Eigenvalue Problems Computation and Applications

Che-Rung Lee

cherung@gmail.com

National Tsing Hua University

Outline

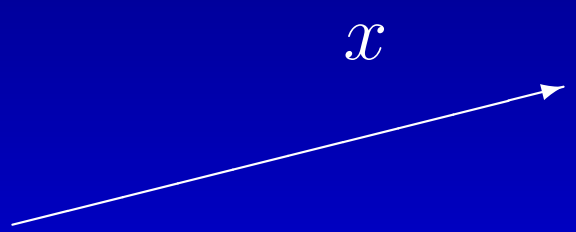
- Definitions
- Applications
  - Google’s pagerank.
  - Latent semantic indexing.
  - Point set segmentation.
- Computation
  - Power method.
  - Shift-invert enhancement.
  - Subspace methods.
  - Residual Arnoldi method.
- Conclusion

Eigenvalue and Eigenvector

- For a given $n \times n$ matrix A, if a scalar λ and a nonzero vector x satisfies $Ax = \lambda x$, we say (λ, x) is an eigenpair of A,
  - λ is called an eigenvalue;
  - x is called an eigenvector.

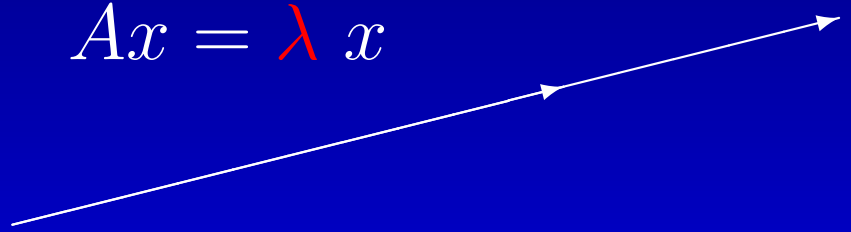
Eigenvalue and Eigenvector

- For a given $n \times n$ matrix A, if a scalar λ and a nonzero vector x satisfies $Ax = \lambda x$, we say (λ, x) is an eigenpair of A,
  - λ is called an eigenvalue;
  - x is called an eigenvector.



Eigenvalue and Eigenvector

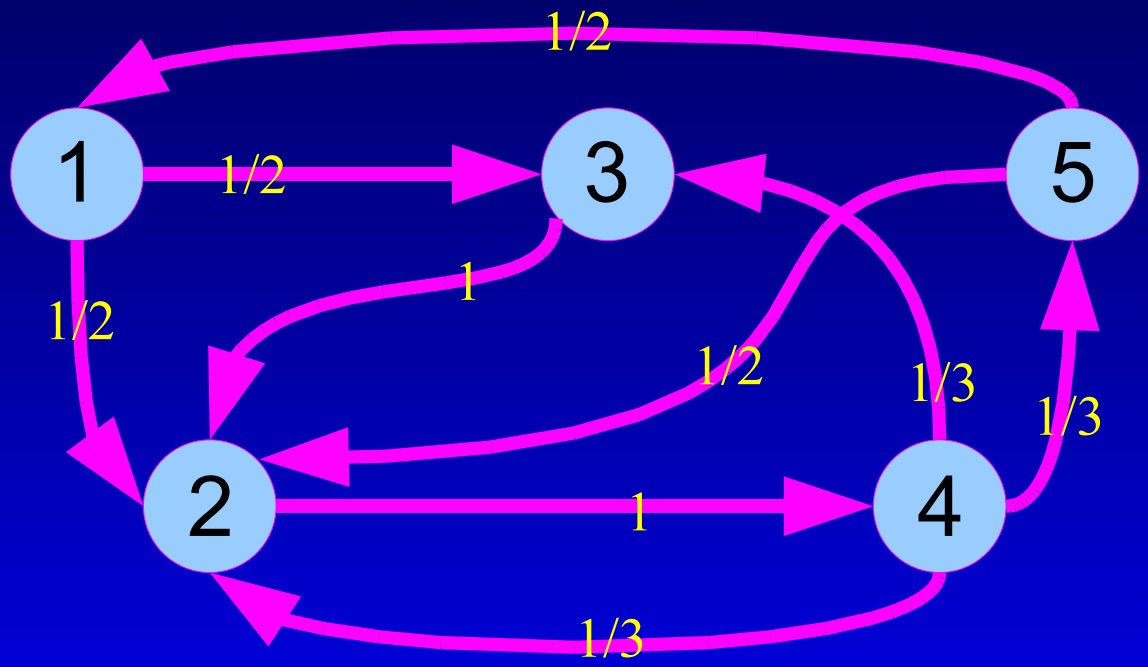
- For a given $n \times n$ matrix A, if a scalar λ and a nonzero vector x satisfies $Ax = \lambda x$, we say (λ, x) is an eigenpair of A,
  - λ is called an eigenvalue;
  - x is called an eigenvector.



Applications

Google's Pagerank

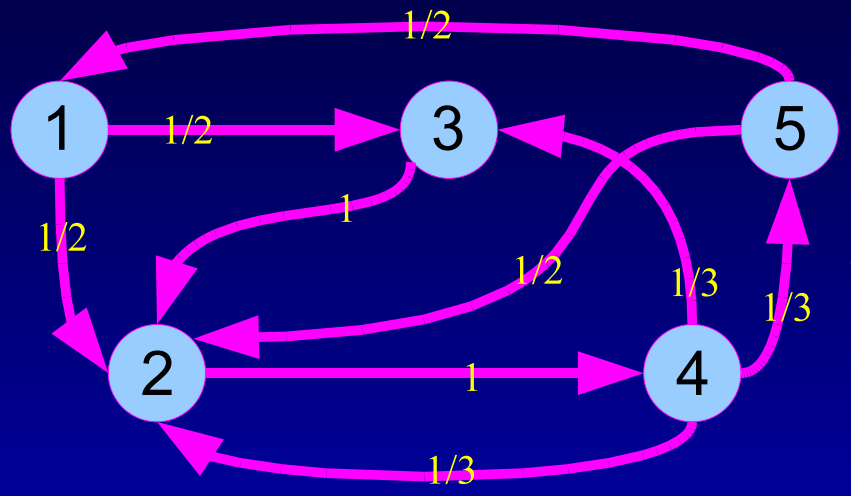
- Rank web pages by visiting probability.[a]
- Users' browsing behavior is modeled by random walk.



[a] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Manuscript in progress.

How to Compute?

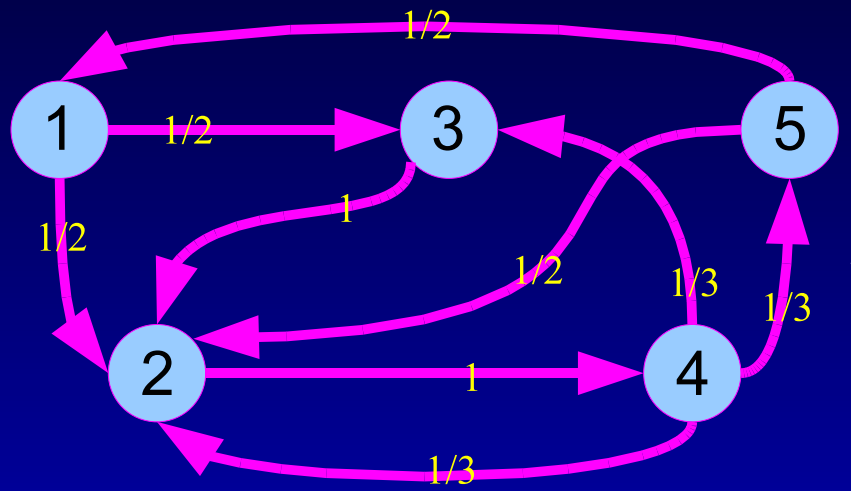
- The row-stochastic matrix P.



$$P = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 \\ 1/2 & 1/2 & 0 & 0 & 0 \end{bmatrix}$$

How to Compute?

- The row-stochastic matrix P.

