 1 \).
Back to estimating the rank: Threshold selection

- Recall: numerical rank = # sing. values ≥ \( \epsilon \)

Q: How to select \( \epsilon \)?

A: Obtain it from the DOS function

Exact DOS plots for three different types of matrices.
➢ To find: point immediately following the initial sharp drop observed.

➢ Simple idea: use derivative of DOS function $\phi$

➢ For an $n \times n$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$:

$$\epsilon = \min\{t : \lambda_1 \leq t \leq \lambda_n, \phi'(t) = 0\}.$$  

➢ In practice replace by

$$\epsilon = \min\{t : \lambda_1 \leq t \leq \lambda_n, |\phi'(t)| \geq \text{tol}\}.$$
Experiments

(A) The DOS found by KPM.
(B) Approximate rank estimation by The Lanczos method for the example netz4504.
Approximate Rank Estimation of various matrices

<table>
<thead>
<tr>
<th>Matrix Name</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpi_ceria3d (linear programming)</td>
<td>3,576</td>
</tr>
<tr>
<td>S80PI_n1 (model reduction prbm.)</td>
<td>4,028</td>
</tr>
<tr>
<td>ukerbe1 (2D finite elem. prbm.)</td>
<td>5,981</td>
</tr>
<tr>
<td>Erdos992 (collaboration network)</td>
<td>6,100</td>
</tr>
<tr>
<td>Geom (computl. geometry)</td>
<td>7,343</td>
</tr>
<tr>
<td>California (web search)</td>
<td>9,664</td>
</tr>
<tr>
<td>C-40 (non-linear optimization)</td>
<td>9,941</td>
</tr>
<tr>
<td>Matrices</td>
<td>Threshold $\epsilon$</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>lpi_ceria3d</td>
<td>28.19</td>
</tr>
<tr>
<td>S80PI_n1</td>
<td>1.76</td>
</tr>
<tr>
<td>ukerbe1</td>
<td>0.169</td>
</tr>
<tr>
<td>Erdos992</td>
<td>3.96</td>
</tr>
<tr>
<td>Geom</td>
<td>90</td>
</tr>
<tr>
<td>California</td>
<td>0.02</td>
</tr>
<tr>
<td>C-40</td>
<td>48160.4</td>
</tr>
</tbody>
</table>

Concluding remarks

- Many tasks in applications deal with invariant subspaces
- Beneficial to explore algorithms that treat invariant subspaces as Grassmannian objects
- Krylov subspace methods not best choice for types of problems that arise in some applications ...
- ... but they are amazingly powerful for other tasks [e.g. Spectral densities]