

연세대학교 통계 데이터 사이언스 학회 ESC 23-2 SUMMER WEEK5

Eigenvalue Problem for Sparse Matrices

[ESC 방학세션 1조] 김준엽 송시은 오동윤 최영준



Contents

1. Arnoldi Iteration
2. Krylov Subspaces
3. How Arnoldi Locates Eigenvalues
4. Lanczos Process
5. Ghost Eigenvalues





1. Arnoldi Iteration

Overview of Iterative Methods

Sparse Matrix

Sparse matrix : 행렬의 구성요소인 element들이 대부분 0인 행렬

데이터의 사이즈가 커지면 커질수록 (N뿐만 아니라 M또한 커지면)

Sparsity가 커질 가능성이 높음

예를 들어 유튜브 시청기록에 대한 행렬을 표현하고자 하면 $N(\text{유저 수}) \times M(\text{영상 수})$ 로 표현할 수 있는데,
M의 사이즈가 아주 크기 때문에 sparsity가 커질 가능성이 높은 것

이처럼 실제 데이터의 경우에는 sparse matrix를 마주할 가능성이 높음 -> sparse matrix에 특화된 processing하는 과정 필요



Overview of Iterative Methods

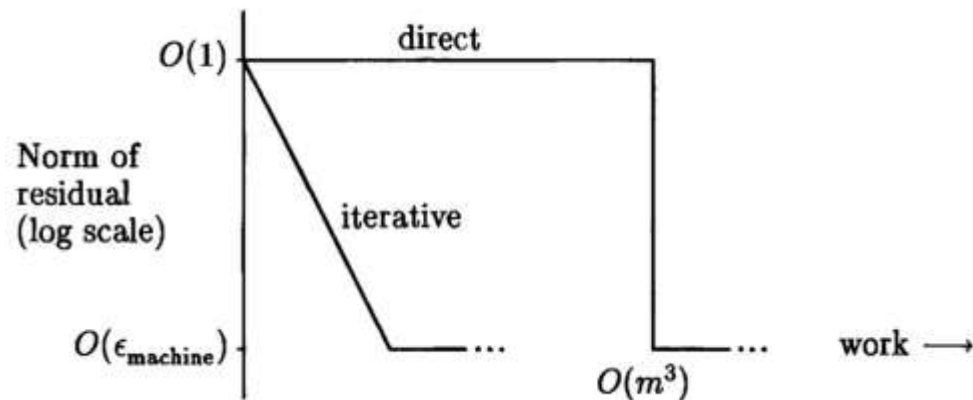
Why Iterate?

계산에 필요한 cost가 줄어들기 때문에! (m : 행렬의 차원)

- Non-iterative 알고리즘: $O(m^3)$
- iterative 알고리즘: $O(m^2)$

특히 sparse matrix의 경우, 차원은 크지만 non-zero 성분이 적으므로 적절한 iterative method를 사용해 cost를 크게 줄일 수 있다.

예를 들어, 차원은 $m = 10^5$ 이지만 0이 아닌 성분은 $v = 10$ 개인 행렬 A 를 생각해보자
행렬 A 와 벡터 x 의 곱을 구하기 위해 iterative 알고리즘을 사용하면
 $O(vm)$ 번의 계산을 통해 답을 구할 수 있다.



Arnoldi Iteration

The Arnoldi/Gram-Schmidt Analogy

sparse matrix의 eigenvalue를 구하기 위해 **Arnoldi Iteration**이 사용된다.

먼저, Householder transformation을 이용해 아래와 같이 upper Hessenberg form으로 나타낼 수 있다.

$$Q^T A Q = H, \quad \text{where } A \in \mathbb{R}^{n \times n}$$

Gram-Schmidt 방식으로 QR 분해($A=QR$)을 계산하는 것과,

Hessenberg reduction ($A=QHQ^*$)를 계산하는 방식은 매우 유사함. (Gram-schmidt의 장점 활용 가능 – can be stopped)

	$A = QR$	$A = QHQ^*$
orthogonal structuring	Householder	Householder
structured orthogonalization	Gram-Schmidt	Arnoldi

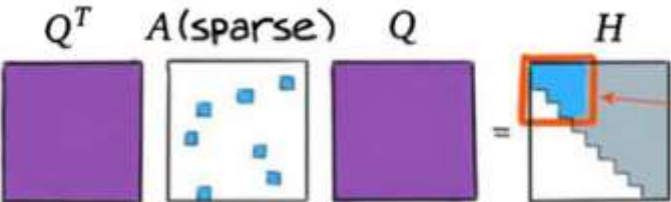


Arnoldi Iteration

Mechanics of the Arnoldi Iteration

dense matrix의 경우, H 의 eigenvalue는 A 의 eigenvalue와 같지만, 우리는 sparse matrix의 경우를 고려해야 함! (estimate)
 H 의 $k \times k$ 만큼의 블록을 떼어내 먼저 eigenvalue를 계산하고, 점차 k 를 늘려가며 A 의 eigenvalue를 계산할 수 있다.

Step 1:

$$Q^T A(\text{sparse}) Q = H$$


We can compute this $k \times k$ block quickly because A is sparse

Step 2:

$$\lambda \left(\begin{array}{|c|} \hline \text{blue block} \\ \hline \end{array} \right)$$

We can compute the eigenvalues of this $k \times k$ block quickly because k is small. Then we will use these to estimate the eigenvalues of A



Arnoldi Iteration

Mechanics of the Arnoldi Iteration

$Q^T A Q = H$ 식에서 Q 는 orthogonal하므로, $AQ = QH$ 로 식을 변형할 수 있다.

다음, Q 의 k 번째 열과 H 의 $k - 1$ 번째 열까지 알고 있다고 가정했을 때, 다음과 같이 고유값을 계산할 수 있다.

$$\begin{aligned} \text{for } i \leq k, \quad q_i^T A q_k &= h_{ik} \\ \rightarrow A q_k &= h_{1k} q_1 + \cdots + h_{kk} q_k + h_{k+1,k} q_{k+1} \end{aligned}$$

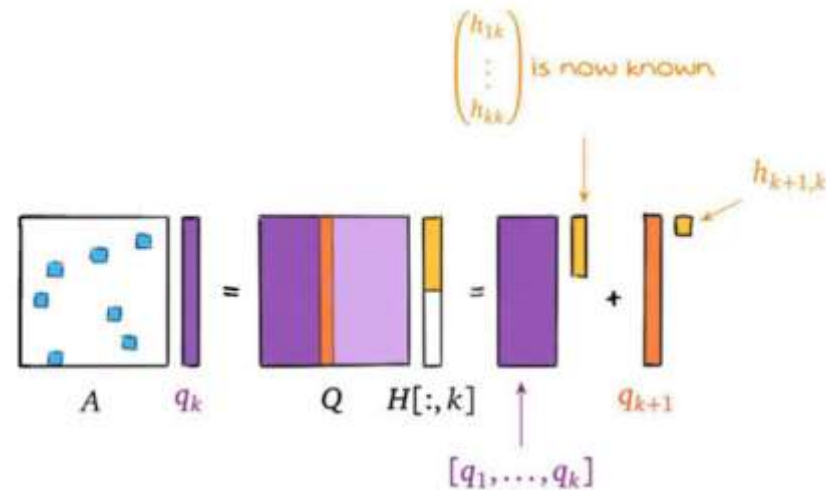
위 식을 변형시켜 아래와 같이 나타낼 수 있다.

$$r_k = A q_k - h_{1k} q_1 - \cdots - h_{kk} q_k = h_{k+1,k} q_{k+1}$$

$$\rightarrow h_{k+1,k} = \|r_k\|, \quad q_{k+1} = r_k / h_{k+1,k}$$

$h_{i,k+1}, q_{k+2}$ 도 같은 과정을 통해 얻을 수 있다.

