(복습) 대각화가 되기 위한 조건 :

어떤 행렬 A가 대각화 되기 위한 조건은 nxn 정방 행렬의 경우선형 독립인 n개의 고유벡터들을 갖고 있는 것.

$$S^{-1}AS=\Lambda=\begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{bmatrix}$$

- 여기서 A가 nxn 행렬 즉 정방행렬이라고 했을 때,

- N개의 선형적으로 독립인 고유벡터들을 갖게 된다면 좌측의 그림처럼 고윳값 분해가 성립됨

- 여기서 S는 고유벡터들을 열벡터로 갖고 있는 고유벡터 행렬이고, A는 고윳값들을 대각요소에 갖고 있는 대각행렬로 이를 고윳값 행렬이라고 부름

$$S^{-1}AS=\Lambda=\begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{bmatrix} \qquad AS = A\begin{bmatrix}| & | & & | \\ x_1 & x_2 & \cdots & x_n \\ | & | & & |\end{bmatrix} = \begin{bmatrix}| & | & & | \\ \lambda_1 x_1 & \lambda_2 x_2 & \cdots & \lambda_n x_n \\ | & | & & |\end{bmatrix}$$

이를 증명하는 방법은 먼저 고유벡터 행렬 S에 행렬 A를 곱해주는 것으로 시작함

AS의 각 열 벡터는 S의 각 열벡터에 행렬 A를 곱한 것

$A_{X1} = \lambda_{1X1}, A_{X2} = \lambda_{2X2}, \ldots A_{Xn} = \lambda_{nXn}$ 이므로 아래와 같이 연산과 분리가 가능

$$\begin{bmatrix} \lambda_1 x_1 & \lambda_2 x_2 & \cdots & \lambda_n x_n \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}\begin{bmatrix}\lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n\end{bmatrix} = S\Lambda$$

대각화를 이용하면 행렬의 거듭제곱을 쉽게 구할 수 있고,

대각화를 이용해서 차분방정식을 간단하게 풀 수 있음

대각화라는 기법이 연산에 있어 상당히 중요한 스킬이라는 점을 확인할 수 있었음

그런데 만약 대각화가 되지 않는 행렬은 어떻게 처리해야 할까?

-> 요르단 표준형을 찾아야함

요르단 행렬은 대각행렬과 유사한 형태를 띄고 있음

만약 행렬 A가 s개의 독립인 고유벡터들을 갖는다면, 행렬 A는 s개의 블럭들을 갖는 행렬과 유사함

$$J = M^{-1}AM = \begin{bmatrix} J_1 & & \\ & \ddots & \\ & & J_s \end{bmatrix}$$

각각의 요르단 블록(또는 요르단 셀) J_i 는 오직 한 개의 고윳값 λi와 한 개의 고유벡터를 지니는 삼각행렬

$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

$$J = \left(\begin{array}{ccccc|cc|c|ccc} 3 & 1 & & & & & & & & & \\ & 3 & 1 & & & & & & & & \\ & & 3 & 1 & & & & & & & \\ & & & 3 & 1 & & & & & & \\ & & & & 3 & & & & & & \\ \hline - & - & - & - & - & - & - & - & - & - & - \\ & & & & & 3 & 1 & & & & \\ & & & & & & 3 & & & & \\ \hline - & - & - & - & - & - & - & - & - & - & - \\ & & & & & & & 4 & & & \\ \hline - & - & - & - & - & - & - & - & - & - & - \\ & & & & & & & & 5 & 1 & \\ & & & & & & & & & 5 & 1 \\ & & & & & & & & & & 5 \end{array}\right)$$

요르단 셀 예시

크기 5의 요르단 셀, 크기 2의 요르단 셀, 크기 1의 요르단 셀, 크기 3의 요르단 셀

* 요르단 셀에서 대각 요소를 차지하고 있는 고윳값들 바로 위 대각에 1을 추가해준다는 것이 포인트

요르단 셀의 성질

1. 하나의 고윳값에 대응하는 요르단 셀의 개수는 기하학적 중복도, 즉 일차독립인 고유벡터들의 개수와 일치함

2. 요르단 셀의 크기는 해당하는 고윳값에 대한 고유벡터들의 성질에 의하여 크기가 결정됨

3. 단, 그 크기들의 합은 그 고윳값에 대한 대수적 중복도가 됨

4. 만일 행렬 A의 모든 고윳값의 기하적 중복도와 대수적 중복도가 같게 되면, 모든 요르단 셀의 크기는 1x1이 되고 그 개수는 “기하적 중복도의 합 = 대수적 중복도의 합 = 행렬의 크기 " 가 된다. 즉 대각선행렬이 되며 이 경우가 대각화 가능한 행렬이 될 필요충분조건

대각화가 불가능한 행렬들을 요르단 표준형을 통해 유사대각화 해준다는 것이 핵심

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 0 & 2 & 0 \\ 0 & -2 & 1 \end{bmatrix}$$

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$= a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

$$= a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - (a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33} + a_{13}a_{22}a_{31})$$

$$\text{et}(A - \lambda I) = (\lambda\text{-}2)(\lambda\text{-}1)\text{^}2$$

$\therefore \lambda = 2,1,1$

$$x_1 = \begin{bmatrix} -8 \\ 1 \\ -2 \end{bmatrix} \qquad x_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$ -> 두 개의 고유벡터를 갖게 됨

$$J = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$ 결과적으로 두 개의 고유벡터를 가지므로 총 2개의 요르단 블록으로 구성된 요르단 표준형이 우측의 행렬처럼 완성!

```
import numpy as np
from sympy import Matrix

a = np.array([[5, 4, 2, 1], [0, 1, -1, -1], [-1, -1, 3, 0], [1, 1, -1, 2]])
```

```
a
```

```
array([[ 5,  4,  2,  1],
       [ 0,  1, -1, -1],
       [-1, -1,  3,  0],
       [ 1,  1, -1,  2]])
```

```
m = Matrix(a)
```

```
m
```

$$\begin{bmatrix} 5 & 4 & 2 & 1 \\ 0 & 1 & -1 & -1 \\ -1 & -1 & 3 & 0 \\ 1 & 1 & -1 & 2 \end{bmatrix}$$

```
P, J = m.jordan_form()
```

```
# Jordan normal form
J
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

```
# transition matrix P
P
```

$$\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$