UNIVERSITY OF MINNESOTA TWIN CITIES

Supspace Iteration and Variants, Revisited

Yousef Saad

Department of Computer Science and Engineering

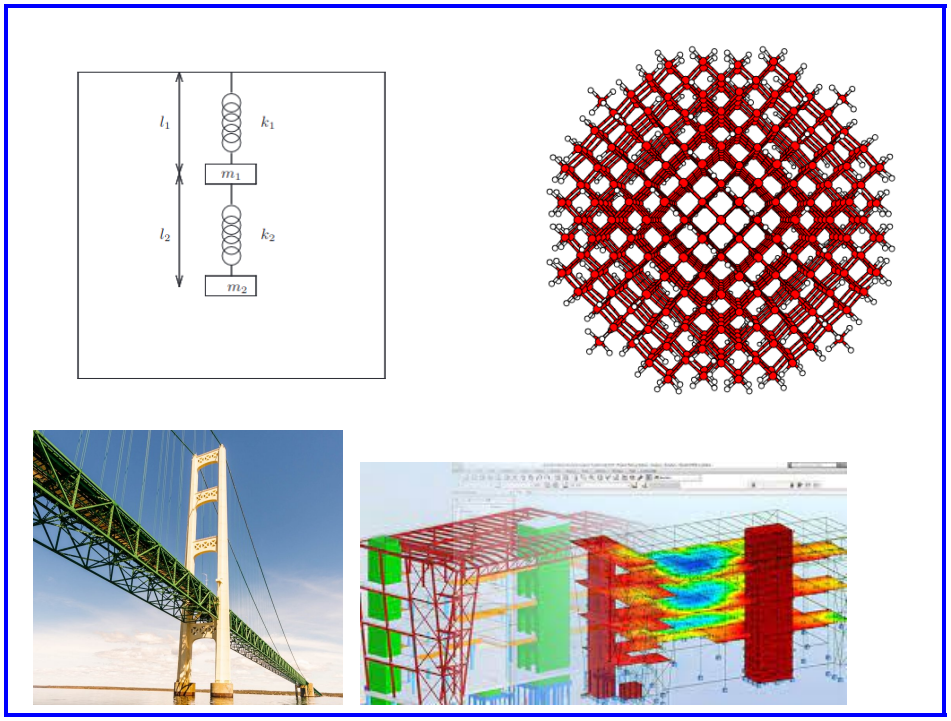
University of Minnesota

Luminy

Nov. 8–13, 2021

Introduction & Background

➤ 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,..]
- ...

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

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

$Y =$ basis of subspace $\mathcal{X}$ 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 $\boxed{Y = A^k X}$ - Also called just the: "Power method"

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

ALGORITHM : 1 $[X_{new}, D]$ = *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:* $\hat{X} = A^k X$*]*
4. $[X_{new}, D]$ = *Rayleigh-Ritz* $(A, \hat{X})$
5. *If convergence satisfied Return.*
   Else $X := X_{new}$ *& select a new polynomial* $C'_{k'}$
6. *EndDo*

Assumptions:

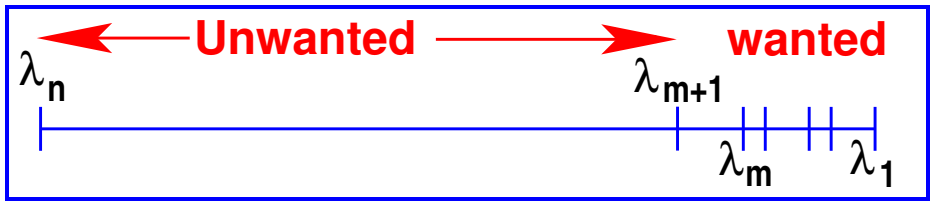
- $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_m| > |\lambda_{m+1}| \geq \ldots$
- P = eigenprojector (associated with $\lambda_1, \cdots, \lambda_m$)
- $\mathcal{L}_0 = span\{x_1, x_2, \ldots, x_m\}$. Assume:
- $\{Px_i\}_{i=1,\ldots,m}$ linearly independent.
- $\mathcal{P}_k = \perp$ projector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

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

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

where ϵ_k tends to zero as k tends to infinity.

Q: What Chebychev polynomial?
Typical scenario $\longrightarrow$

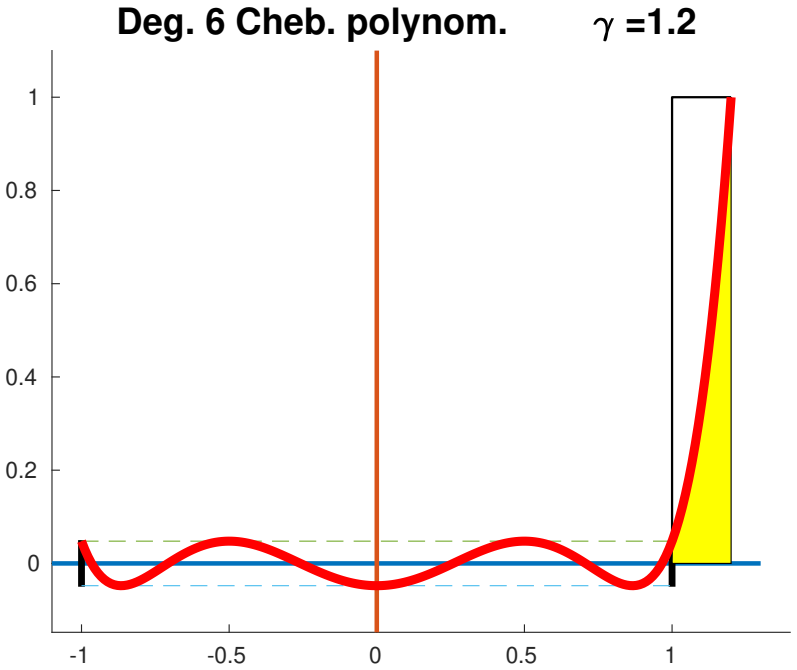


Common thinking: shift and scale A to $B = (A - cI)/h$:

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

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 λ_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	*Subspace iteration methods*
+ Fast	+ Updates are easy
+ Optimal in a certain sense	+ Geared toward subspaces [vs individual eigenvalues]
+ Requires one starting vector	+ Tolerates changes in A
– Not easy to update	– Slower
– Changes in A not allowed	