(놈의 부호는 반대로 해도 상관없다)



Arnoldi Iteration

Mechanics of the Arnoldi Iteration

이처럼 랜덤 벡터 q_1 에서 시작해 반복함으로써 H 와 Q 의 k 개의 열을 구하는 방법을 Arnoldi process라고 한다.

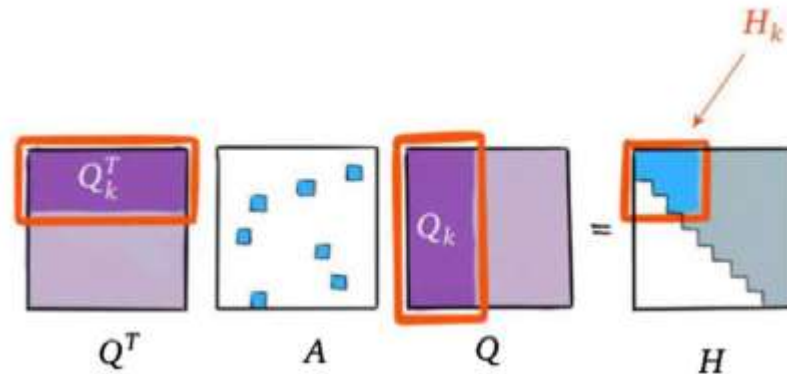
Arnoldi process를 통해 H_k 를 구하면, H_k 의 eigenvalue를 구하여 A 의 eigenvalue로 근사할 수 있다.

H_k 는 크기가 작으므로 eigenvalue를 빨리 구할 수 있다는 장점이 있다.

한편 A 의 eigenvalue는 n 개이고, H_k 의 eigenvalue는 k 개이므로($n \leq k$),

모든 eigenvalue를 추정할 수 없다는 한계가 있다.

단, k 의 크기가 커질수록 더 많은 값을 추정할 수 있다.



Arnoldi Iteration

Mechanics of the Arnoldi Iteration

Arnoldi iteration은 다음과 같은 프로세스로 요약할 수 있다.


1. last vector q_k 를 이용해 sequence 생성
2. A를 행렬곱
3. $\{q_1, q_2, \dots, q_k\}$ 에 대하여 orthogonal하게 만들기

Arnoldi recurrence relation

Suppose we already have q_1, \dots, q_k and $h_{:,j}$ for $j < k$. Then we can find q_{k+1} and $h_{:,k}$ as follows:

$$\begin{aligned}h_{ik} &= q_i^T A q_k \quad \text{for } i \leq k, \\r_k &= A q_k - \sum_{i=1}^k h_{ik} q_i, \\h_{k+1,k} &= \|r_k\|_2, \\q_{k+1} &= \frac{r_k}{h_{k+1,k}}.\end{aligned}$$





2. Krylov Subspaces

Krylov Subspaces

Krylov Sequence

행렬 A 와 벡터 b 가 주어졌다고 생각해보자.

- Krylov sequence: b, Ab, A^2b, A^3b, \dots , 의 벡터들의 집합

$$K_k = [b, Ab, \dots, A^{k-1}b]$$

- Krylov subspace: 위의 벡터들로 span한 공간

$$\mathcal{K}_k = \text{span}\{b, Ab, \dots, A^{k-1}b\}$$

Krylov subspace

The space spanned by all $A^i q_1$, $0 \leq i < k$, is called a **Krylov subspace**:

$$\mathcal{K}(A, q_1, k) = \text{span}\{q_1, Aq_1, A^2q_1, \dots, A^{k-1}q_1\}.$$

The matrix $K_k = [q_1, Aq_1, \dots, A^{k-1}q_1]$ is called a **Krylov matrix**.



Krylov Subspaces

Krylov Sequence

이러한 sequence는 $b, Ab, A(Ab), A(A(Ab)), \dots$ 와 같은 형태의 black box로 계산할 수 있다.

이때, Arnoldi Iteration의 벡터 $\{q_k\}$ 는 위와 같은 Krylov 부분공간의 기저를 형성한다.

또한 $\{q_k\}$ 는 정규직교벡터이므로, 이들은 정규직교기저가 된다.

즉 Arnoldi process는 연속적인 Krylov 부분공간의 정규직교기저로 이루어진 시스템이다.

$$\mathcal{K}_k = \text{span}\{b, Ab, \dots, A^{k-1}b\} = \text{span}\{q_1, q_2, \dots, q_k\}$$

- Krylov 의 사용 목적? M dimensional한 문제를 낮은 차원(krylov space)으로 풀려고

	$Ax = b$	$Ax = \lambda x$
$A = A^*$	CG	Lanczos
$A \neq A^*$	GMRES CGN BCG et al.	Arnoldi



Krylov Subspaces

QR Factorization of a Krylov Matrix

K_k 은 아래와 같은 reduced QR Factorization을 갖는다.

$$K_k = Q_k R_k$$

먼저 $Q^T K_k$ 에서 j 번째 열은 다음과 같다.

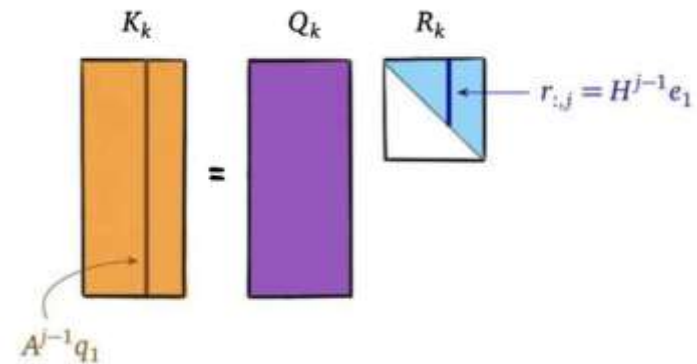
$$Q^T A^{j-1} q_1 = Q^T Q H^{j-1} Q^T q_1 = H^{j-1} e_1$$

그렇다면 R_k 의 원소 r_{ij} 를 위 식에서 구한 $H^{j-1}(i, 1)$ 로 고를 수 있다. ($H^0 = I$)

H 는 Hessenberg 행렬이기 때문에 R_k 는 upper triangular 행렬이다. 따라서 K_k 를 QR factorization 할 수 있다.