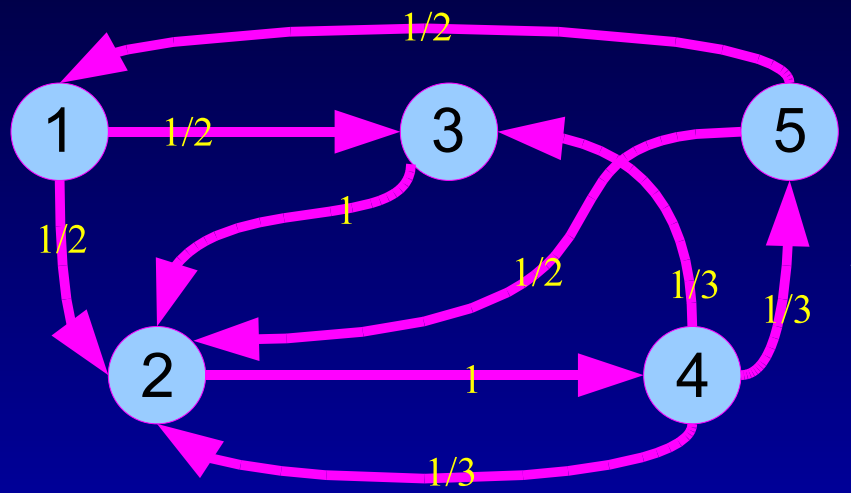


$$P = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 \\ 1/2 & 1/2 & 0 & 0 & 0 \end{bmatrix}$$

- The largest eigenvalue, λ_1, of P^T is 1; the corresponding eigenvector, x_1, gives the ranks.

How to Compute?

- The row-stochastic matrix P.



$$P = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 \\ 1/2 & 1/2 & 0 & 0 & 0 \end{bmatrix}$$

- The largest eigenvalue, λ_1, of P^T is 1; the corresponding eigenvector, x_1, gives the ranks.
- Markov chain: x_1 is the stationary distribution.
  - Many, many, many applications.

Semantic Query

Term \ Doc	D1	D2	D3	D4	D5
apple	53	65	0	30	1
computer	10	20	40	43	0
imac	30	10	25	52	70

- A query on "computer" returns D1, D2, D3, D4.
- A query on "apple" returns D1, D2 and D4. [a] [b]

[a] S. Deerwester, Susan Dumais, G. W. Furnas, T. K. Landauer, R. Harshman (1990). "Indexing by Latent Semantic Analysis". J of the ASIS 41 (6): 391V407.

[b] Example is modified from Dianne O'Leary's talk in MMDS 2006

Latent Semantic Indexing

- Term-Doc matrix $A = \begin{pmatrix} 53 & 65 & 0 & 30 & 1 \\ 10 & 20 & 40 & 43 & 0 \\ 30 & 10 & 25 & 52 & 70 \end{pmatrix}$.
- Low rank appr $\hat{A} = \begin{pmatrix} 49 & 65 & 7 & 34 & -5 \\ 23 & 22 & 14 & 30 & 21 \\ 25 & 9 & 34 & 57 & 63 \end{pmatrix}$.

Latent Semantic Indexing

- Term-Doc matrix $A = \begin{pmatrix} 53 & 65 & 0 & 30 & 1 \\ 10 & 20 & 40 & 43 & 0 \\ 30 & 10 & 25 & 52 & 70 \end{pmatrix}$.
- Low rank appr $\hat{A} = \begin{pmatrix} 49 & 65 & 7 & 34 & -5 \\ 23 & 22 & 14 & 30 & 21 \\ 25 & 9 & 34 & 57 & 63 \end{pmatrix}$.
- Using $\hat{A}$ instead of A.
  - Now the query on "computer" returns all docs.
  - The query on "apple" returns D1, D2, D4.

Low Rank Approximation

- Singular value decomposition (SVD)

$$A = U\Sigma V^T = U\mathrm{diag}(\sigma_1, \cdots, \sigma_n)V^T$$

- Low rank approximation

$$\hat{A} = U\mathrm{diag}(\sigma_1, \cdots, \sigma_p, 0, \cdots, 0)V^T$$

  - $\hat{A}$ is the best rank-p approximation.

Low Rank Approximation

- Singular value decomposition (SVD)

$$A = U\Sigma V^T = U\mathrm{diag}(\sigma_1, \cdots, \sigma_n)V^T$$

- Low rank approximation

$$\hat{A} = U\mathrm{diag}(\sigma_1, \cdots, \sigma_p, 0, \cdots, 0)V^T$$

  - $\hat{A}$ is the best rank-p approximation.

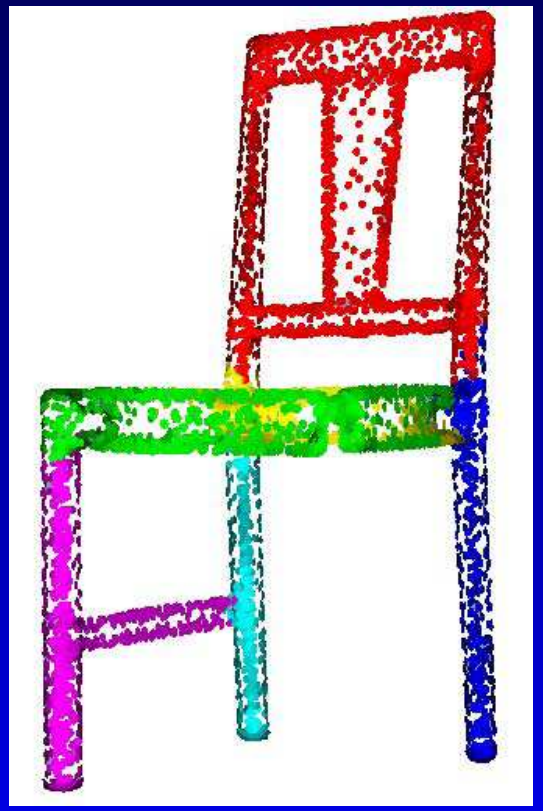
- $\sigma_1^2, \cdots, \sigma_p^2$ is the p largest eigenvalues of $A^T A$.

Applications of SVD

- Information retrieval, Text/speech summarization, Document organization
- Image processing and compression
- Pattern recognition, Eigenface, Watermarking
- Model reduction, Dimension reduction
- Digital signal processing
- Independent component analysis
- Total least square problem
- Bioinformatics
- Junk E-mail Filtering
- ...

Point Set Segmentation

- Partitioning a given point-sampled surface into distinct parts without explicit construction of a triangle mesh. [a]



[a] Ichitaro Yamazaki, etc Segmenting Point Sets, IEEE ICSMA 2006

Graph Partition

- Given a graph $G = (V, E)$, the *normalized cut* of a vertex partition $V = V_1 \cup V_2$ is

$$\mathrm{Ncut}(V_1, V_2) = \frac{\mathrm{cut}(V_1, V_2)}{\mathrm{assoc}(V_2, V)} + \frac{\mathrm{cut}(V_1, V_2)}{\mathrm{assoc}(V_1, V)}.$$

  - $\mathrm{cut}(V_1, V_2) = \sum_{v_1 \in V_1, v_2 \in V_2} W(v_1, v_2)$
  - $\mathrm{assoc}(V_1, V) = \sum_{v_1 \in V_1, v \in V} W(v_1, v)$

Graph Partition

- Given a graph $G = (V, E)$, the *normalized cut* of a vertex partition $V = V_1 \cup V_2$ is

$$\mathrm{Ncut}(V_1, V_2) = \frac{\mathrm{cut}(V_1, V_2)}{\mathrm{assoc}(V_2, V)} + \frac{\mathrm{cut}(V_1, V_2)}{\mathrm{assoc}(V_1, V)}.$$

  - $\mathrm{cut}(V_1, V_2) = \sum_{v_1 \in V_1, v_2 \in V_2} W(v_1, v_2)$
  - $\mathrm{assoc}(V_1, V) = \sum_{v_1 \in V_1, v \in V} W(v_1, v)$

- Discrete minimization of NCut is NP-complete.
- *Spectral graph partition* gives an approximate solution.

How to Compute?

- Laplacian matrix is $L = D - A$, where
  - D is a diagonal matrix whose elements are the degrees of vertices.
  - A is the adjacent matrix.

How to Compute?