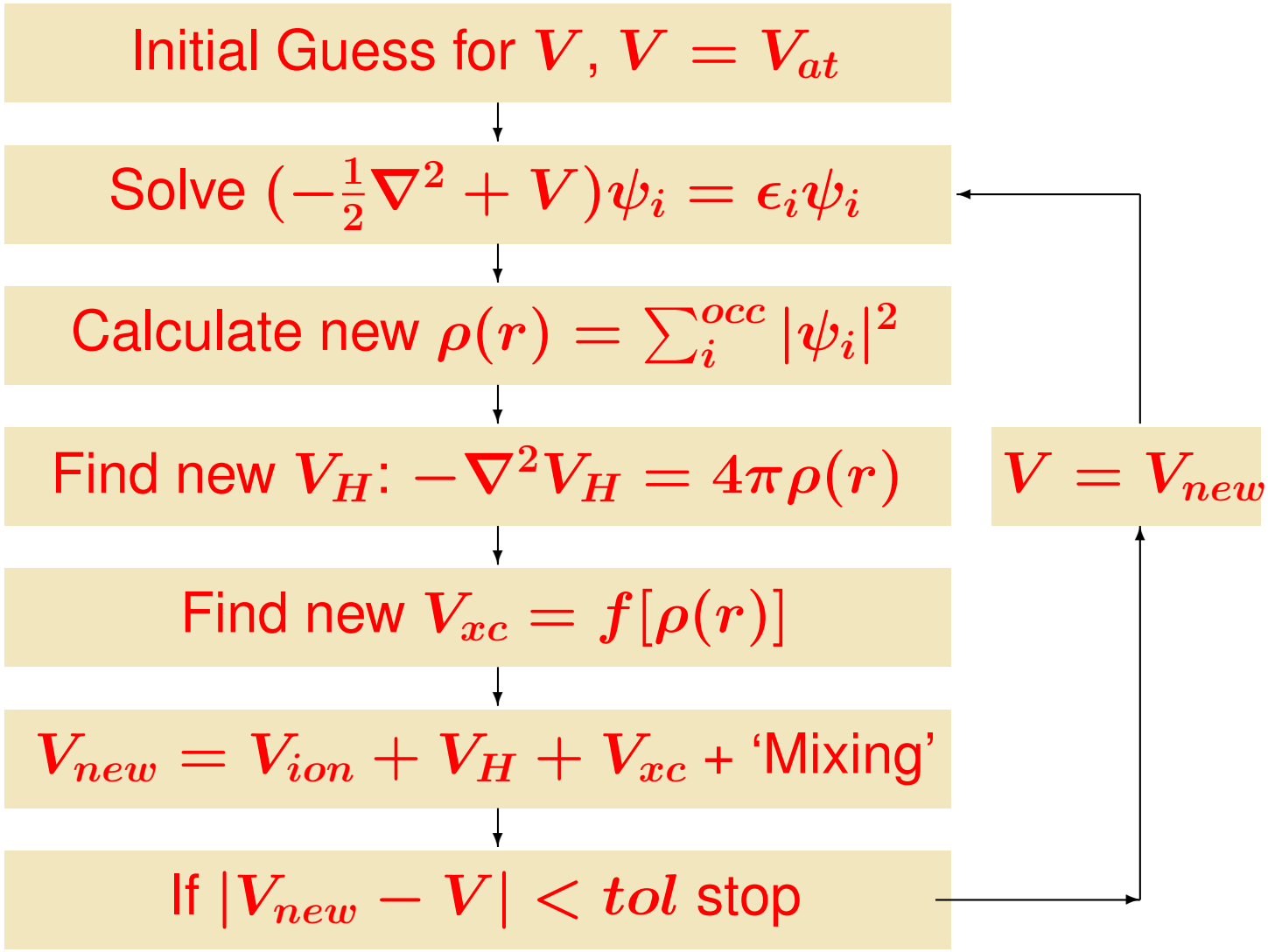
Important note: both types of methods require only matrix-vector products. Can get superior convergence with shift-and-invert [replace A with $(A - \sigma I)^{-1}$ in Algorithms]. Issue: cost

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 ρ $$V_{ion} = V_{loc} + \sum_a P_a$$
- V_H and V_{xc} depend nonlinearly on eigenvectors: $$\rho(r) = \sum_{i=1}^{occup} |\psi_i(r)|^2$$