Arnoldi process에서는 K_k 의 열들이 A 의 dominant eigenvalue에 대한 근사치이므로

위의 행렬들은 일반적으로 ill-conditioned한 행렬들일 것이고, 따라서 이는 직관적이지만 unstable한 알고리즘이다.



Krylov Subspaces

Projection onto Krylov Subspaces

이때 Hessenberg matrix H_k 은 다음과 같은 projection이다.

$$H_k = Q_k^T A Q_k$$

이러한 projection은 Rayleigh-Ritz 방법으로도 알려져 있다.

또한 H_k 이 A 의 projection이므로 A 의 eigenvalue와 H_k 의 eigenvalue는 서로 연관이 있다.

이때 아래의 k 개의 숫자들이 Arnoldi eigenvalue estimates (at step k), 또는 Ritz values (with respect to K_k)라고 불린다.

$$\{\theta_j\} = \{\text{eigenvalues of } H_k\}$$



Krylov Subspaces

Arnoldi and Polynomial Approximation

Krylov subspace를 이용하는 이유 중 하나는 A 와 powers와의 connection을 제공할 수 있다는 점이다.
matrix의 polynomial은 다음과 같다.

$$\begin{aligned} f(z) &= f_0 + f_1 z + \cdots + f_k z^k \\ \rightarrow f(A) &= f_0 I + f_1 A + \cdots + f_k A^k \end{aligned}$$

만약 A 가 대각화가 가능하다면 $f(A) = Xf(\Lambda)X^{-1}$ 이 성립한다.

위처럼 polynomial 형태로 만든 후에 행렬 A 에 대한 특성방정식을 세울 수 있다.

$$\rightarrow P_A(z) = \det(zI - A)$$

그렇다면 $A = X\Lambda X^{-1}$ 의 eigenvalue에 대하여 $P_A(A) = XP_A(\Lambda)X^{-1} = 0$ 이 성립하게 된다.

즉, 특성방정식이 0이 되므로 이때의 해가 행렬의 eigenvalue가 된다.



Krylov Subspaces

Arnoldi and Polynomial Approximation

이처럼 $P_{H_k}(H_k) = 0$ 을 만족시키는 해를 찾아서 A 의 eigenvalue를 추정할 수 있다.

이를 위해서는 $\|P_{H_k}(A)\|_2$ 가 최소가 되는 값을 찾아야 한다.

→ $\|P_{H_k}(A)\|_2$ 가 아니라 $\|P_{H_k}(A)q_1\|_2$ 이 최소가 되는 것을 보여야 한다!

$$D = \prod_{i=1}^k (\Lambda - \mu_i I_n) = \begin{bmatrix} \prod_{i=1}^k (\lambda_1 - \mu_i) & & & \\ & \prod_{i=1}^k (\lambda_2 - \mu_i) & & \\ & & \ddots & \\ & & & \prod_{i=1}^k (\lambda_n - \mu_i) \end{bmatrix}$$

H_k 의 eigenvalue를 μ_1, \dots, μ_k 라고 할 때,

$$P_{H_k}(z) = \prod_{i=1}^k (z - \mu_i)$$

A 대신 $X\Lambda X^{-1}$ 를 대입하면

$$P_{H_k}(A)q_1 = XP_{H_k}(\Lambda)X^{-1}q_1 = X \prod_{i=1}^k (\Lambda - \mu_i I_n) X^{-1}q_1 = XD X^{-1}q_1$$



Krylov Subspaces

Proof

k 차 함수 f 에 대하여 $\|f(A)q_1\|_2$ 가 minimal한 예시는 다음과 같다.

$$f(x) = x^k + f_{k-1}x^{k-1} + \cdots + f_0$$

$$f(A)q_1 = (A^k + f_{k-1}A^{k-1} + \cdots + f_0I)q_1$$

$$= A^k q_1 + K_k \cdot \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{k-1} \end{pmatrix}$$

$$= A^k q_1 + Q_k y$$

이때 $K_k = Q_k R_k$ 로 분해할 수 있으므로 $y \in \mathbb{R}^k$ 의 벡터로 나타낼 수 있다.

따라서 $\|f(A)q_1\|_2$ 를 minimal하게 만들어주는 f 를 찾는 것은

$\|A^k q_1 + Q_k y\|_2$ 를 minimal하게 만들어주는 y 를 찾는 것과 동일하다.

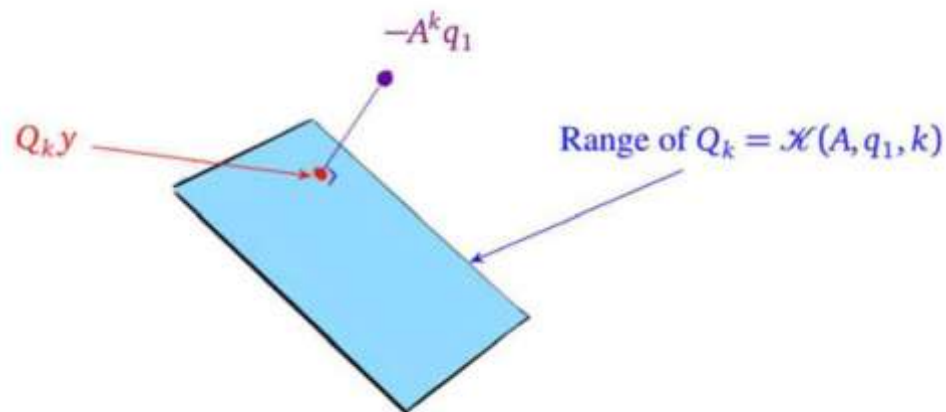


Krylov Subspaces

Proof

이를 least-squares problem으로 상정하면 솔루션은 $Q_k^T f(A)q_1 = 0$ 을 만족한다.

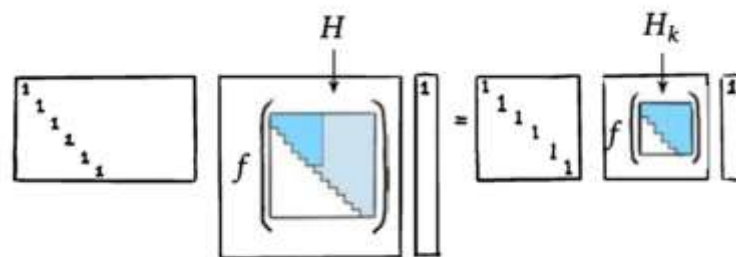
즉 에러 벡터가 range of Q_k 에 orthogonal하다.