- Laplacian matrix is $L = D - A$, where
  - D is a diagonal matrix whose elements are the degrees of vertices.
  - A is the adjacent matrix.
- The eigenvector of second smallest eigenvalue of L partitions the graph

$$\begin{cases} v_i \in V_1 & \text{if } x_2(i) > 0 \\ v_i \in V_2 & \text{if } x_2(i) < 0 \end{cases}$$

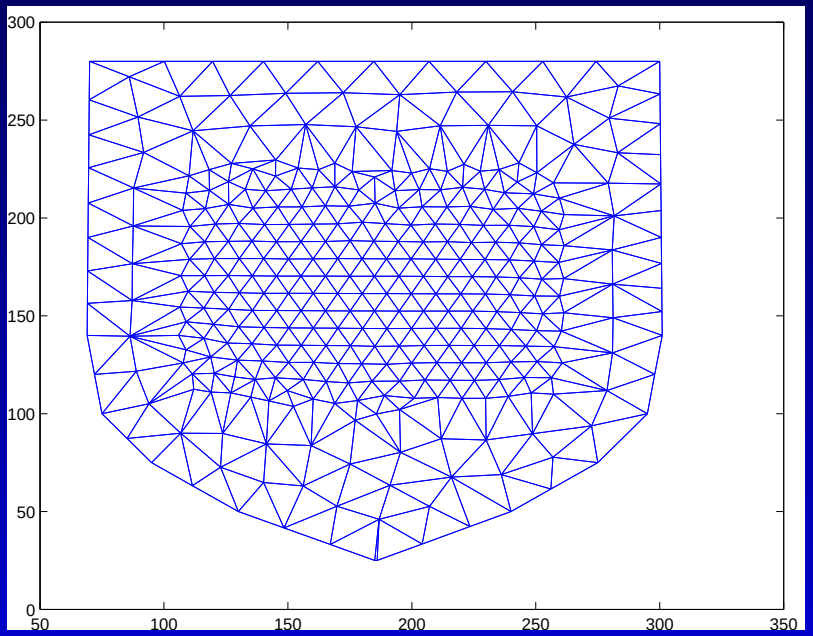
  - The second smallest eigenvalue of L is called *algebraic connectivity*.

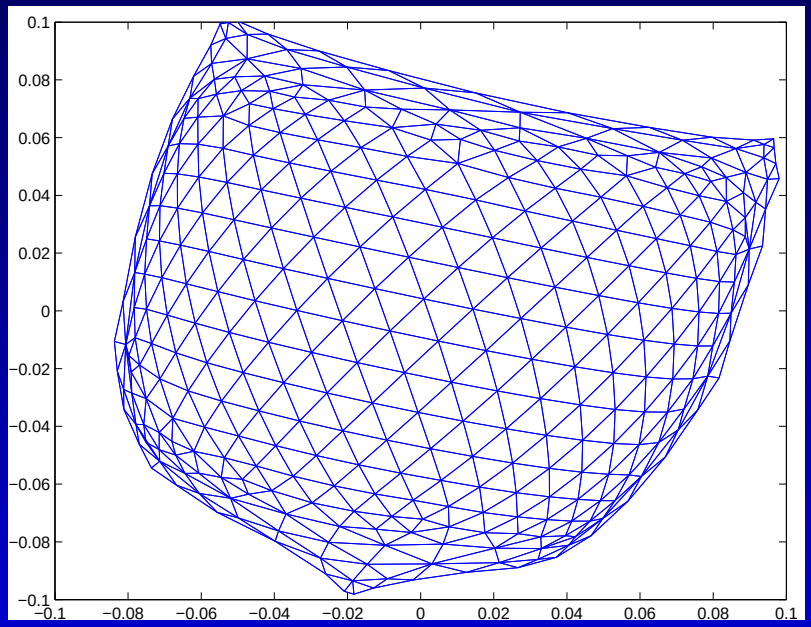
Applications of Spectral Graph

- Telephone network design
- Load balancing while minimizing communication
- Sparse matrix times vector multiplication
- VLSI layout
- Sparse Gaussian elimination
- Data mining and clustering
- Physical mapping of DNA
- Graph embedding
- Image segmentation
- ...

Spectral Graph Embedding

- Compute the eigenvectors of the second and third smallest nonzero eigenvalues, x_2, x_3.
- Use x_2, x_3 as the xy-coordinates of vertices.





a

[a] Dan Spielman. Spectral Graph Theory and its Applications.

Computation

The Power Method

- Algorithm:
  1. Given an initial vector p_0, $\|p_0\| = 1$.
  2. For $i = 1, 2, \ldots$ until converged
     (a) $p_i = Ap_{i-1}$
     (b) $p_i = p_i / \|p_i\|$ //normalization

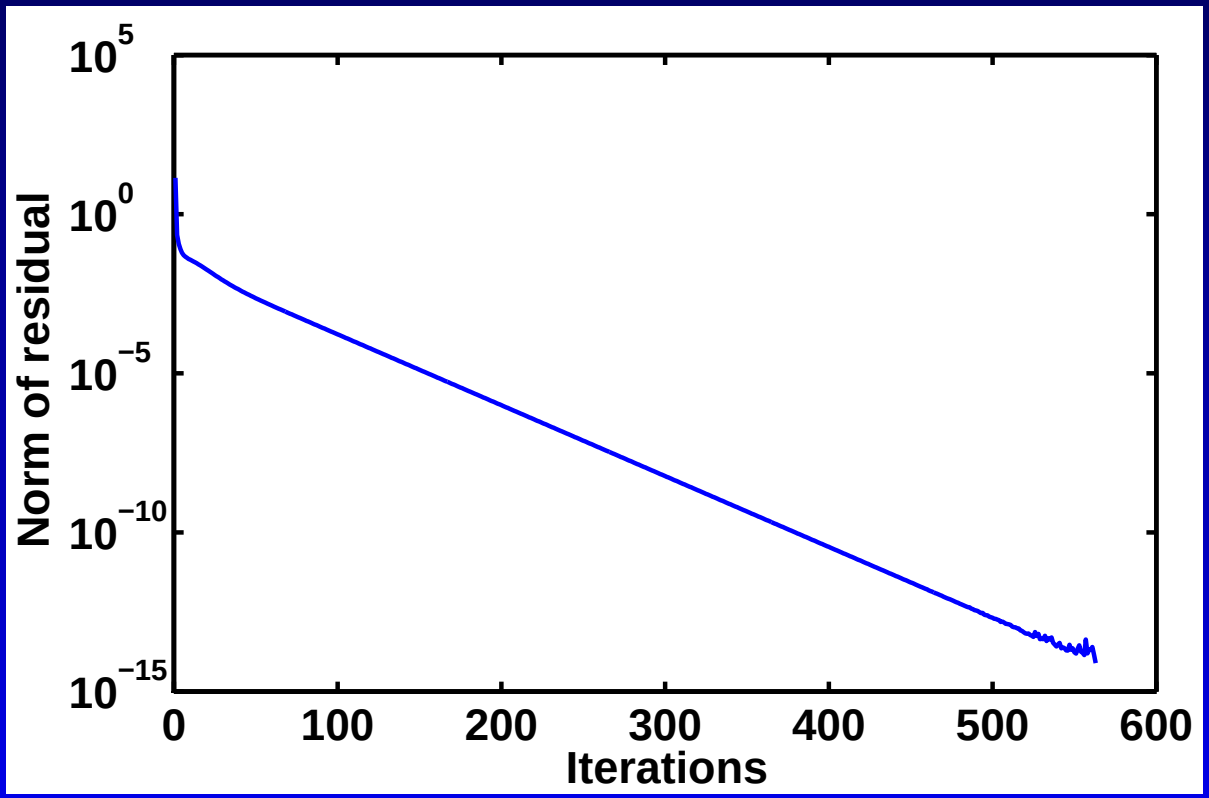
The Power Method

- Algorithm:
  1. Given an initial vector p_0, $\|p_0\| = 1$.
  2. For $i = 1, 2, \ldots$ until converged
     (a) $p_i = Ap_{i-1}$
     (b) $p_i = p_i / \|p_i\|$ //normalization

- Properties
  1. Only matrix-vector multiplication is needed.
  2. Vector p_k converges to the eigenvector of λ_1, assuming $|\lambda_1| \geq |\lambda_2| \geq \cdots |\lambda_n|$.
  3. The convergent rate is the ratio $\frac{|\lambda_2|}{|\lambda_1|}$

An Example

- A 100×100 matrix with eigenvalues $1, 0.95, \ldots, 0.95^{99}$.
- Converge to 10^{-14} in 550+ iterations.