Self-Consistent Iteration



The subspace filtering viewpoint

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

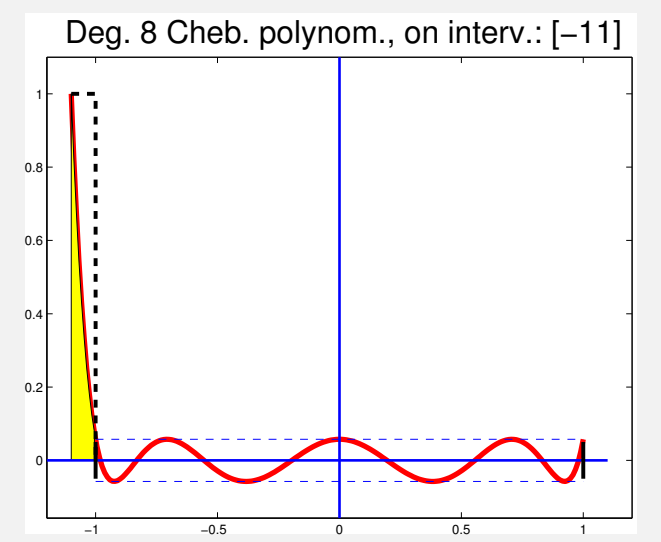
$$\hat{v}_i = P_k(A)v_i$$

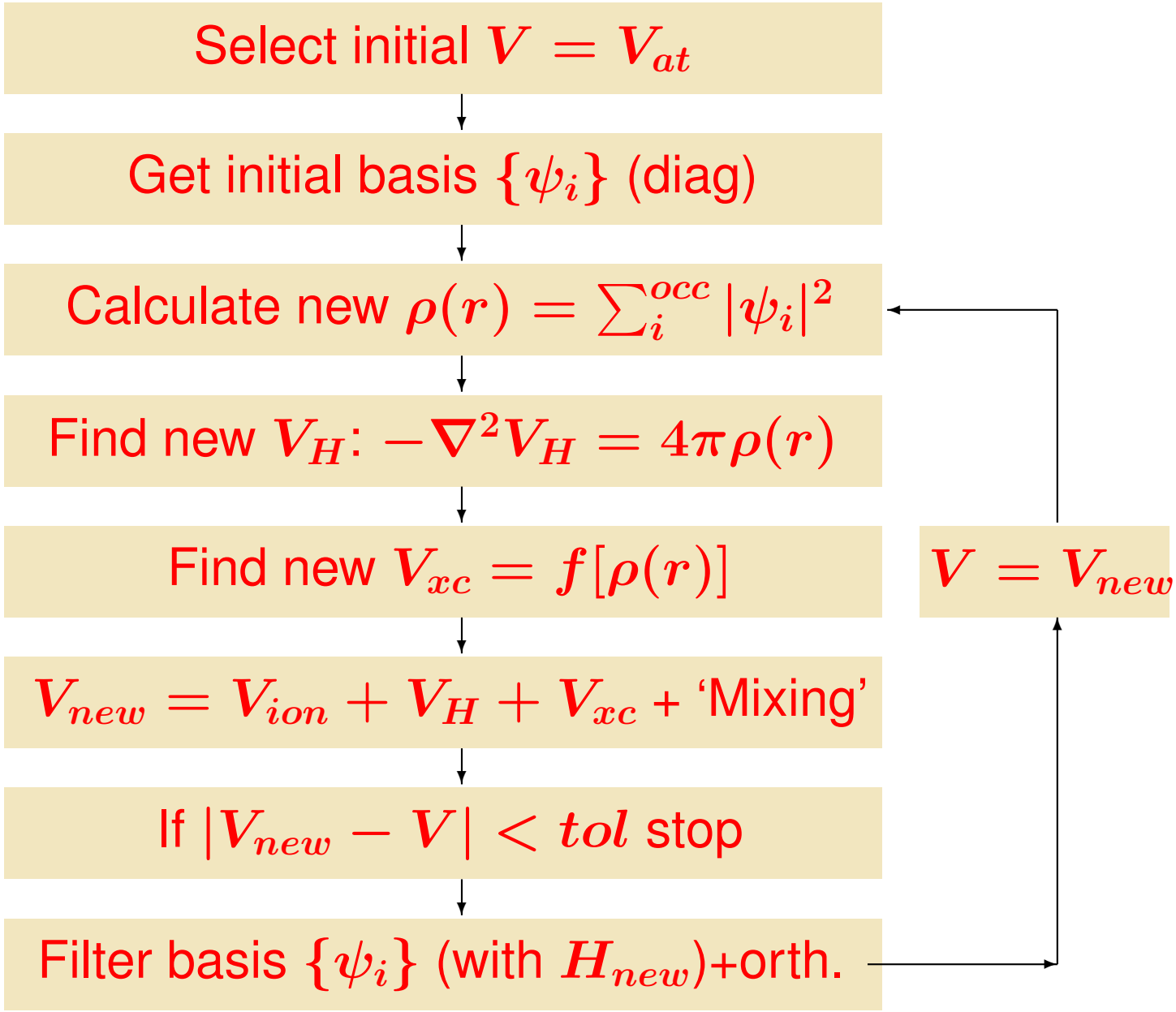
➤ p_k = Low deg. polynomial [Chebyshev]

➤ 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 $\rho(r) = \sum_i^{occ} |\psi_i|^2$
Find new V_H: $-\nabla^2 V_H = 4\pi\rho(r)$
Find new $V_{xc} = f[\rho(r)]$
$V_{new} = V_{ion} + V_H + V_{xc}$ + 'Mixing'
If $|V_{new} - V| < tol$ stop
Filter basis $\{\psi_i\}$ (with H_{new})+orth.
$V = V_{new}$

Yunkai Zhou, Y.S., Murilo L. Tiago, and James R. Chelikowsky, *Parallel Self-Consistent-Field Calculations with Chebyshev Filtered Subspace Iteration*, Phy. Rev. E, vol. 74, p. 066704 (2006)

$Si_{525}H_{276}$, Polynomial deg. == 8. Single proc.

method	# $A*x$	SCF	CPU(s.)
ChebSI	124761	11	5946.69
ARPACK	142047	10	62026.37
TRLan	145909	10	26852.84

$Si_{9041}H_{1860}$ # PEs = 48; n_H =2,992,832. Degree $m = 8$

n_{state}	# $A*x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
19015	4804488	18	-92.00412	102.12 h.	294.36 h.

The Grassmannian perspective

- Recall: Stiefel manifold ('compact' Stiefel manifold):

$$\boldsymbol{St(p,n) = \{Y \in \mathbb{R}^{n\times p} : Y^T Y = I\}.}$$

- Set of matrices with p orthonormal columns
- Grassmann manifold is the quotient manifold

$$\boldsymbol{G(p,n) = S(p,n)/O(p)}$$

where $O(p) \equiv$ orthogonal group of unitary $p \times p$ matrices.

- Each point on $G(p,n) \equiv$ a subspace of dimension p 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]

- A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIMAX, 20 (1999)

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

$$\boldsymbol{Y}^T \boldsymbol{\Delta} = \boldsymbol{0}$$

➤ The EAS paper (above) considers minimizing

$$\boldsymbol{F}(\boldsymbol{Y}) = \tfrac{1}{2}\mathrm{Tr}\,[\boldsymbol{Y}^T \boldsymbol{A} \boldsymbol{Y}]$$

where $\boldsymbol{Y}^T \boldsymbol{Y} = \boldsymbol{I}$ by a Newton approach

➤ The gradient of $\boldsymbol{F}(\boldsymbol{Y})$ on the manifold at point $[\boldsymbol{Y}]$ is

$$\boldsymbol{G} = (\boldsymbol{I} - \boldsymbol{Y}\boldsymbol{Y}^T)\boldsymbol{A}\boldsymbol{Y}$$

- For Newton: We need to solve Hess$\boldsymbol{\Delta} = -\boldsymbol{G}$ on manifold
- Notation: $\boldsymbol{\Pi} = \boldsymbol{I} - \boldsymbol{Y}\boldsymbol{Y}^{\boldsymbol{T}}, \boldsymbol{C_Y} = \boldsymbol{Y}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{Y}$
- Newton leads to Sylvester equation:

$$\boldsymbol{\Pi}[\boldsymbol{A}\boldsymbol{\Delta} - \boldsymbol{\Delta}\boldsymbol{C_Y}] = -\boldsymbol{\Pi}\boldsymbol{A}\boldsymbol{Y}$$

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

$$\boldsymbol{A}\boldsymbol{Z} - \boldsymbol{Z}\boldsymbol{C_Y} = \boldsymbol{Y}$$

A few other well-known references

1. P. -A. Absil, R. Mahony, R. Sepulchre and P. Van Dooren *"A Grassmann-Rayleigh Quotient Iteration for Computing Invariant Subspaces"*, SIAM Review, (2002)
2. P. A. Absil, R. Mahony and R. Sepulchre, *Riemannian Geometry of Grassmann Manifolds with a View on Algorithmic Computation*, Acta Applicandae Mathematicae, 80 (2004)
3. G. W. Stewart, *"Error and perturbation bounds for subspaces associated with certain eigenvalue problems"*, SIAM Rev., 15 (1973)
4. J. W. Demmel, *"Three methods for refining estimates of invariant subspaces"*, Computing 38 (1987)
5. F. Chatelin, *Simultaneous Newton's iterations for the eigenproblem*, Proc. Oberwolfach Conference (1984)
6. A. Sameh, J. Wisniewski, *The TraceMin algorithm*, 1982.