이때 $f(A) = Qf(H)Q^T$ 를 이용해 아래와 같이 식을 풀어 쓸 수 있다.

$$0 = Q_k^T f(A)q_1 = Q_k^T Q f(H) Q^T q_1 = [I_k, 0] f(H) e_1.$$

$$[I_k, 0] f(H) e_1 = I_k f(H_k) e_1.$$



Krylov Subspaces

Invariance Properties

이렇게 구한 H_k 의 eigenvalue(=Ritz value)는 아래와 같은 properties를 만족한다.

(1) Translation-invariance

만약 A 가 $A + \sigma I$ 로 바뀌고 b 는 그대로라면, $\{\theta_j\}$ 는 각 스텝 k 에서 $\{\theta_j + \sigma\}$ 로 바뀐다.

(2) Scale-invariance


만약 A 가 σA 로 바뀌고 b 는 그대로라면, $\{\theta_j\}$ 는 각 스텝 k 에서 $\{\sigma\theta_j\}$ 로 바뀐다.

(3) Invariance under unitary similarity transformation

만약 A 가 UAU^T 로 바뀌고 b 는 Ub 로 바뀐다면, $\{\theta_j\}$ 는 변하지 않는다.

(U : unitary 행렬)





3. How Arnoldi Locates Eigenvalues

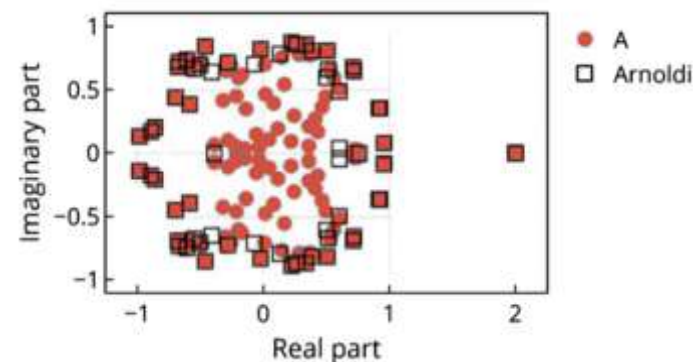
How Arnoldi Locates Eigenvalues

Computing Eigenvalues by the Arnold Iteration

Arnoldi Iteration은 수치선형대수에서 사용되는 대부분의 iterative 알고리즘의 기초이자, non hermitian matrix의 eigenvalues를 찾는 기법이다.

이전 장에서 소개된 바와 같이, 각 스텝 n 에서 Hessenberg 행렬 H_n 의 eigenvalue를 QR 알고리즘 등을 통해 계산한다. 이러한 값은 기하학적으로 빠르게 수렴하는 모습을 보이는데, 이때 수렴한 값을 A 의 eigenvalue로 생각할 수 있다.

Arnold Iteration을 사용하면 A 의 모든 eigenvalue를 찾기는 힘들 수 있지만 outlying eigenvalue를 먼저 찾을 수 있다.



How Arnoldi Locates Eigenvalues

Arnoldi Lemniscates

- Lemniscates: 아래와 같은 curve (p : polynomial, C : 실수)

$$\{z \in \mathbb{C}: |P(z)| = C\}$$

이때 p 를 Arnoldi polynomial p_k 로 두고 C 를 아래와 같은 값으로 두자.

$$C = \frac{\|p_k(A)b\|}{\|b\|}$$

위 식을 대입해 만들어지는 curve를 Arnoldi Lemniscates라고 한다.

Iteration number n 이 커짐에 따라 이러한 lemniscate가 A 의 outlying eigenvalue를 둘러싸는 것처럼 보이다가 한 점(eigenvalue)으로 수렴한다.



How Arnoldi Locates Eigenvalues

Arnoldi Lemniscates

예를 들어 $m=100$ 인 정방행렬 A 의 원소 자리에 평균이 0이고 표준편차가 $m^{-1/2}$ 인 실수 정규분포를 따르는 무작위 수를 넣는다고 가정해보자.

이는 A 의 eigenvalue가 unit disk $|z| \leq 1$ 안에 고르게 분포하도록 하기 위함이다.

만약 원소 a_{11} 을 1.5로 바꾸어 outlying eigenvalue를 만들고 Arnoldi Iteration을 적용하면, 아래 그림과 같다.

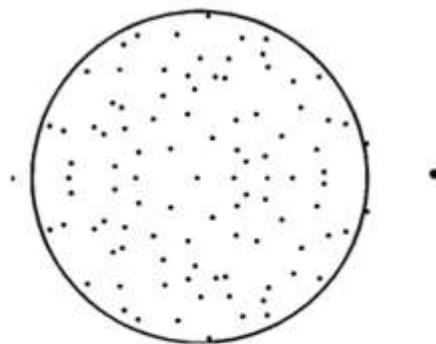


Figure 34.2. Eigenvalues of a 100×100 matrix A , random except in the 1, 1 position. The circle is the unit circle in \mathbb{C} . The eigenvalues are approximately uniformly distributed in the unit disk except for the outlier $\lambda \approx 1.4852$.



How Arnoldi Locates Eigenvalues

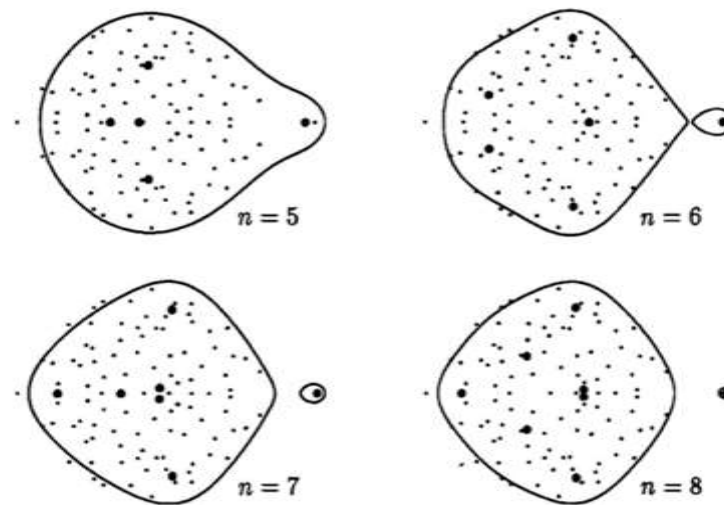
Arnoldi Leminiscates

아래 그림 속 작은 점은 A 의 eigenvalue, 굵은 점은 H_k 의 eigenvalue이다.

n 이 1일 때는 leminiscate가 원에 가깝고 outlying eigenvalue에 영향을 받지 않는다.

그러나 n 이 커짐에 따라 다음 그림과 같이 λ 의 방향으로 부풀었다가 떨어져나가고 새로운 leminiscate가 생성된다.

이후 이 성분은 수축해서 특정 eigenvalue 값으로 수렴하게 된다.