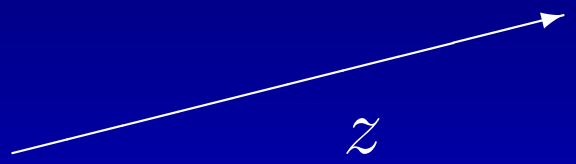


Residual

- Residual of an approximation:
  - Let (μ, z) be an approximation to an eigenpair of A. Its residual is $r = Az - \mu z$.

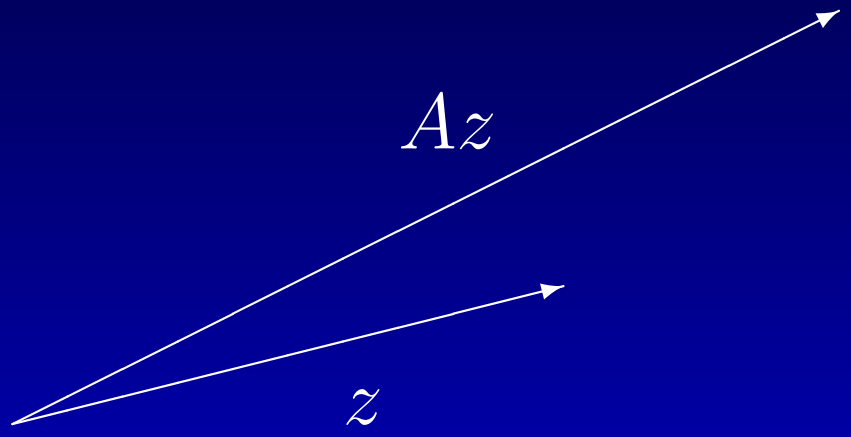
Residual

- Residual of an approximation:
  - Let (μ, z) be an approximation to an eigenpair of A. Its residual is $r = Az - \mu z$.



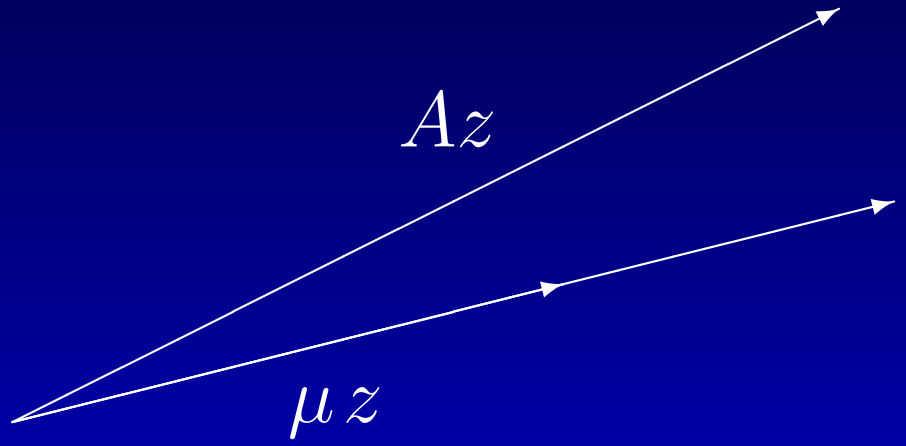
Residual

- Residual of an approximation:
  - Let (μ, z) be an approximation to an eigenpair of A. Its residual is $r = Az - \mu z$.



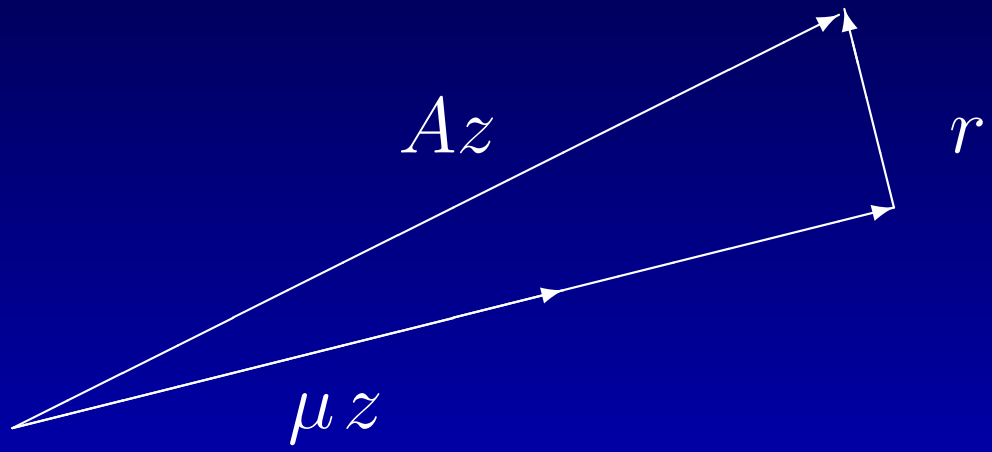
Residual

- Residual of an approximation:
  - Let (μ, z) be an approximation to an eigenpair of A. Its residual is $r = Az - \mu z$.



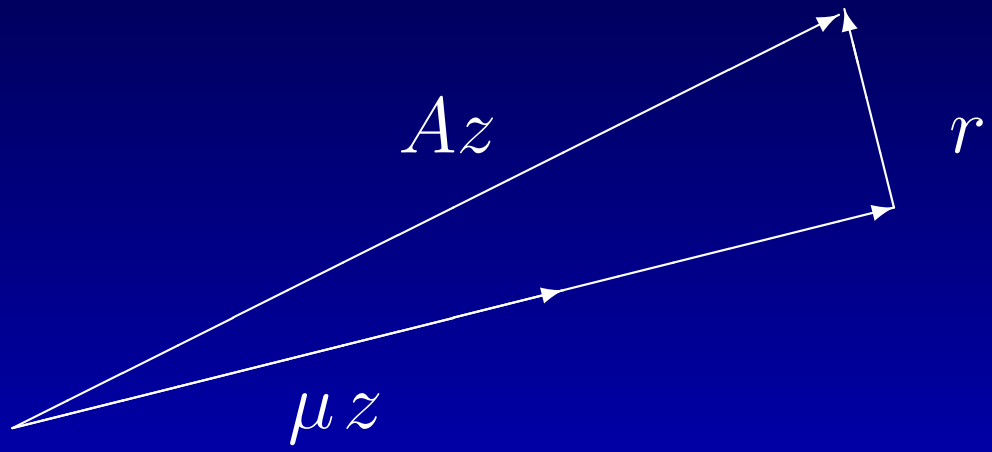
Residual

- Residual of an approximation:
  - Let (μ, z) be an approximation to an eigenpair of A. Its residual is $r = Az - \mu z$.



Residual

- Residual of an approximation:
  - Let (μ, z) be an approximation to an eigenpair of A. Its residual is $r = Az - \mu z$.



- Small residual implies (μ, z) is a good approximation. (in the sense of backward error.)

Speedup Methods

Speedup Methods

1. Shift-invert enhancement
   - Enlarge the ratio of λ_1 and λ_2.
   - Can be used to find other eigenpairs.

Speedup Methods

1. Shift-invert enhancement
   - Enlarge the ratio of λ_1 and λ_2.
   - Can be used to find other eigenpairs.

2. Subspace methods
   - Use linear combination of vectors to approximate the desired eigenvector.
   - Can be used to compute more than one eigenpairs.

Speedup Methods

1. Shift-invert enhancement
   - Enlarge the ratio of λ_1 and λ_2.
   - Can be used to find other eigenpairs.

2. Subspace methods
   - Use linear combination of vectors to approximate the desired eigenvector.
   - Can be used to compute more than one eigenpairs.