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 $\boldsymbol{G(p,n)}$, gradient of objective function ϕ at $[\boldsymbol{Y}]$ is

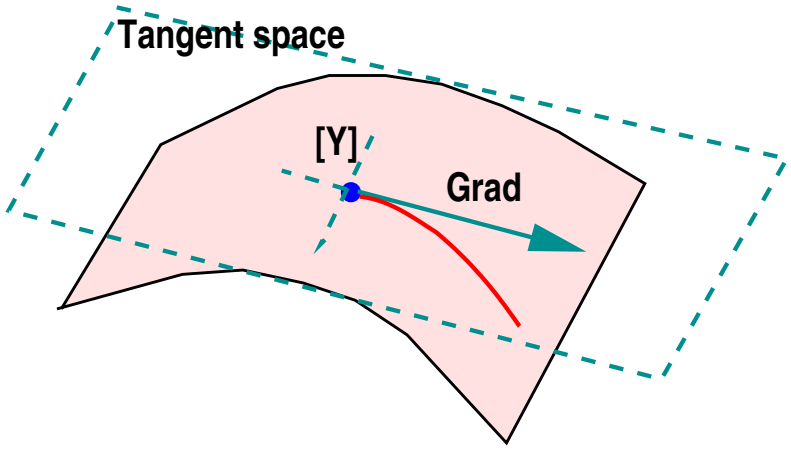
$$\boldsymbol{G = \nabla\phi_Y = (I - YY^T)AY \equiv AY - YC_Y}$$

with $\boldsymbol{C_Y = Y^T AY}$.

Gradient approach

➤ Next iterate is of the following form (μ to be determined)

$$\tilde{Y} = Y + \mu G,$$



➤ Direction of gradient will increase ϕ 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 $Y^TG = 0$ we have:

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

- Let: $G^TG = UD_\beta U^T \equiv$ spectral decomposition of G^TG
- Want: To orthonormalize $\tilde{Y}$ without changing its span
- Sol: Right-multiply $\tilde{Y}$ by UD_μ^{-1}, i.e., define new Y as:

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

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

Set:

$$\begin{aligned} \boldsymbol{Y_u} &= \boldsymbol{YU} \\ \boldsymbol{\alpha_i} &= (\boldsymbol{Y_u^T A Y_u})_{ii} \\ \boldsymbol{D_\alpha} &= \text{Diag}(\boldsymbol{\alpha_i}); \end{aligned}$$

$$\begin{aligned} \boldsymbol{G_u} &= \boldsymbol{GU} \\ \boldsymbol{\gamma_i} &= (\boldsymbol{G_u^T A G_u})_{ii} \\ \boldsymbol{D_\gamma} &= \text{Diag}(\boldsymbol{\gamma_i}); \end{aligned}$$

Then:

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

This is a rational function $\rightarrow$

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

Derivative of $\boldsymbol{Y}(\boldsymbol{\mu}) \rightarrow$

$$\frac{\boldsymbol{dY}(\boldsymbol{\mu})}{\boldsymbol{d\mu}} = \sum_{i=1}^{m} \frac{\boldsymbol{\beta_i} + (\boldsymbol{\gamma_i} - \boldsymbol{\alpha_i\,\beta_i})\boldsymbol{\mu} - \boldsymbol{\beta_i^2\mu^2}}{\boldsymbol{(1 + \beta_i\mu^2)^2}}$$

- Each numerator is an inverted parabola: ↗ then ↘
- Easy to devise procedures to optimize $\phi(\boldsymbol{Y}(\mu))$

Z Careful in case β_i's are small !

ALGORITHM : 2 *Gradient Ascent algorithm*

0. ***Start:*** *Select initial* $\boldsymbol{Y}$ *such that* $\boldsymbol{Y^T Y = I}$.
1. *Compute* $\boldsymbol{G = AY - YC_Y}$
2. ***While*** $\|\boldsymbol{G}\|_F > \boldsymbol{tol}$
3. *Compute and Diagonalize* $\boldsymbol{G^T G}$ *as* $\boldsymbol{G^T G = U D_\beta U^T}$
4. *Compute* $\boldsymbol{D_\alpha, D_\gamma}$
5. *Call* `get_mu` *to approximately maximize* $\phi(\boldsymbol{Y}(\mu))$
6. *Set* $\boldsymbol{Y := (Y + \mu G)U[I + \mu^2 D_\beta]^{-1/2}}$
7. *Compute* $\boldsymbol{G = AY - YC_Y}$
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 [$\phi(\boldsymbol{Y})$]
- An approach based on a Polak-Ribiere formulation works quite well. New Conj. Direction $\boldsymbol{P}$:

$$\boldsymbol{P}_{new} = \boldsymbol{P} + \beta \boldsymbol{G}_{new} \quad \text{where} \quad \beta = \frac{\langle G_{new} - G, G_{new} \rangle}{\langle G, G \rangle}$$

- But we will also project new $\boldsymbol{P}$ on tangent space:

$$\boldsymbol{P}_{new} \leftarrow (\boldsymbol{I} - \boldsymbol{Y}\boldsymbol{Y}^T)\boldsymbol{P}_{new}$$

- Since $\boldsymbol{Y}_{new}^T \boldsymbol{P} = \boldsymbol{0}$ formulas similar to Grad. case available [Slightly more expensive]

Conjugate Gradients – Polak-Ribiere

ALGORITHM : 3 *Conjugate Gradient Ascent algorithm*

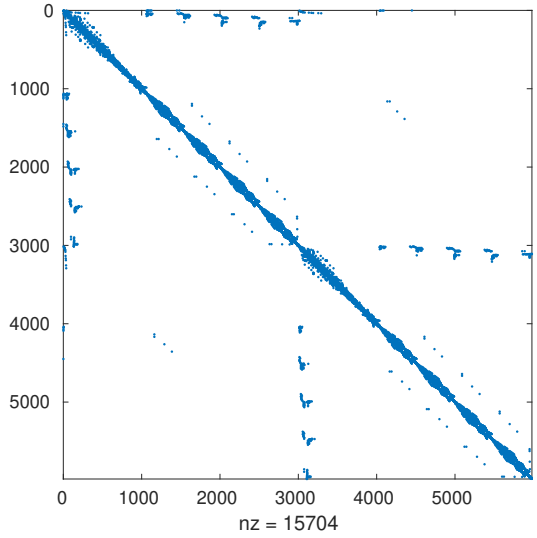
0. **Start:** Select initial Y such that $Y^T Y = I$.
1. Compute $G = AY - YC_Y$; Set $P := G$
2. **While** $\|G\|_F > tol$
3. Call `get_mu` to approximately maximize $\phi(Y(\mu))$
4. Set $[Y, R] = qr(Y + \mu P, 0)$ [Matlab]
5. Compute $G_{new} = AY - YC_Y$
6. Compute $\beta = \frac{\langle G_{new}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