How Arnoldi Locates Eigenvalues

Arnoldi Lemniscates

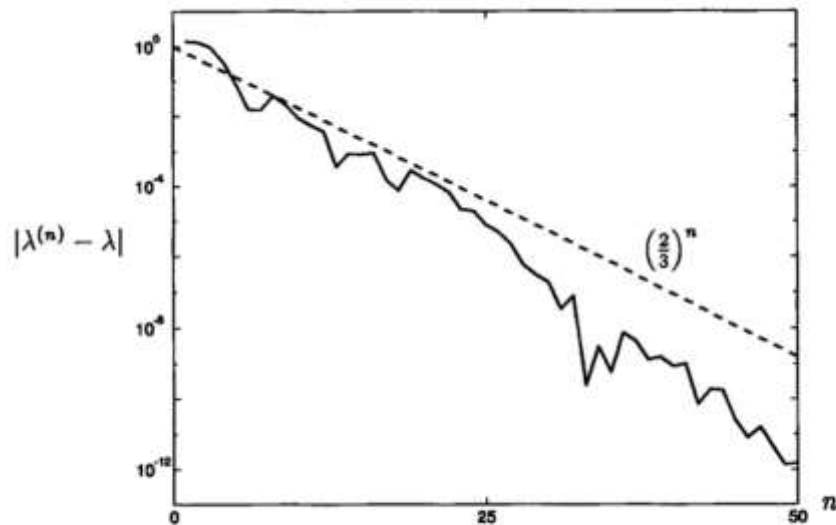


Figure 34.4. Convergence of the rightmost Arnoldi eigenvalue estimate.

다음 그림은 $|\lambda^{(k)} - \lambda|$ 이 수렴하는 모습을 그래프로 나타낸 것이다.

($\lambda^{(k)}$): k번째 스텝에서 λ 를 Arnoldi eigenvalue estimates로 구한 값)

약 50번의 반복 후에 소수점 12자리 정도의 차이가 나는 정확도를 얻을 수 있었다.





4. Lanczos Process

Basic concepts for Lanczos process

Arnoldi process

$$Q^T A Q = H$$

H 행렬의 $k \times k$ block을 통해서 A 행렬의 eigenvalue를 추정

1. Q의 1~k번째 열과 H의 1~k-1번째 열을 알고있다고 가정
2. $i \leq k$ 일 때 h_{ik} 를 다음을 이용해 구할 수 있다.

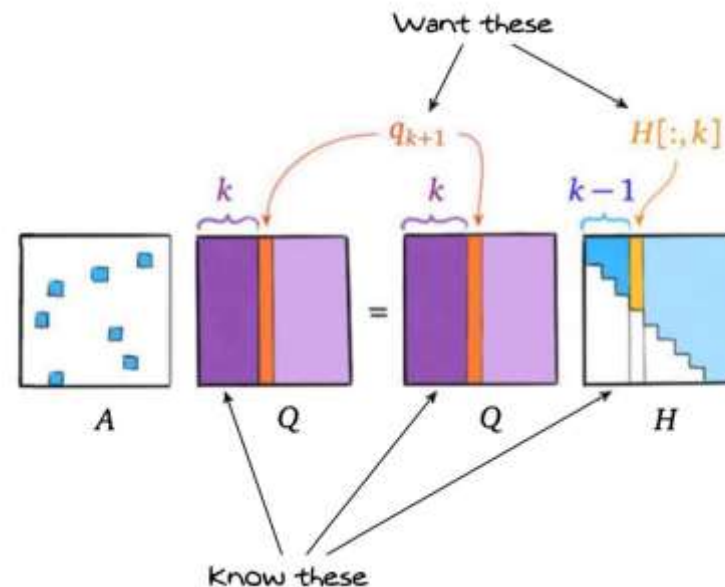
$$q_i^T A q_k = h_{ik}$$

3. $i \leq k$ 일 때 h_{ik} 를 구했으므로 $h_{k+1,k}$, q_{k+1} 를 다음을 이용해 구할 수 있다.

$$A q_k = h_{1k} q_1 + \dots + h_{kk} q_k + h_{k+1,k} q_{k+1}$$

$$r_k = A q_k - h_{1k} q_1 - \dots - h_{kk} q_k = h_{k+1,k} q_{k+1}$$

$$h_{k+1,k} = \|r_k\|, q_{k+1} = r_k / h_{k+1,k}$$



Basic concepts for Lanczos process

Lanczos process

Lanczos process는 Arnoldi process의 특별한 경우이다.

Arnoldi process에서 A행렬이 symmetric이라면 $Q^T A Q = T$ 에서 T는 헤센베르크 행렬이자 대칭 행렬이어야 한다. 즉 T는 tri-diagonal matrix(삼중 대각행렬)이다.

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & \cdots & & 0 \\ \beta_1 & \alpha_2 & \cdots & & \\ \vdots & & \ddots & & \vdots \\ & & & \ddots & \beta_{n-1} \\ 0 & \cdots & \beta_{n-1} & \alpha_n & \end{pmatrix}.$$



Basic concepts for Lanczos process

Lanczos process

T행렬의 특징으로 인해 Lanczos process는 Arnoldi process보다 r_k 를 구하는 계산이 간략해진다.

$q_1, \dots, q_k, \alpha_1, \dots, \alpha_{k-1}$, and $\beta_1, \dots, \beta_{k-1}$ 을 알고 있다고 가정할 때, $q_{k+1}, \alpha_k, \beta_k$ 는 다음과 같이 구할 수 있다.

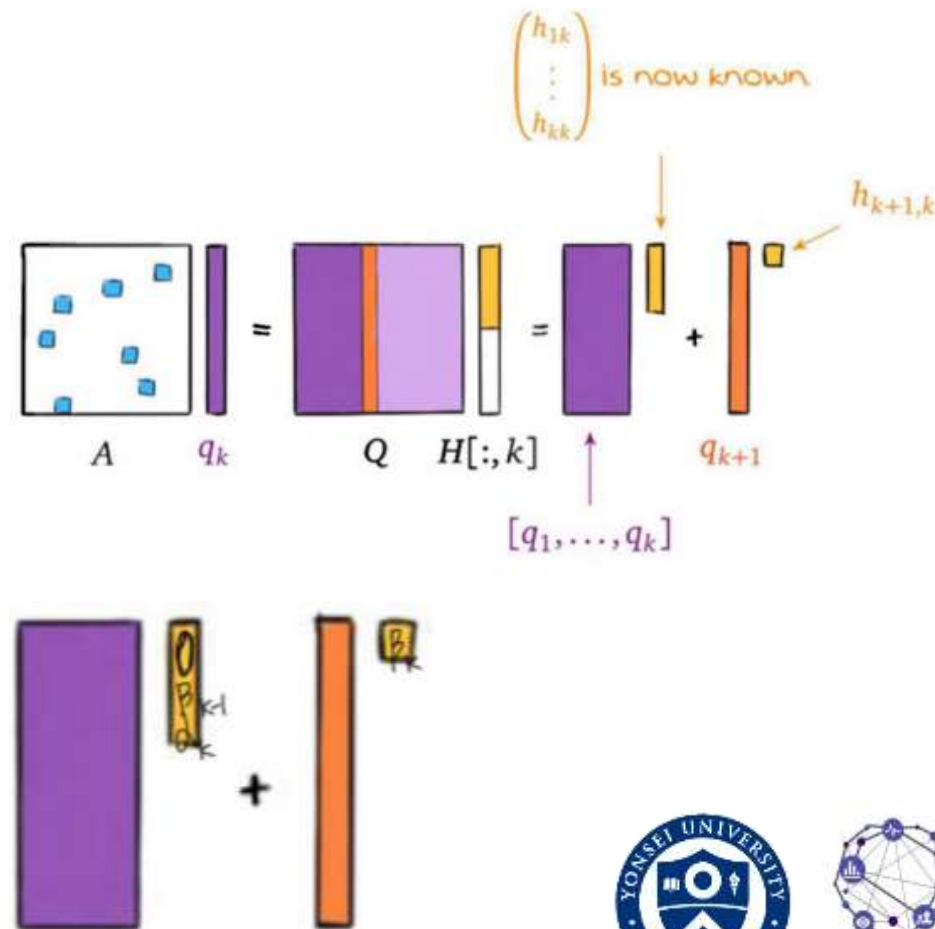
1. $\alpha_k = q_k^T A q_k$
2. $r_k = A q_k - \beta_{k-1} q_{k-1} - \alpha_k q_k = \beta_k q_{k+1}$
3. $\beta_k = \|r_k\|_2$
4. $q_{k+1} = r_k / \beta_k$