3. Subspace methods + shift-invert enhancement

Shift-invert Enhancement

- Use $C = (A - \sigma I)^{-1}$ in the power method. (Assume $\sigma \neq \lambda_i$ for all i.)
  - The eigenvectors of C are the same as A's.
  - The eigenvalues of C are $\frac{1}{\lambda_i - \sigma}$.

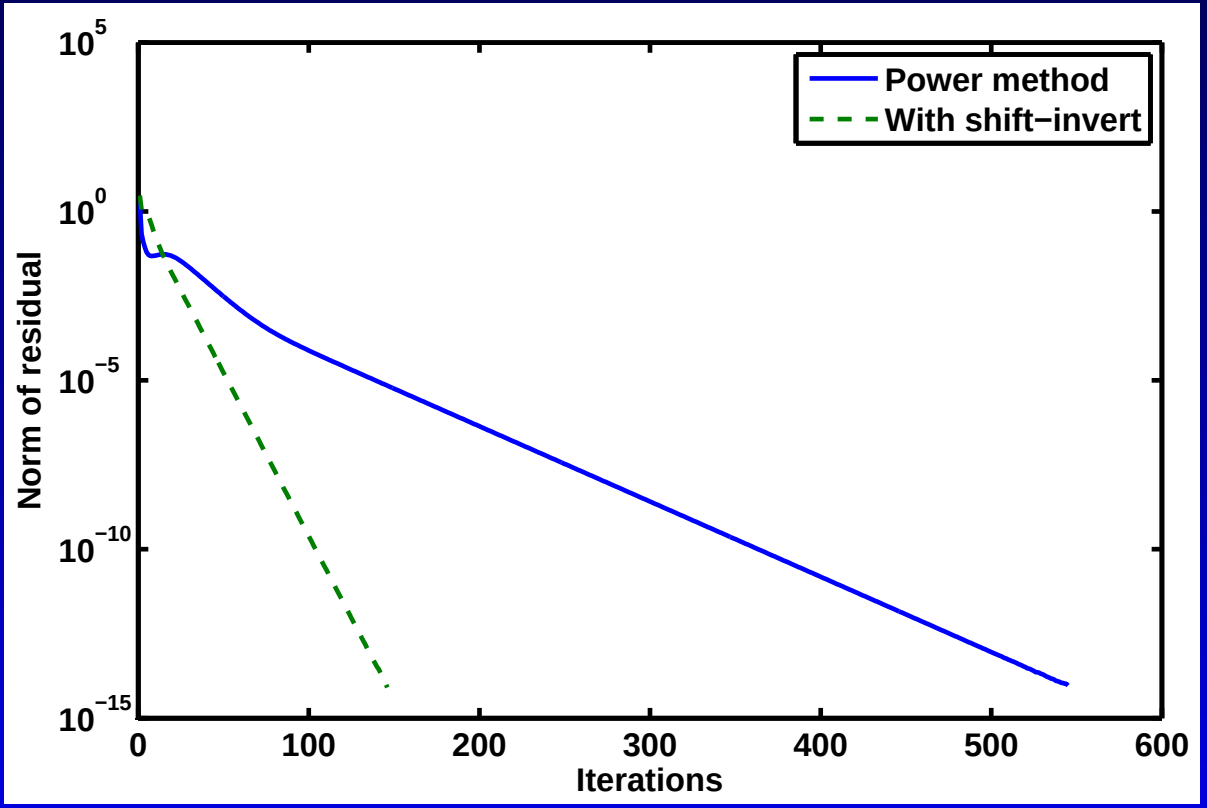
Shift-invert Enhancement

- Use $C = (A - \sigma I)^{-1}$ in the power method. (Assume $\sigma \neq \lambda_i$ for all i.)
  - The eigenvectors of C are the same as A's.
  - The eigenvalues of C are $\frac{1}{\lambda_i - \sigma}$.

- If σ is close enough to λ_1, the convergence rate of x_1 becomes $\frac{|\lambda_1 - \sigma|}{|\lambda_2 - \sigma|}$.

Example

- Use the previous example and set $\sigma = 1.2$.
  - Convergent rate from 0.95 to $\frac{|1-1.2|}{|0.95-1.2|} = 0.8$.



Subspace Methods

- The linear combination of eigenvector approximations can bring better approximations.

Subspace Methods

- The linear combination of eigenvector approximations can bring better approximations.

- Algorithm:
  1. Construct a subspace $\mathrm{span}\{u_1, u_2, \cdots, u_k\}$.
  2. Extract eigenvector approximations from the subspace.
  3. Test convergence.

Subspace Methods

- The linear combination of eigenvector approximations can bring better approximations.

- Algorithm:
  1. Construct a subspace $\mathrm{span}\{u_1, u_2, \cdots, u_k\}$.
  2. Extract eigenvector approximations from the subspace.
  3. Test convergence.

- Can compute more than one eigenvectors.

Krylov Subspace

- For a given unit vector u_1, the kth order Krylov subspace of matrix A and vector u_1 is

$$\mathcal{K}_k(A, u_1) = \text{span}\{u_1, Au_1, \cdots, A^{k-1}u_1\}.$$

Krylov Subspace

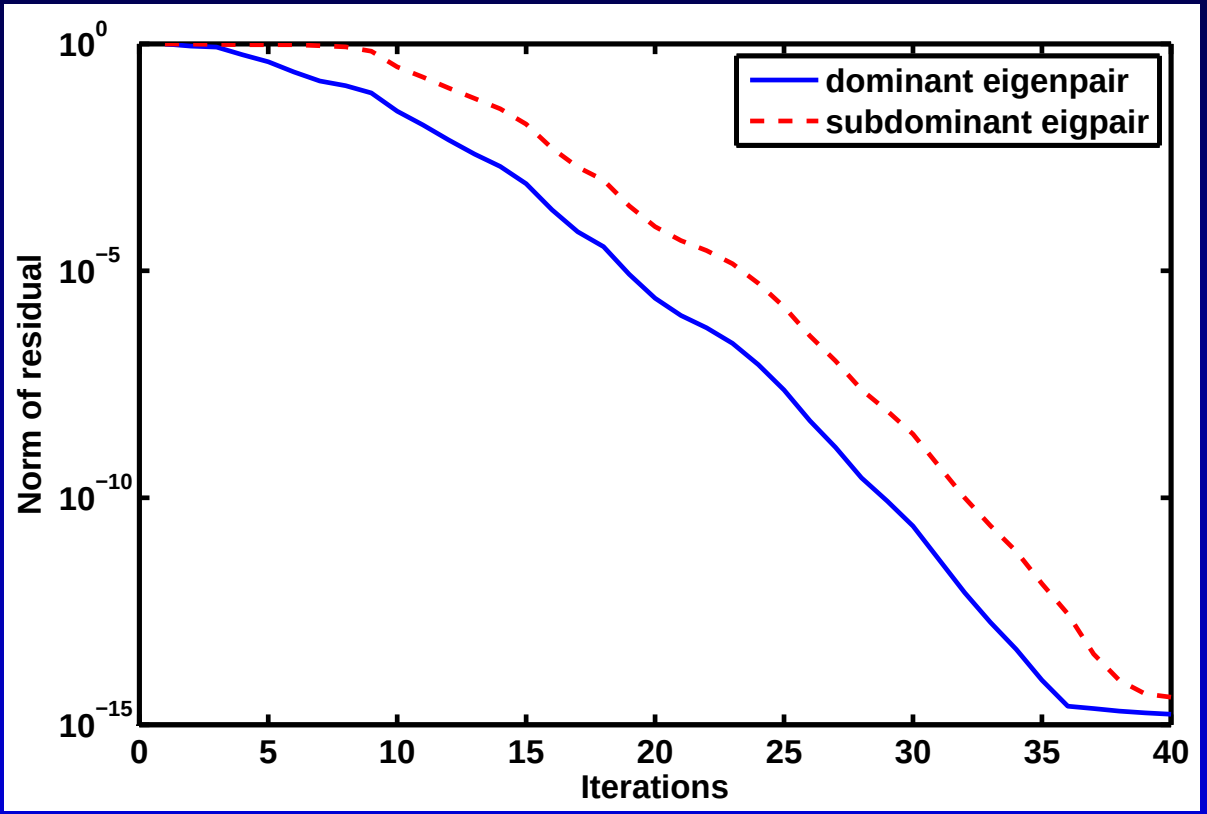
- For a given unit vector u_1, the kth order Krylov subspace of matrix A and vector u_1 is

$$\mathcal{K}_k(A, u_1) = \text{span}\{u_1, Au_1, \cdots, A^{k-1}u_1\}.$$

- Properties:
  1. Only matrix-vector multiplication is required.
  2. Usually contains good eigenvector approximations to those whose eigenvalues are on the peripheral of spectrum.
  3. Superlinear convergence.