A few numerical tests

Test cases:

1) Finite Difference Laplacean on 35×40 grid ($n = 1,400$)

2) Matrix *Ukerbe1* from `SuiteSParse` collection $\rightarrow$

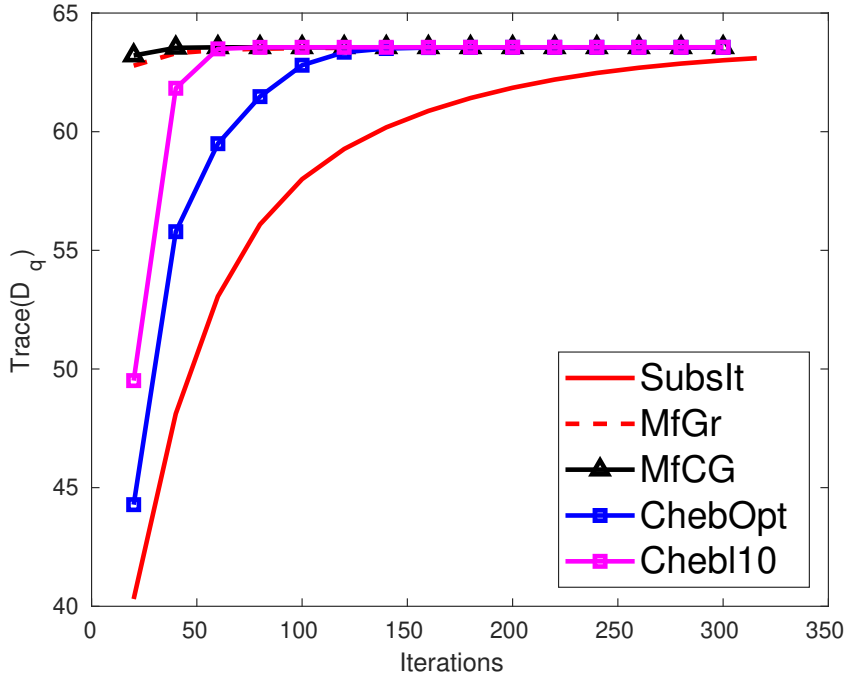


➤ All tests: $m =$ Subsp. dim. $\equiv 8$

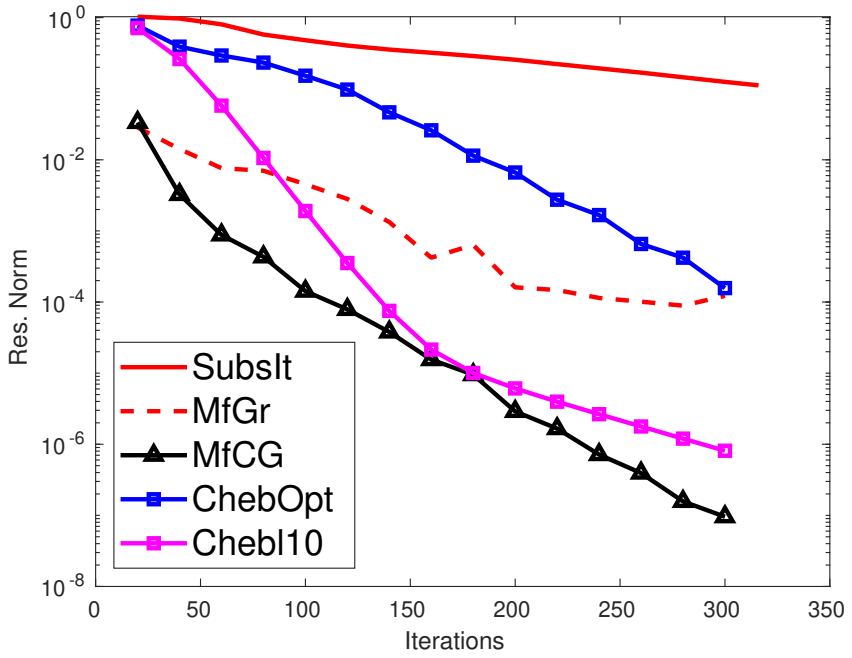
➤ For Standard Subspace iteration – we apply optimal shift so $A \rightarrow A - \sigma I$ [where $\sigma = (\lambda_n + \lambda_9)/2$]

➤ Tests: 1) Standard subspace iteration 2) Manifold Gradient method and 3) Conj. Gradient version of manifold SubsitMf, 4) Chebyshev subspace iteration

Small Laplacean [35×40 grid, $n = 1400, nnz = 6850$]