처음에 랜덤으로 q_1 을 설정하고 위 과정을 반복하면 Q 행렬과 T 행렬의 열들을 구할 수 있다.

이 과정을 통해 T_k 의 k 개의 고유값으로 A 의 고유값을 추정할 수 있다.

모든 step에서 1번의 내적, 1번의 norm 계산, 5번의 벡터 연산이 필요하므로 연산량은 $O(n)$ 이다.

(Arnoldi는 iteration k 에서 $O(kn)$ 의 연산량 필요)



Basic concepts for Lanczos process

Convergence estimates

먼저 대칭행렬 A 의 largest eigenvalue λ_1 은 Courant-Fischer minmax theorem을 통해 다음과 같이 구할 수 있다.

$$\lambda_1 = \max_{x \neq 0} \frac{x^T A x}{\|x\|_2^2}.$$

λ_1 을 추정하기 위하여 T_k 행렬의 top eigenvalue μ_1 을 사용할 것이다. 이때 $T_k = Q_k^T A Q_k$ 라는 사실과 Q_k 의 columns이 Krylov subspace $\mathcal{K}(A, q_1, k)$ 를 span 한다는 사실을 통해 다음과 같이 표현할 수 있다.

$$\mu_1 = \max_{x \neq 0} \frac{x^T Q_k^T A Q_k x}{\|x\|^2} = \max_{y \in \mathcal{K}(A, q_1, k) \setminus \{0\}} \frac{y^T A y}{\|y\|^2}$$

$\mathcal{K}(A, q_1, k)$ 가 \mathbb{R}^n 의 subset이므로 $\mu_1 \leq \lambda_1$ 이다.

Courant-Fischer minmax theorem

If $A \in \mathbb{R}^{n \times n}$ is symmetric, then

$$\lambda_k(A) = \max_{\dim(S)=k} \min_{0 \neq y \in S} \frac{y^T A y}{y^T y}$$

with $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. The set S that attains the maximum is the span of the eigenvectors for the k largest eigenvalues, $\{q_1, \dots, q_k\}$.



Basic concepts for Lanczos process

Convergence estimates

$y \in \mathcal{K}(A, q_1, k)$ 인 벡터 y 는 $f(x) = f_0 + f_1x + f_2x^2 + \dots + f_{k-1}x^{k-1}$ 를 이용하여 다음과 같이 표현할 수 있다.

$$y = f(A)q_1 \quad y = f_0q_1 + f_1Aq_1 + f_2A^2q_1 + \dots + f_{k-1}A^{k-1}q_1$$

이를 이용하여 μ_1 을 다음과 같이 표현할 수 있다.

$$\mu_1 = \max_{f \text{ of degree } k-1} \frac{q_1^T f(A)^T A f(A) q_1}{q_1^T f(A)^2 q_1}.$$

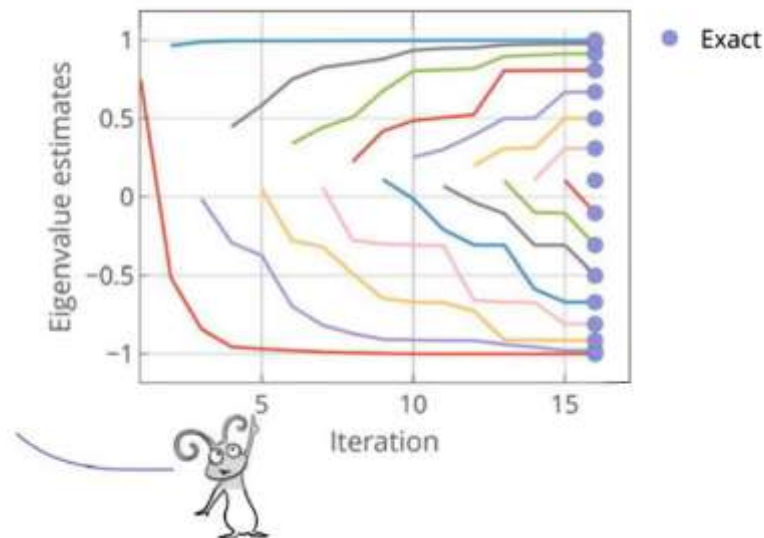


Basic concepts for Lanczos process

Convergence estimates

지금까지는 A 를 사용했지만 A 대신 $-A$ 를 사용하면 위의 과정과 유사한 방법으로 λ_n 을 얻을 수 있다.

Arnoldi case에서 봤듯이 convergence는 extremal eigenvalues λ_1 and λ_n 부터 시작하여 점차 안쪽에 있는 eigenvalue로 이동한다.