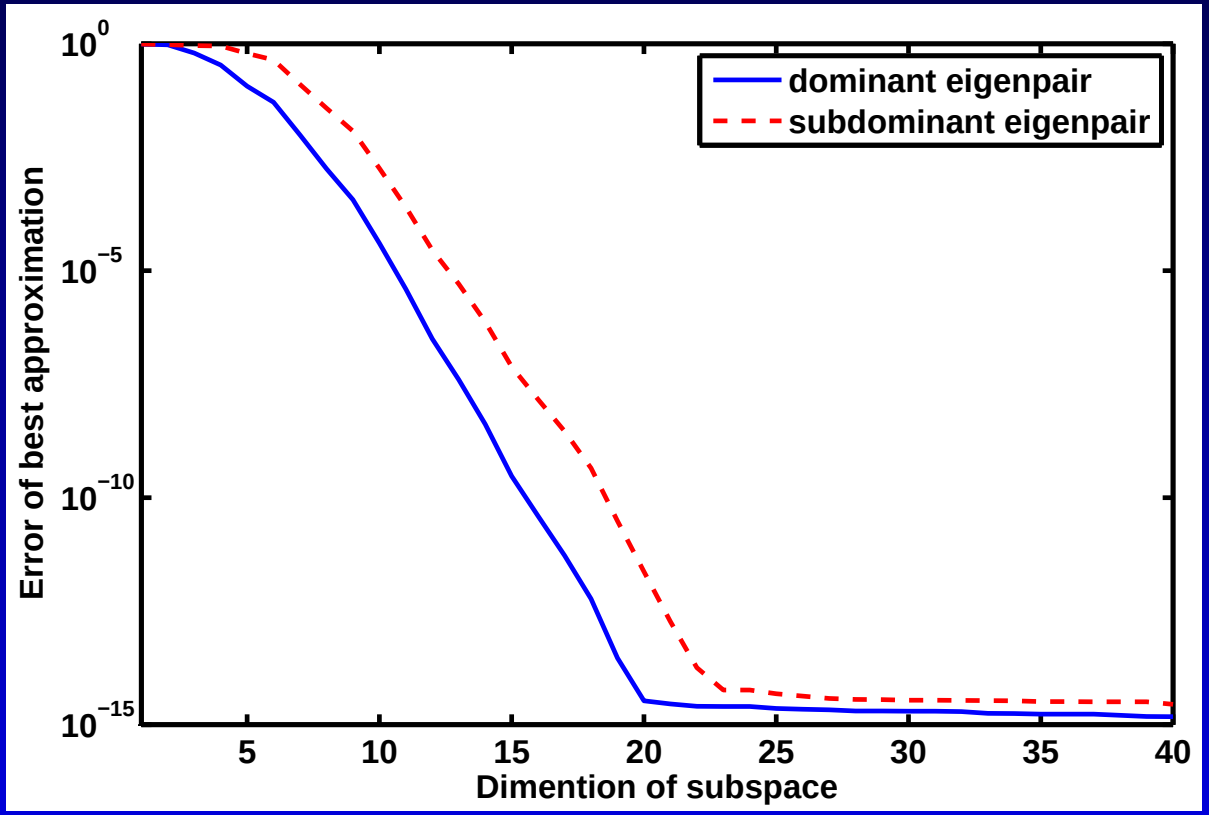
Krylov Subspace

- Two eigenpairs with largest eigenvalues converge to 10^{-14} in 40 iterations.



Krylov Subspace + Shift-invert

- Use subspace $\mathrm{span}\{u_1, Cu_1, \cdots, C^{k-1}u_1\}$ where $C = (A - \sigma I)^{-1}$, and $\sigma = 1.2$.



Problems

- For Krylov subspace method, $(A - \sigma I)u_{k+1} = u_k$ need be solved "exactly" in every iteration.

Problems

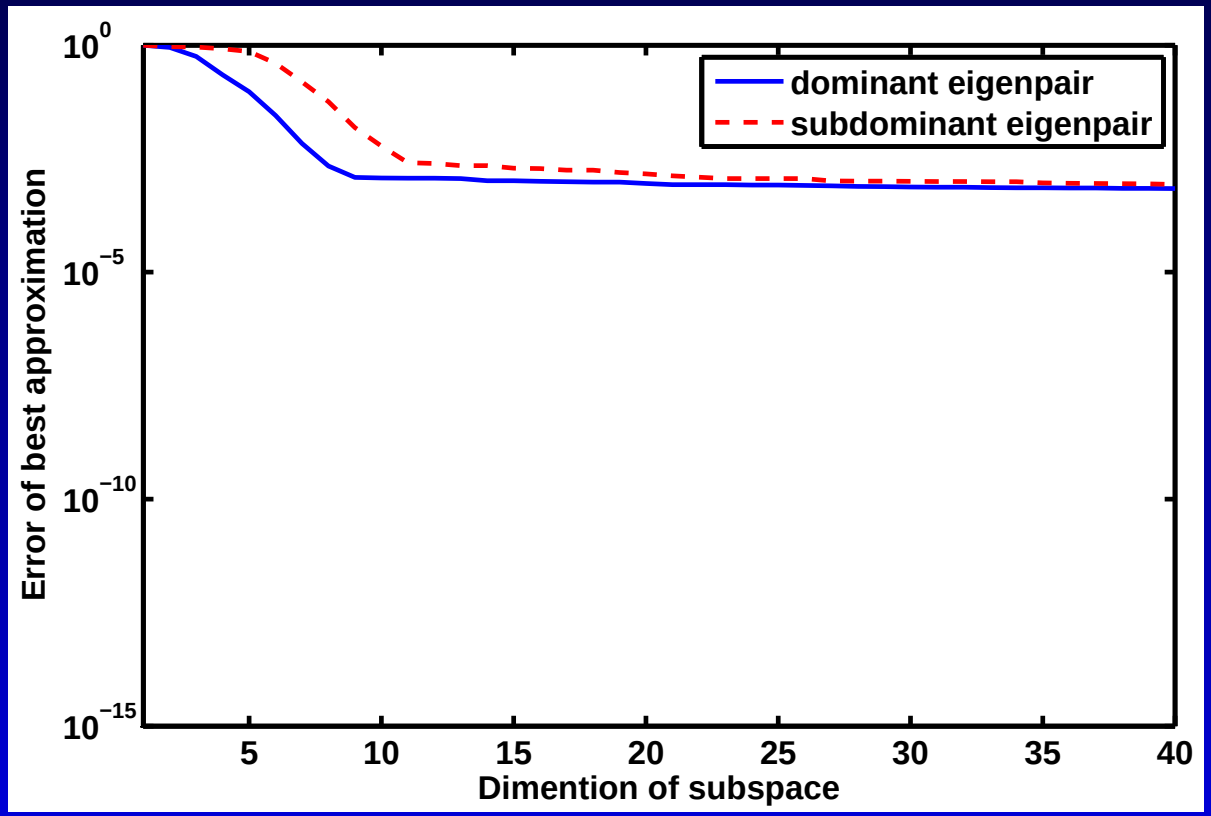
- For Krylov subspace method, $(A - \sigma I)u_{k+1} = u_k$ need be solved "exactly" in every iteration.

- When A is large, iterative methods for solving linear systems are often used.

  - Computation time $\propto$ desired precision.

With Inexact Shift-invert

- Linear systems are solved to the accuracy 10^{-3} in every iteration.



Residual Arnoldi Method

- Construct the subspace by $r_0, r_1, \cdots r_{k-1}$
  - r_i is the residual of an approximation in the ith iteration.