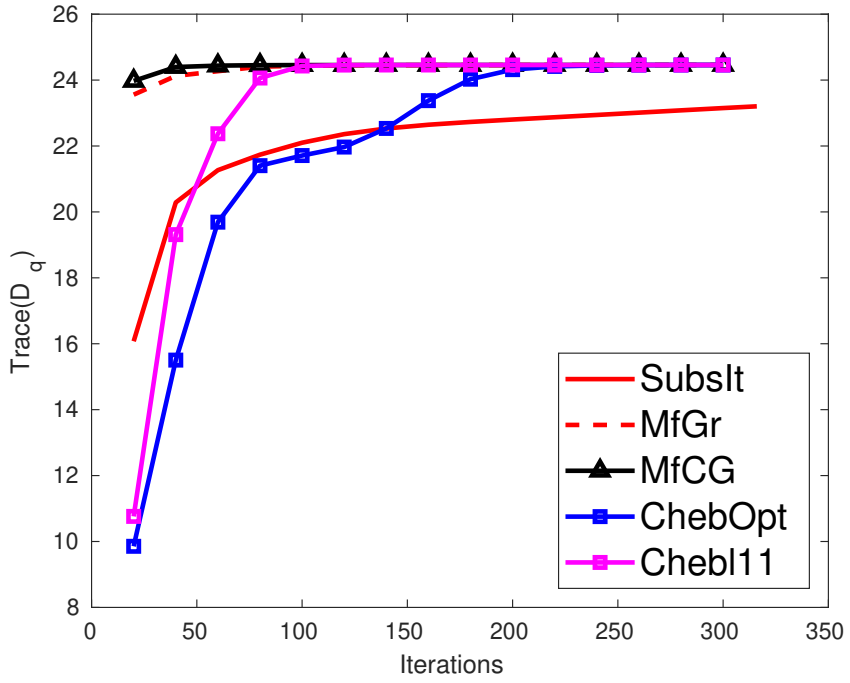
Trace of C_Y vs. its



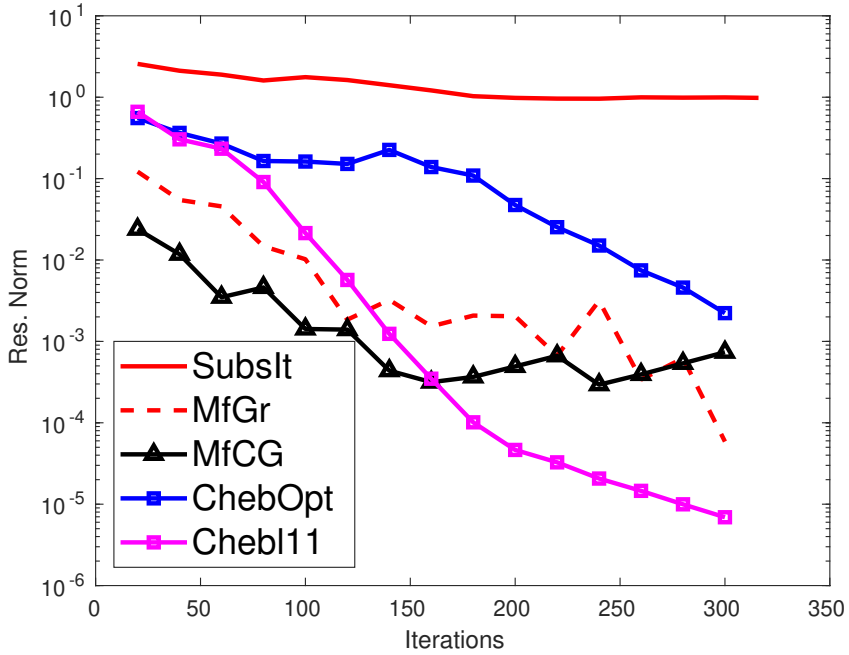
Invariance Meas. vs. its

Performance measures: 1) Trace; 2) Invariance $\|AY - YC_Y\|_1$

Matrix **Ukerbe1** $[n = 5,981, nnz = 15704]$



Trace of C_Y vs. its



Invariance Meas. vs. its

JOINT DIAGONALIZATION

Application: Joint Diagonalization

➤ Current joint work with Karim Seghouane

Standard Orthogonal Joint Diagonalization (OJD): given p matrices $A_1, \cdots, A_p$ 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:
(Off$(X) \equiv X - Diag(X)$)

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

➤ 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:

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

- 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* $Q^T Q = I$
While *{ Not converged }*
  For $j = 1, \cdots, p$
    Compute $X_j = A_j Q$
  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)

$$\psi(Y) = \frac{1}{2} \sum_{i=1}^{p} \mathrm{Tr}\,[Y^T A_i Y] \; - \eta \sum_{i=1}^{p} \|A_i Q - Q C_{Q,i}\|_F^2$$

Updating the SVD (E. Vecharynski and YS'13)

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

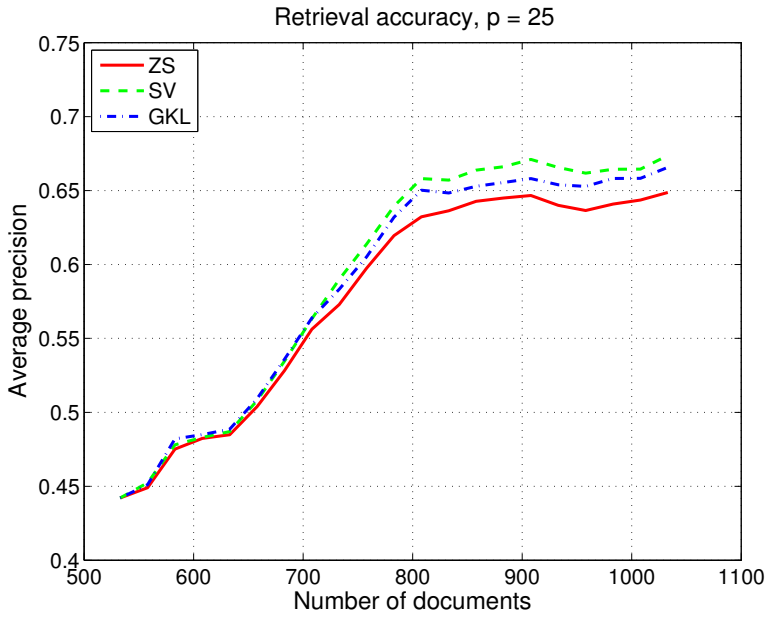
- Example: In information retrieval, updates of the form $X_{new} = [X, D]$ (documents added) where $D \in \mathbb{R}^{n\times p}$
- Assume $X \approx X_k \equiv U_k \Sigma_k V_k^T$
- Compute $D_k = (I - U_k U_k^T) D$ and its QR factorization:

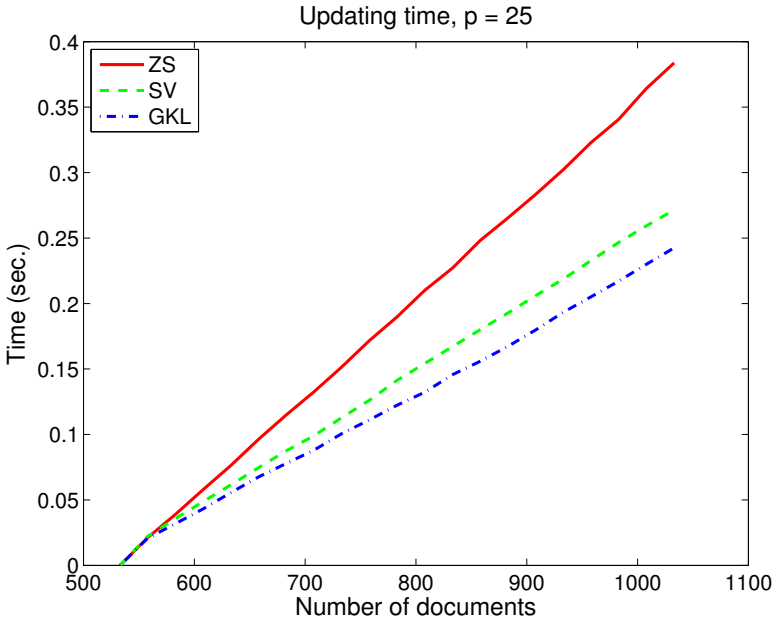
$$[\hat{U}_p, R] = qr(D_k, 0),\ R \in \mathbb{R}^{p\times p},\quad \hat{U}_p \in \mathbb{R}^{n\times p} \qquad \rightarrow$$