Basic concepts for Lanczos process

Convergence estimates

m개의 Chebyshev Point는 $[-1,1]$ 범위에 다음과 같이 정의된다

$$x_j = \cos \theta_j, \quad \theta_j = \frac{(j - \frac{1}{2})\pi}{m}, \quad 1 \leq j \leq m.$$

Chebyshev Point는 내부에서는 $O(m^{-1})$, ± 1 근처에서 $O(m^{-2})$ 의 사이에 위치하게 된다

이때 Convergence는 A 고유값 분포와 관련이 있다

- (1) A의 고유값이 Chebyshev Point보다 균일하게 위치하면 Lanczos Iteration으로 Outlier를 찾을 수 있다
- (2) A의 고유값이 (1)보다 더 균등하게 분포하면 Outlying Eigenvalue에 첫번째로 수렴한다
- (3) 고유값이 Uniform distribution이면 Outlier로 빠르게 수렴한다
- (4) A의 고유값이 오히려 Endpoint에 밀집되면 "Inliers"에 수렴한다



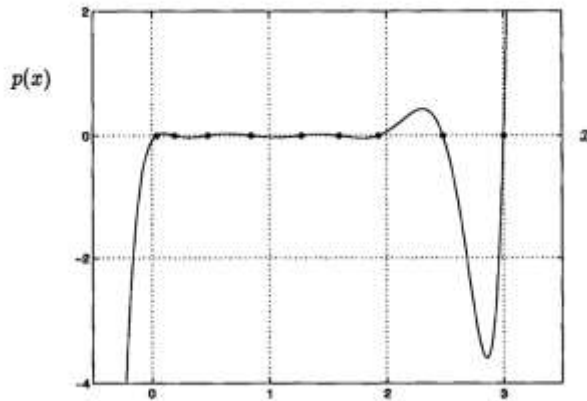
Basic concepts for Lanczos process

Convergence estimates

Ex) A: 203 X 203행렬

$A = \text{diag}(0, .01, .02, \dots, 1.99, 2, 2.5, 3.0)$

A는 eigenvalue가 $[0, 2]$ 사이에 밀집되어 있고 2.5와 3.0의 두 개의 Outlier가 존재한다



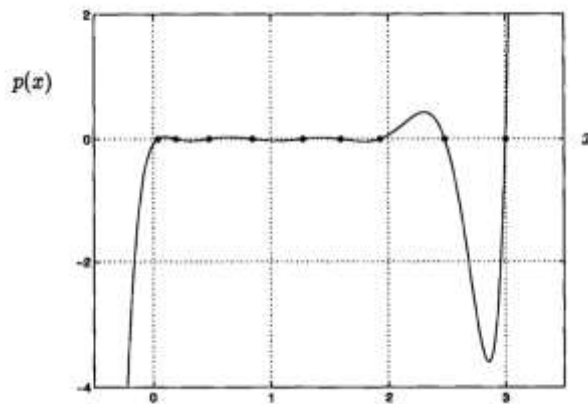
Lanczos Polynomial의 **Step 9**에서 Lanczos Eigenvalue Estimates는 7개 $[0, 2]$, 2개는 2.5과 3.0 근처에 존재한다
이때 3개의 Leading Eigenvalue 1.93, 2.48, 2.999962로 존재한다 (Five-digit Accuracy)



Basic concepts for Lanczos process

Convergence estimates

Ex)



$p(x)$ 그래프는 $x \approx 3$ 근처에서 기울기가 가파르기 때문에 $p(3)$ 값이 작게 된다면 3 근처에서 근을 가지게 된다

$p(2.5)$ 값도 작게 된다면 2.5 근처에서 근을 가지게 된다

따라서 $p(x)$ 와 같은 그래프 개형이면 Outlier인 Eigenvalue를 정확하게 추정할 수 있게 된다



Basic concepts for Lanczos process

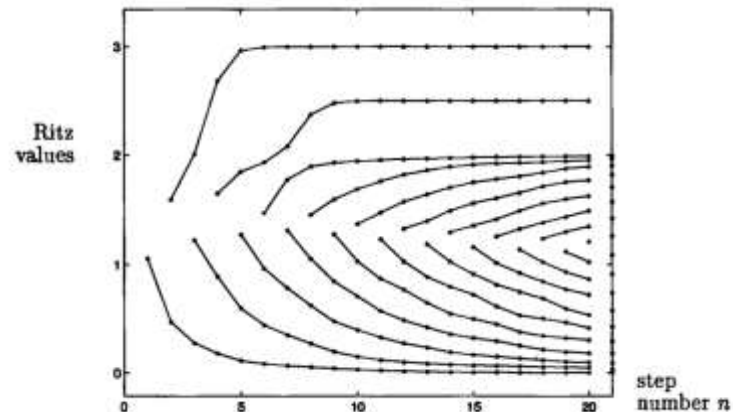
Convergence estimates

Ex)

Lanczos Polynomial의 **Step 20**에서 3개의 Leading Eigenvalues는

1.9906, 2.499999999987, 3.00000000000000로 존재한다 (Fifteen-digit Accuracy)

이때도 $p(x)$ 그래프는 x 는 2.5, 3.0 근처에서 기울기가 가파르기 때문에 해당 값으로 수렴하게 될 것이다



$n = 5$ 에서 3.0, $n = 10$ 부터 2.5로 수렴해가는 과정이 보인다





5. Ghost Eigenvalues

Practical issues and ghost eigenvalues

Ghost eigenvalues

arnoldi에서 lanczos로 오는 과정에서 우리는 q_k 를 q_i ($i < k - 1$)에 orthogonal하게 project하지 않았다. 만약 roundoff error가 없다면 문제가 되지 않는다.

그러나 roundoff error가 있으면 매우 작은 error가 점차 쌓이면서 q_k 가 q_1 에 orthogonal 하지 않게 된다.

이로 인해 ghost eigenvalues 문제가 발생한다.