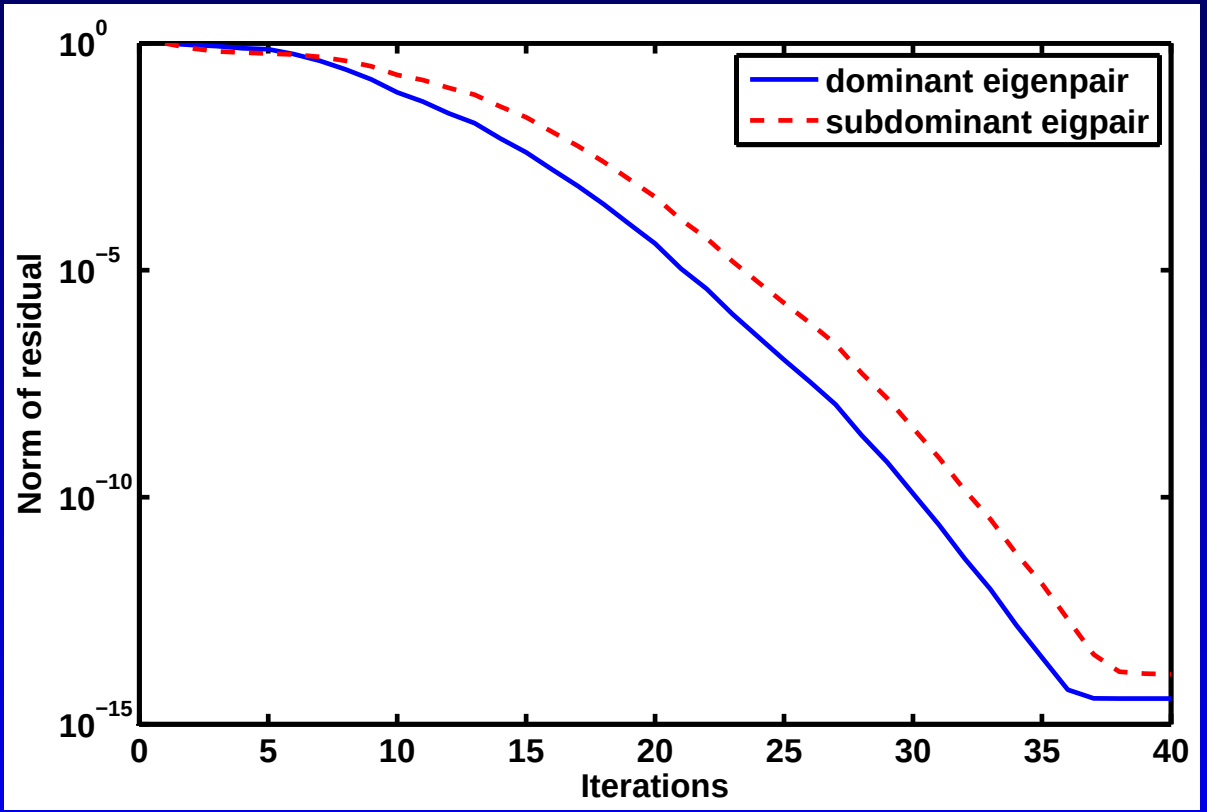
Residual Arnoldi Method

- Construct the subspace by $r_0, r_1, \cdots r_{k-1}$
  - r_i is the residual of an approximation in the ith iteration.

- Algorithm:
  1. Choose an approximation (μ_k, p_k) from the current subspace.
  2. Compute residual $r_k = Ap_k - \mu_k p_k$.
  3. Add r_k to the current subspace.

Without Shift-invert

- Use the residuals of the approximations to (λ_1, x_1).
- (λ_1, x_1) is called the target.



Residual Arnoldi + Shift-invert

- Residual Arnoldi method + Shift-invert
  1. Choose an approximation (μ_k, p_k) from the current subspace.
  2. Compute residual $r_k = Ap_k - \mu_k p_k$.
  3. Add $(A - \sigma I)^{-1} r_k$ to subspace.

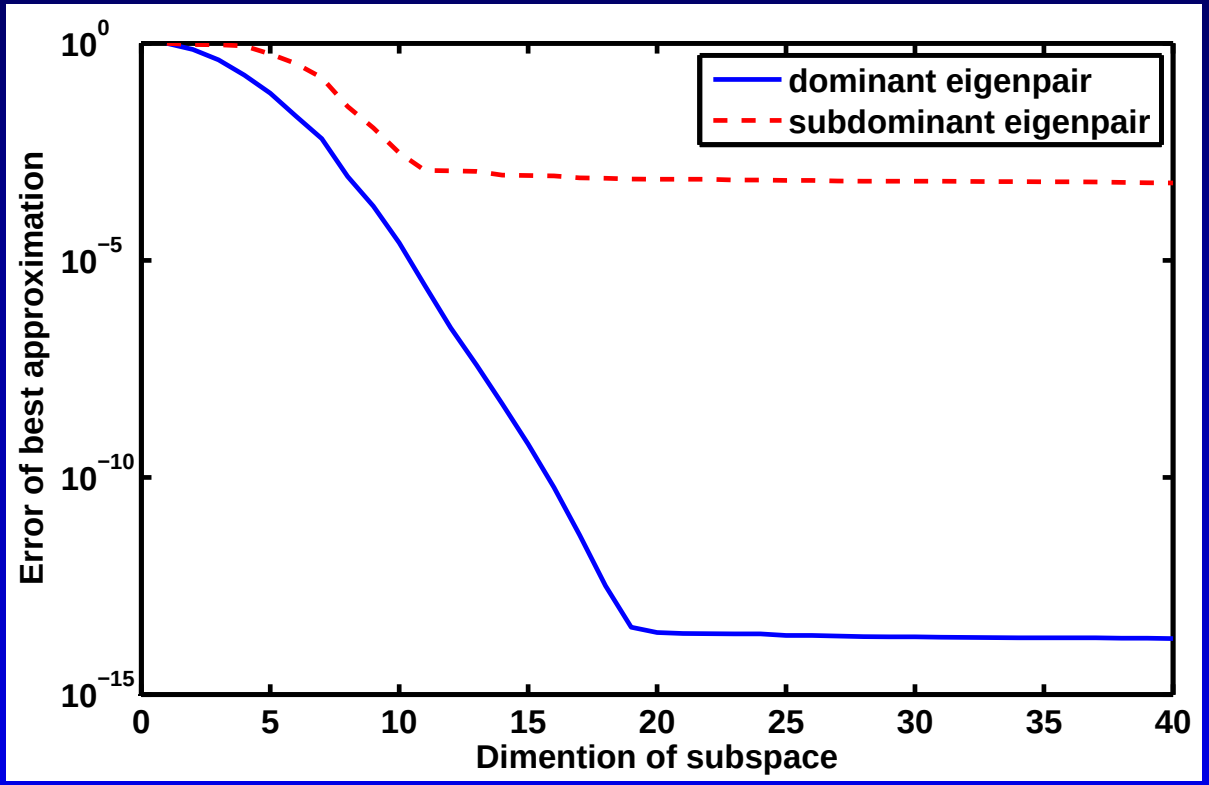
Residual Arnoldi + Shift-invert

- Residual Arnoldi method + Shift-invert
  1. Choose an approximation (μ_k, p_k) from the current subspace.
  2. Compute residual $r_k = Ap_k - \mu_k p_k$.
  3. Add $(A - \sigma I)^{-1} r_k$ to subspace.

- Properties of inexact shift-invert
  1. The approximations to the target can converge.
  2. Other approximations fail to converge.

