Diagonalization

Learning Objectives

- Determine if a given matrix is diagonalizable or not.
- If possible, diagonalize a matrix.

The Diagonalization Theorem

- An nxn matrix *A* is diagonalizable if and only if *A* has *n* linearly independent eigenvectors.

- If the above holds, we can write $\mathbf{A=PDP^{-1}}$ where *D* is a diagonal matrix. The diagonal entries of *D* are **eigenvalues of *A*** that correspond, respectively, to the **eigenvectors in *P***.

- In other words, *A* is diagonalizable if and only if there are enough eigenvectors to form a basis of $\mathbb{R}^n$.

- In this case we say *A* and *D* are ***similar.***

Example

- Diagonalize the following matrix, if possible.

$$A = \begin{bmatrix} 1 & 3 & 3 \\ -3 & -5 & -3 \\ 3 & 3 & 1 \end{bmatrix}$$

Solution in Python

```python
A = np.array([[1,3,3],[-3,-5,-3],[3,3,1]])
A
```

```
array([[ 1,  3,  3],
       [-3, -5, -3],
       [ 3,  3,  1]])
```

```python
l, ev = linalg.eig(A)
l
```

```
array([ 1., -2., -2.])
```

```python
ev
```

```
array([[-0.57735027, -0.30726304, -0.63333227],
       [ 0.57735027,  0.80875904, -0.12961592],
       [-0.57735027, -0.501496  ,  0.76294819]])
```

Solution in Python

- Are the eigen values linearly independent?
- We will look at the reduced row echelon form: If three is any column in the RREF that consists entirely of zeros, then the corresponding column in the original matrix was a linear combination of other columns, and hence, the columns are not linearly independent.

```python
import sympy as sp
rref_M, pivot_cols = sp.Matrix(ev).rref()
rref_M
```

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

Solution in Python

- Now, that we know that eigenvectors are linearly independent we know, we can diagonalize the matrix A. Let's form D and P.

```
D = np.diag(l)
D
```

```
array([[ 1.,  0.,  0.],
       [ 0., -2.,  0.],
       [ 0.,  0., -2.]])
```

```
P = ev
P
```

```
array([[-0.57735027, -0.30726304, -0.63333227],
       [ 0.57735027,  0.80875904, -0.12961592],
       [-0.57735027, -0.501496  ,  0.76294819]])
```

```
P_inv = linalg.inv(ev)
P_inv
```

```
array([[-1.73205081, -1.73205081, -1.73205081],
       [ 1.14725922,  2.52931314,  1.38205392],
       [-0.55659624,  0.35184619,  0.90844243]])
```

```
P@D@P_inv
```

```
array([[ 1.,  3.,  3.],
       [-3., -5., -3.],
       [ 3.,  3.,  1.]])
```

Theorem

- An nxn matrix with *n* distinct eigenvalues is diagonalizable
- It is not *necessary* for an nxn matrix to have n distinct eigenvalues in order to be diagonalizable, but this provides a sufficient condition for a matrix to be diagonalizable.

Example

- Diagonalize the following matrices, if possible and verify that $\mathbf{A=PDP^{-1}}$ in Python.
- $A = \begin{bmatrix} 5 & -8 & 1 \\ 0 & 0 & 7 \\ 0 & 0 & -2 \end{bmatrix}$
- $B = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 1 & 4 & -3 & 0 \\ -1 & -2 & 0 & -3 \end{bmatrix}$

Solution

```
A = np.array([[5,-8,1],[0,0,7],[0,0,-2]])
A
```

```
array([[ 5, -8,  1],
       [ 0,  0,  7],
       [ 0,  0, -2]])
```

```
l, ev = linalg.eig(A)
l
```

```
array([ 5.,  0., -2.])
```

```
ev
```

```
array([[ 1.        ,  0.8479983 , -0.75122223],
       [ 0.        ,  0.52999894, -0.63465327],
       [ 0.        ,  0.        ,  0.1813295 ]])
```

```
ev@np.diag(l)@linalg.inv(ev)
```

```
array([[ 5., -8.,  1.],
       [ 0.,  0.,  7.],
       [ 0.,  0., -2.]])
```

Solution

- $B = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 1 & 4 & -3 & 0 \\ -1 & -2 & 0 & -3 \end{bmatrix}$ has repeated eigenvalues. However, having distinct eigenvalues, is a **sufficient condition, but not necessary.**
- Therefore, we need to still check on the independence of the eigenvectors.

Solution

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

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

```
l, ev = linalg.eig(A)
l
```

```
array([-3., -3.,  5.,  5.])
```

```
ev
```

```
array([[ 0.        ,  0.        ,  0.98473193,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.87287156],
       [ 1.        ,  0.        ,  0.12309149,  0.43643578],
       [ 0.        ,  1.        , -0.12309149, -0.21821789]])
```

Solution

```
rref_M, pivot_cols = sp.Matrix(ev).rref()
print(pivot_cols)
rref_M
```

```
(0, 1, 2, 3)
```

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

```
D = np.diag(l)
D
```

```
array([[-3.,  0.,  0.,  0.],
       [ 0., -3.,  0.,  0.],
       [ 0.,  0.,  5.,  0.],
       [ 0.,  0.,  0.,  5.]])
```

Solution

```
P = ev
P
```

```
array([[ 0.        ,  0.        ,  0.98473193,  0.        ],
       [ 0.        ,  0.        ,  0.        ,  0.87287156],
       [ 1.        ,  0.        ,  0.12309149,  0.43643578],
       [ 0.        ,  1.        , -0.12309149, -0.21821789]])
```

```
P_inv = linalg.inv(ev)
P_inv
```

```
array([[-0.125     , -0.5       ,  1.        ,  0.        ],
       [ 0.125     ,  0.25      ,  0.        ,  1.        ],
       [ 1.0155048 ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  1.14564392,  0.        ,  0.        ]])
```

```
P@D@P_inv
```

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