$$Aq_k = h_{1k}q_1 + \dots + h_{kk}q_k + h_{k+1,k}q_{k+1}$$

$$r_k = Aq_k - h_{1k}q_1 - \dots - h_{kk}q_k = h_{k+1,k}q_{k+1}$$

$$h_{k+1,k} = \|r_k\|, q_{k+1} = r_k / h_{k+1,k}$$

$$\alpha_k = q_k^T Aq_k$$

$$r_k = Aq_k - \beta_{k-1}q_{k-1} - \alpha_k q_k = \beta_k q_{k+1}$$

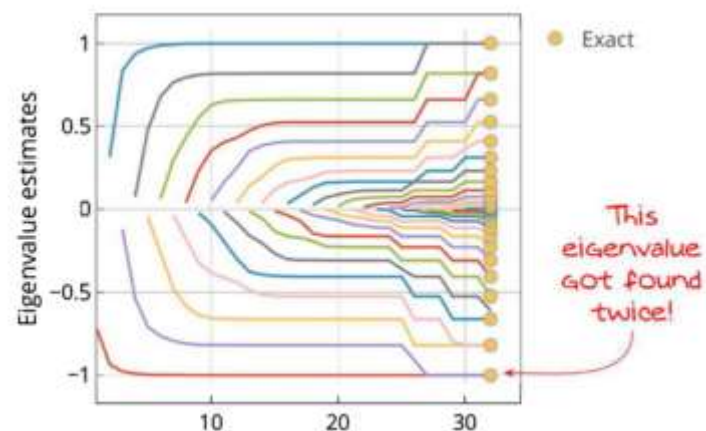
$$\beta_k = \|r_k\|_2$$

$$q_{k+1} = r_k / \beta_k$$

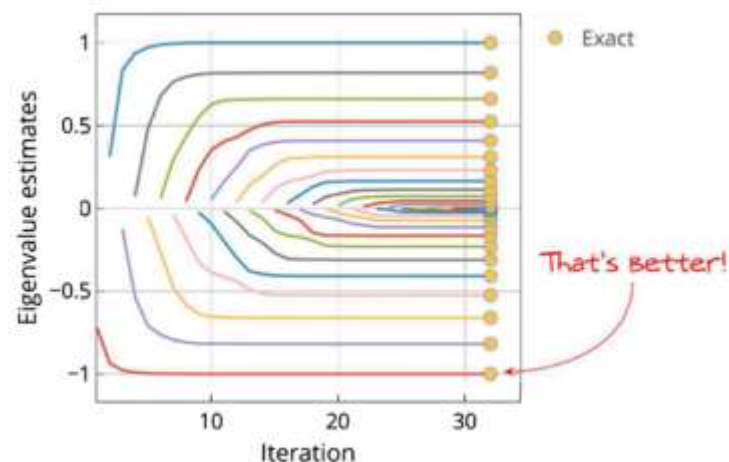


Practical issues and ghost eigenvalues

Ghost eigenvalues



첫번째 그림은 roundoff error로 인해서 T_k 의 eigenvalue가 A 의 똑같은 eigenvalue로 수렴하는 상황이다. 이로 인해 안쪽에 있는 eigenvalue를 놓치게 된다.



두번째 그림은 완전하게 re-orthogonalization을 해주어서 올바르게 수렴하는 상황이다. 완전하게 re-orthogonalization을 해주면 수렴은 잘되지만 연산량이 기존보다 훨씬 많아진다는 단점이 존재한다. (iteration k에서의 연산량 = $O(nk)$)



Practical issues and ghost eigenvalues

Ghost eigenvalues

하나의 eigenvalue가 정확하게 찾아질 때 문제가 발생한다.

예를 들어 λ_1 이 λ_2 보다 많이 크고 수렴한다고 하자. 즉, T_k 의 top eigenvalue $\mu_1^{(k)}$ 가 λ_1 과 매우 가깝다. 이는 다음을 의미한다.

$$\mu_1^{(k)} = \max_{y \in \mathcal{K}(A, q_1, k)} \frac{y^T A y}{\|y\|^2} \approx \max_{x \in \mathbb{R}^n} \frac{x^T A x}{\|x\|^2},$$

이때, top eigenvector x_1 는 krylov subspace $\mathcal{K}(A, q_1, k)$ 에 존재한다.

그 후 iteration을 진행할 때, Aq_k 는 $\text{span}\{q_1, \dots, q_{k-2}\} = \mathcal{K}(A, q_1, k-2)$ 에 orthogonal 해야한다.

이때, arnoldi에서 lanczos로 넘어오는 과정으로 인해 실질적으로 Aq_k 를 $\mathcal{K}(A, q_1, k-2)^\perp$ 에 project 하지 않는다.



Practical issues and ghost eigenvalues

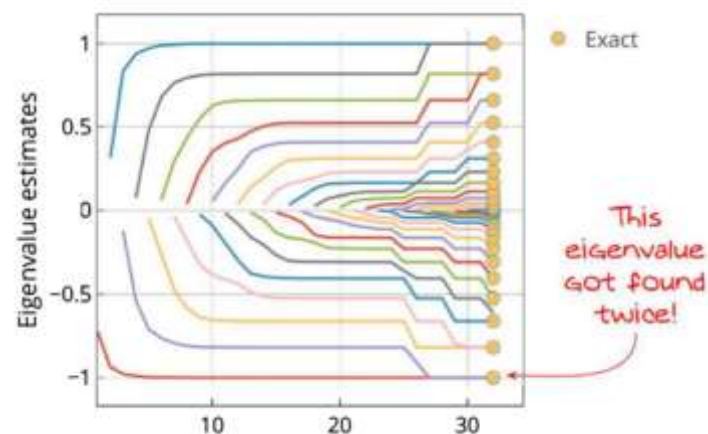
Ghost eigenvalues

roundoff error로 인해 위 과정에서 문제가 발생한다.

iteration을 하면서 A 를 곱하는데 이때, x_1 의 방향의 작은 오차가 점점 누적되게 된다.

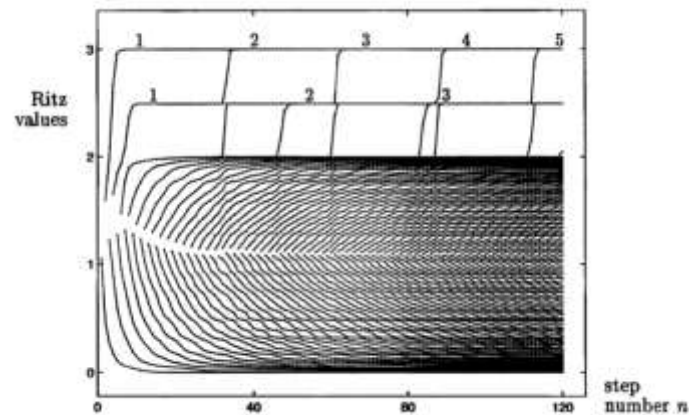
그로 인해, q_k 가 다시 x_1 방향을 가리키게 되고 λ_1 을 재발견 하게 된다.

만약 다른 eigenvalue와 well-separated 되는 eigenvalue가 없다면 모든 eigenvalue가 비슷한 속도로 수렴하게 된다.(일찍 수렴하는 것이 없음) 그렇게 된다면 ghost eigenvalue 문제가 나타나지 않는다.



Practical issues and ghost eigenvalues

Ghost eigenvalues



Ghost Eigenvalue인 3.0과 2.5이 각각 4번, 2번 나타나는 이유로 다음과 같은 직관적인 해석도 가능하다

(1) A 고유값이 Point가 아닌 Small Interval이다

- $p(z)$ 값을 작게 하기 위해서 Ghost Eigenvalue가 나타난다

(2) A 고유값으로 수렴이 대응되는 고유벡터 성분을 소멸시킨다

- round error로 인해 그 성분이 다시 나타나며 iteration이 지날수록 그 성분을 없애기 위해 Ghost Eigenvalue가 나타난다



END

References

- [1] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. SIAM.
- [2] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). SIAM.

Figure

- [1] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p 247), SIAM.
- [2] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p 251), SIAM.
- [3] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.214). SIAM.
- [4] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.215). SIAM.
- [5] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.218). SIAM.
- [6] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.216). SIAM.
- [7] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.219). SIAM.
- [8] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p 245), SIAM.
- [9] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.220). SIAM.
- [10] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.224). SIAM.
- [11] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.218). SIAM.

- [12] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p 263), SIAM.
- [13] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p 264), SIAM.
- [14] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.216). SIAM.
- [15] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.225). SIAM.
- [16] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.227). SIAM.
- [17] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p.280). SIAM.
- [18] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p.281). SIAM.
- [19] Trefethen, L. N., & Bau , D. (1997). Numerical linear algebra (1st ed.). (p.282). SIAM.
- [20] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.229). SIAM.
- [21] Darve, E., & Wootters, M. (2021). Numerical Linear Algebra with Julia. (p.230). SIAM.