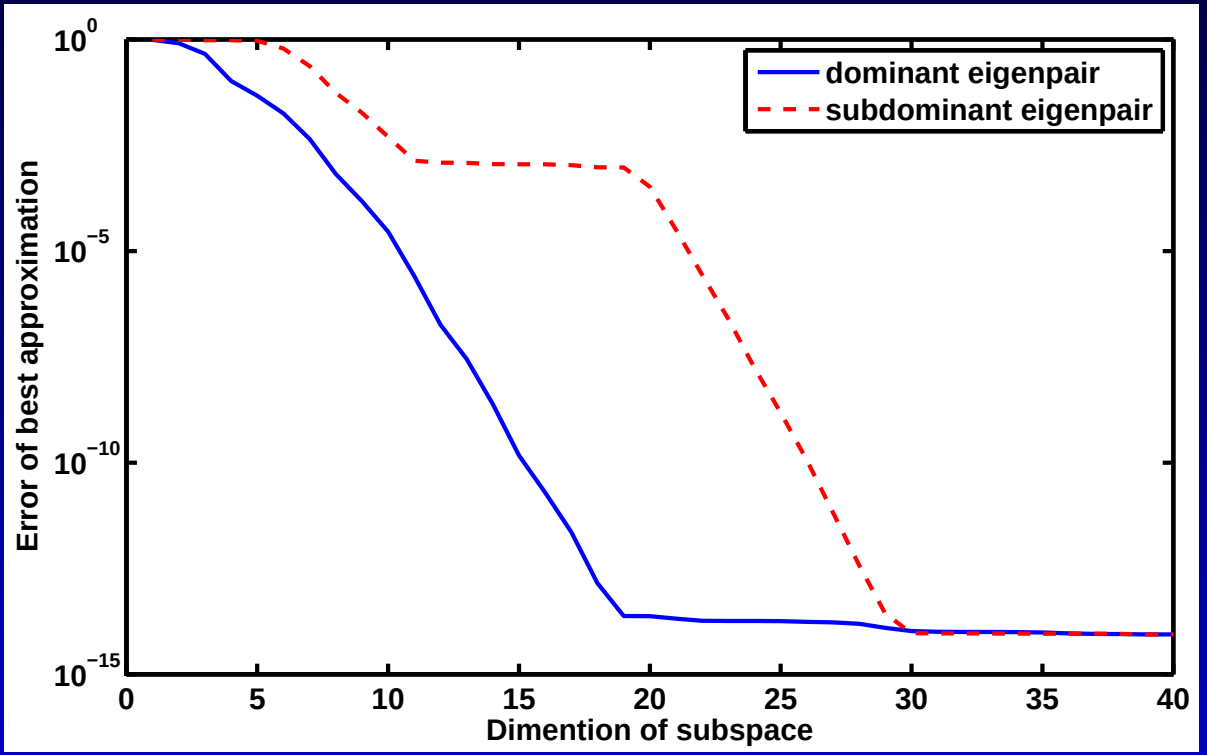
With Inexact Shift-invert

- Linear systems are solved to the accuracy 10^{-3} in every iteration.
- Target is (λ_1, x_1).



Change Target

- Change target to (λ_2, x_2) at iteration 20.



Performance Comparison

- For a 10000×10000 matrix, we want to find 6 smallest eigenvalues and their eigenvectors.

Performance Comparison

- For a 10000×10000 matrix, we want to find 6 smallest eigenvalues and their eigenvectors.
- Compare with the eigen-solver: ARPACK

Performance Comparison

- For a 10000×10000 matrix, we want to find 6 smallest eigenvalues and their eigenvectors.
- Compare with the eigen-solver: ARPACK
- Shift-invert enhancement are applied with shift 0 and linear solver GMRES.

Performance Comparison

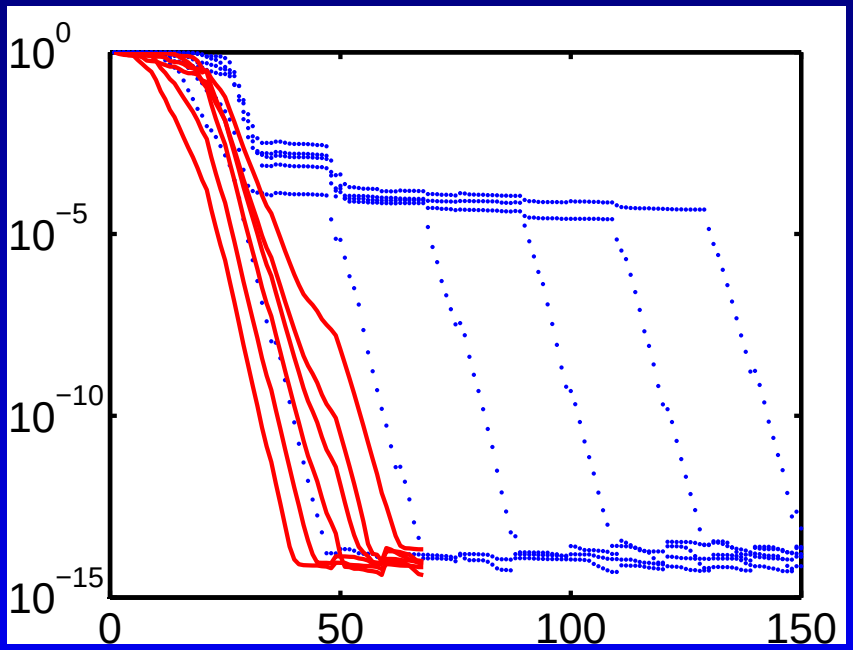
- For a 10000×10000 matrix, we want to find 6 smallest eigenvalues and their eigenvectors.
- Compare with the eigen-solver: ARPACK
- Shift-invert enhancement are applied with shift 0 and linear solver GMRES.
- Performance are measured by the total number of matrix-vector multiplication $\approx$

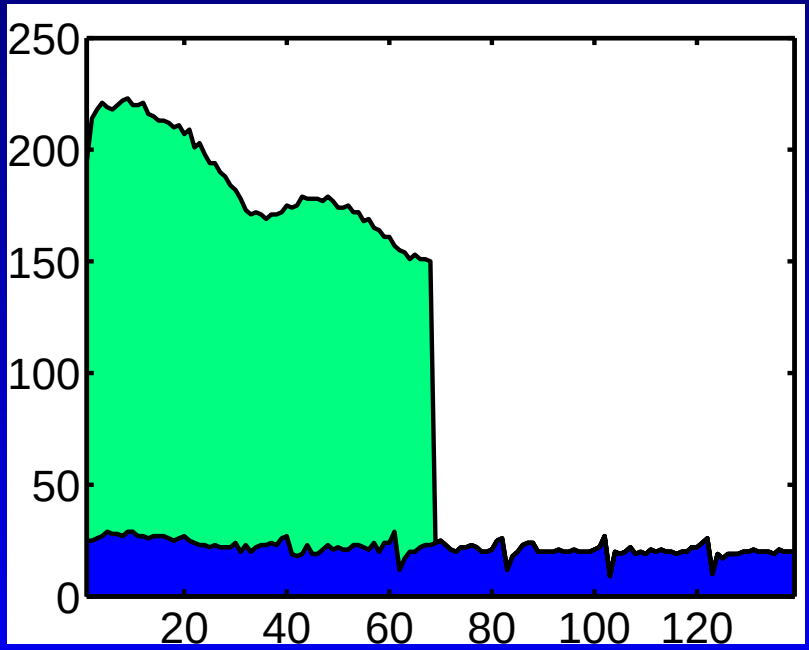
Performance Comparison

- For a 10000×10000 matrix, we want to find 6 smallest eigenvalues and their eigenvectors.
- Compare with the eigen-solver: ARPACK
- Shift-invert enhancement are applied with shift 0 and linear solver GMRES.
- Performance are measured by the total number of matrix-vector multiplication $\approx$ NO. of inner iterations $\times$ NO. of outer iterations.
  - Inner iteration: uses matrix-vector multiplication to solve linear systems.
  - Outer iteration: uses the results in inner iteration to solve eigenproblem.

Convergent Result

- Red lines: ARPACK
- Blue dots: residual Arnoldi method.
- Residual Arnoldi method is 2.25 times faster than ARPACK, and uses less than half matrix-vector multiplications.





Conclusion

- Eigenvalue problems are everywhere.

Conclusion

- Eigenvalue problems are everywhere.
- There is no single best algorithm for solving large eigenproblems.

Conclusion

- Eigenvalue problems are everywhere.
- There is no single best algorithm for solving large eigenproblems.
- Residual Arnoldi method is good for
  - computing interior or clustered eigenvalues.
  - applying shift-invert enhancement.
  - parallelization.