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

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 == rank-l 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 :

ϵ-rank = number k 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 (λ_i, u_i), where

$$\boxed{\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n}$$

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

- 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: $\mathrm{Tr}\,(P) = \mu_{[a,b]}$ Goal now: approximate : $\mathrm{Tr}\,(P)$

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 ψ

➤ Then use statistical estimator for approximating $\mathrm{Tr}\,(\psi(A))$

➤ 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: δ is the Dirac δ-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

- Survey paper: *[Lin-Lin, YS, Chao Yang], SIAM review, 2016.*

The Kernel Polynomial Method

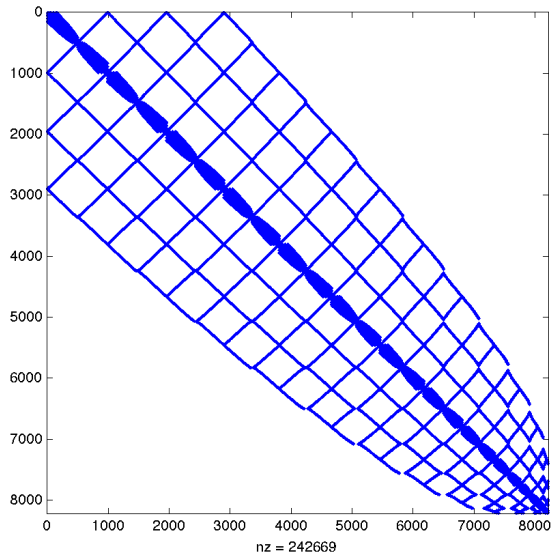
- Used by Chemists to calculate the DOS – see Silver and Röder'94 , Wang '94, Drabold-Sankey'93, + others
- Basic idea: expand DOS into Chebyshev polynomials
- Use trace estimators [discovered independently] to get traces needed in calculations
- Assume change of variable done so eigenvalues lie in $[-1,\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

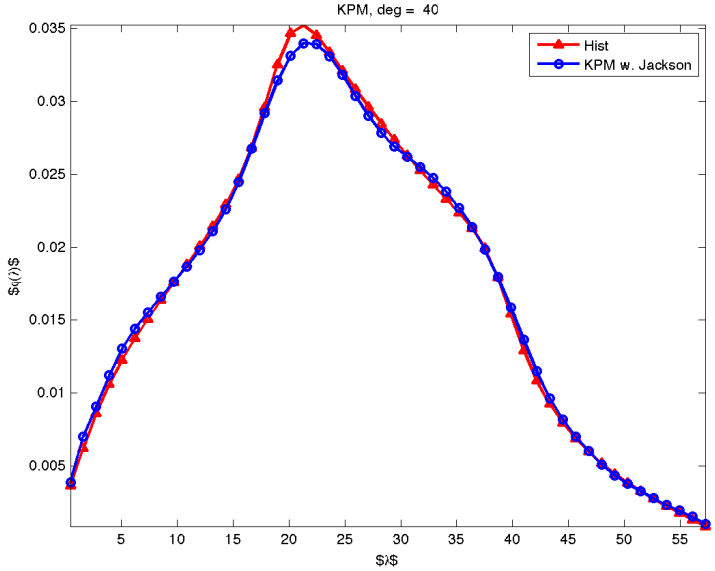
$$\hat{\phi}(t) = \sqrt{1-t^2}\phi(t) = \sqrt{1-t^2} \times \frac{1}{n}\sum_{j=1}^{n} \delta(t-\lambda_j).$$

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.
```





Integrating to get eigenvalue counts

- 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

$$\mu_{[a,b]} \approx \sum_{k=0}^{m} \mu_k \int_a^b \frac{T_k(t)}{\sqrt{1-t^2}} dt$$

- 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:

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

- Formula exact when p is a polynomial of degree $\leq 2m+1$

See: *Golub & Meurant '93, and also Gautschi'81, Golub and Welsch '69.*

- Consider now $\int p(t)dt =< p, 1 >=$ (Stieljes) integral $\equiv$

$$(p(A)v, v) = \sum \beta_i^2 p(\lambda_i) \equiv < \phi_v, p >$$

where $v = \sum \beta_i u_i$ = eigen -expansion of v, $\phi_v = \sum \beta_i^2 \delta_{\lambda_i}$

- Note: Ideal case $\beta_i = 1/\sqrt{n}$ yieds $\phi_v \equiv \phi$
- Then $\langle \phi_v, p \rangle \approx \sum a_i p(\theta_i) = \sum a_i \langle \delta_{\theta_i}, p \rangle \rightarrow$

$$\phi_v \approx \sum a_i \delta_{\theta_i}$$

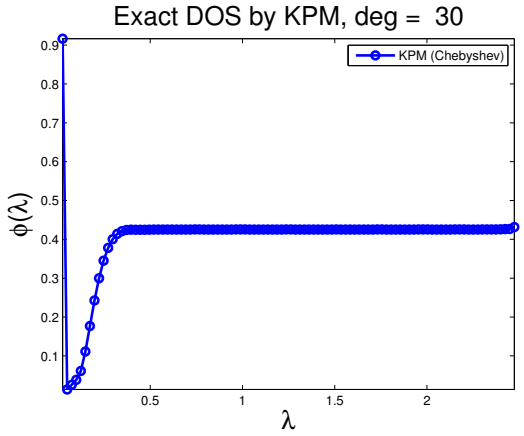
- Statistically produce choice $\boxed{\beta_i \equiv 1/\sqrt{n}, \forall i}$, average results over several vectors v with $\|v\|_2 = 1$.

Back to estimating the rank: Threshold selection

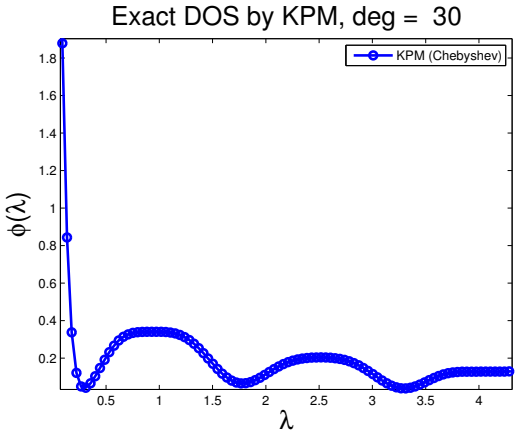
➤ Recall: numerical rank = # sing. values $\geq \epsilon$

Q: How to select ϵ?

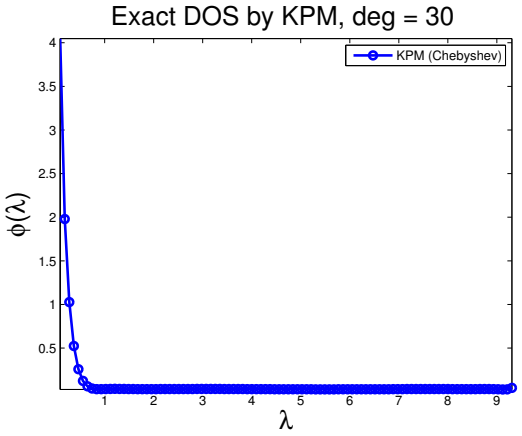
A: Obtain it from the DOS function



(A)



(B)



(C)

Exact DOS plots for three different types of matrices.

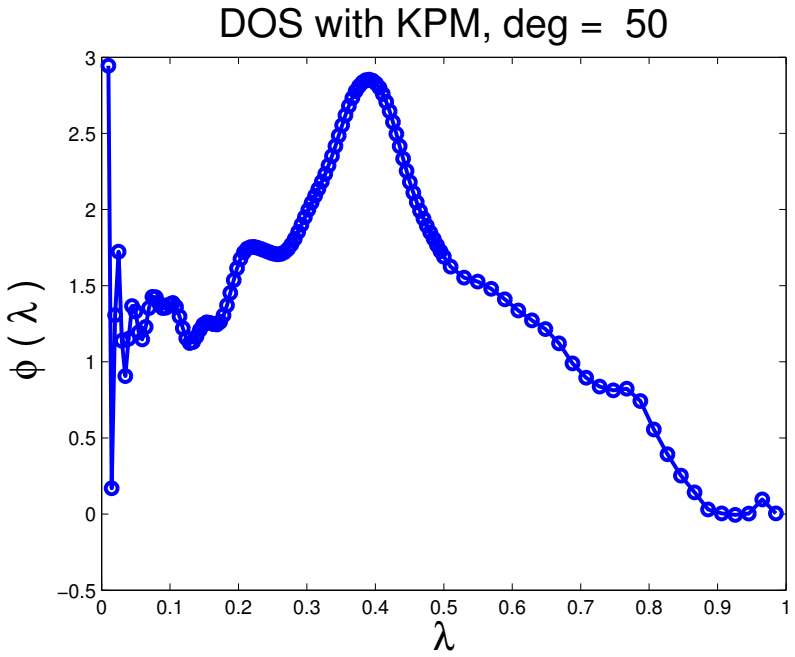
- To find: point immediatly following the initial sharp drop observed.
- Simple idea: use derivative of DOS function ϕ
- For an $n \times n$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$:

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

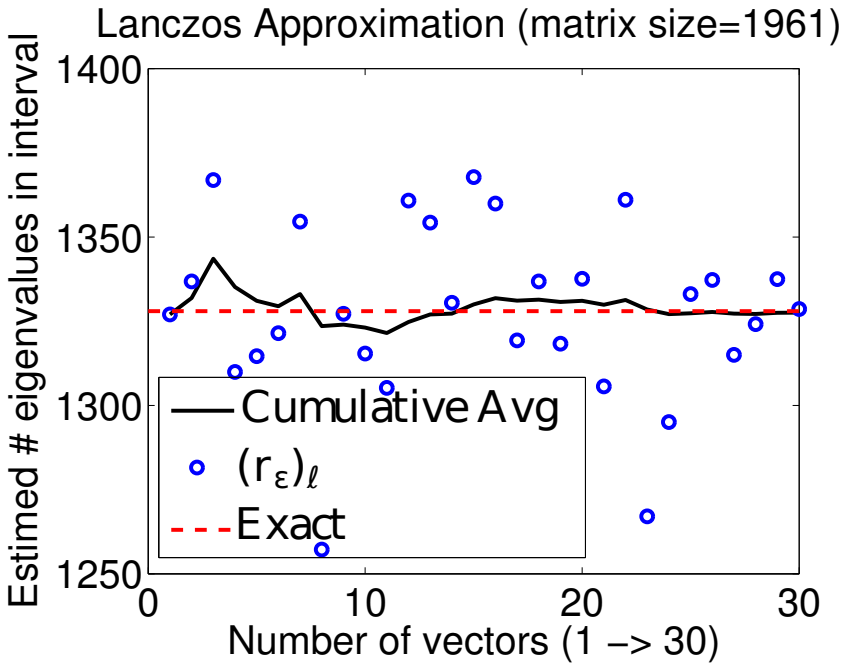
- In practice replace by

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

Experiments



(A)



(B)

(A) The DOS found by KPM.

(B) Approximate rank estimation by The Lanczos method for the example `netz4504`*.*

Approximate Rank Estimation of various matrices

lpi_ceria3d (linear programming)	$\mathbf{3,576}$
S80PI_n1 (model reduction prbm.)	$\mathbf{4,028}$
ukerbe1 (2D finite elem. prbm.)	$\mathbf{5,981}$
Erdos992 (collaboration network)	$\mathbf{6,100}$
Geom (computl. geometry)	$\mathbf{7,343}$
California (web search)	$\mathbf{9,664}$
C-40 (non-linear optimization)	$\mathbf{9,941}$

Matrices	Threshold ϵ	Eigencount above ϵ	M=100, n_v=30	
			r_ϵ -KPM	r_ϵ -Lanczos
lpi_ceria3d	28.19	78	78.69	78.74
S80PI_n1	1.76	2157	2154.04	2156.48
ukerbe1	0.169	4030	4030.84	4031.39
Erdos992	3.96	716	711.20	708.00
Geom	90	240	325.30	240.042
California	0.02	1646	5600.78	1646.66
C-40	48160.4	53	57.16	52.05

➤ Details: *S. Ubaru, Y. S. and A. Seghouane, "Fast Estimation of Approximate Matrix Ranks Using Spectral Densities," in Neural Computation, vol. 29, 2017.*

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]