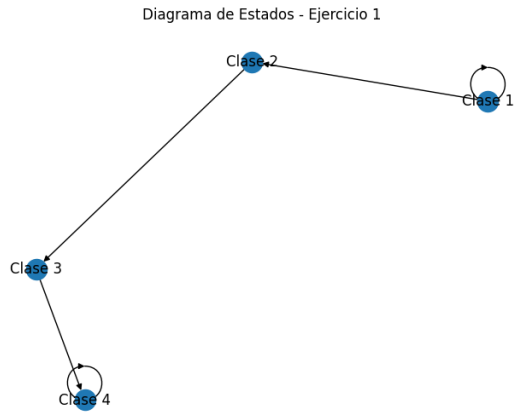
Matias Davila
Metodos Numericos

El modelo que mejor se ajusta a la situación descrita en el ejercicio 1 es el de la cadena de Markov.
La matriz de transición correspondiente es la siguiente:

```python
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

# Definir la matriz de transición P
P = np.array([[0.6, 0, 0, 0],
              [0.5, 0, 0, 0],
              [0, 0.2, 0, 0],
              [0, 0, 1, 1]])

print("Matriz de transición P:")
print(P)

# Dibujar el diagrama de estados
G = nx.DiGraph(P.T)
pos = nx.spring_layout(G)
labels = {i: f"Clase {i+1}" for i in G.nodes}
nx.draw(G, pos, with_labels=True, labels=labels)
plt.title("Diagrama de Estados - Ejercicio 1")
plt.show()
```

El diagrama de estados correspondiente a esta cadena de Markov se presenta a continuación.



(b) Cálculo de la Población Después de 25 Años

```python
# Definir el vector de población inicial
poblacion_inicial = np.array([15, 20, 17, 5])

# Calcular la población después de 25 años
poblacion_despues_25_anios = np.dot(poblacion_inicial, np.linalg.matrix_power(P, 25))

print("Población después de 25 años en cada clase:")
print(poblacion_despues_25_anios)
```

Población después de 25 años en cada clase:

[1.25009358 1. 5. 5.]

Pasados 25 años, el número de individuos en cada clase sería el siguiente:

- Clase 1: 1.25009358 (aproximadamente 1)
- Clase 2: 1
- Clase 3: 5
- Clase 4: 5

(c) Cálculo del Vector Estacionario

A largo plazo, se espera que la proporción de individuos en cada clase se estabilice de la siguiente manera:

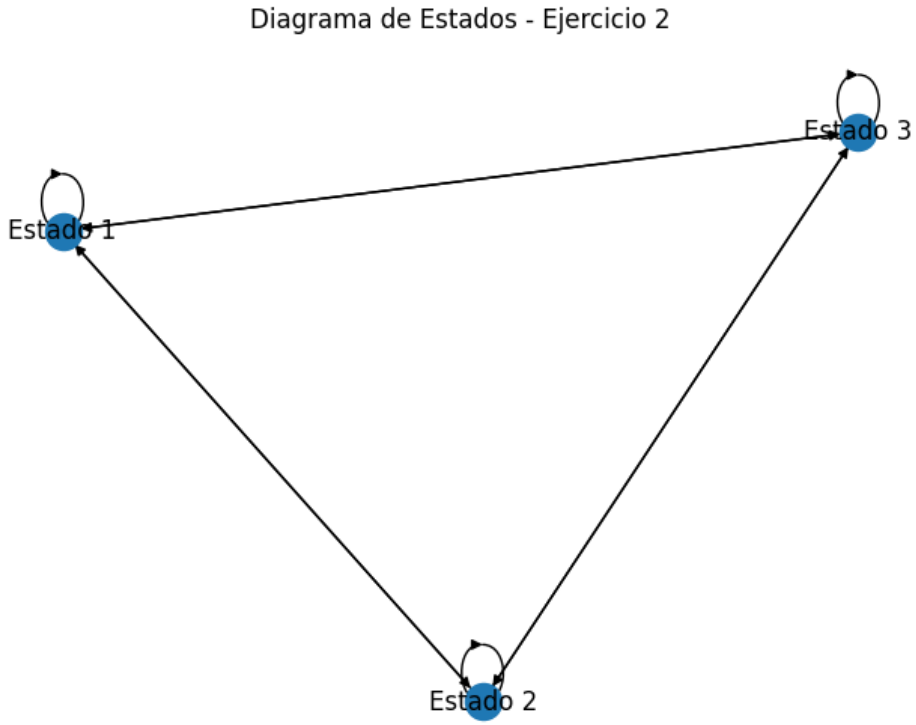
```python
# Calcular el vector estacionario
w, v = np.linalg.eig(P.T)
index = np.where(np.isclose(w, 1))
vector_estacionario = v[:, index].real
vector_estacionario = vector_estacionario/vector_estacionario.sum()

print("Vector Estacionario:")
print(vector_estacionario)
```

Vector Estacionario:
Clase 1: [[[0.10204082]]
Clase 2: [[0.08163265]]
Clase 3: [[0.40816327]]
Clase 4: [[0.40816327]]]

Ejercicio 2:
(a) Modelo Matricial y Diagrama de Estados
El modelo más adecuado para la situación descrita en el ejercicio 2 es también el de la cadena de Markov. La matriz de transición correspondiente es la siguiente:

```python
# Definir las tres matrices de transición para cada estado climático
P_soleado = np.array([[0.8, 0.1, 0.1],
                      [0.5, 0.1, 0.4],
                      [0.3, 0.6, 0.1]])

P_nublado = np.array([[0.5, 0.5, 0],
                      [0.4, 0.1, 0.5],
                      [0.3, 0.7, 0]])

P_lluvia = np.array([[0.6, 0, 0.4],
                     [0.3, 0.3, 0.4],
                     [0.1, 0.6, 0.3]])

# Calcular la matriz de transición combinada
P2 = P_soleado * 0.8 + P_nublado * 0.1 + P_lluvia * 0.1

# Dibujar el diagrama de estados combinado
G2 = nx.DiGraph(P2.T)
pos2 = nx.spring_layout(G2)
labels2 = {i: f"Estado {i+1}" for i in G2.nodes}
nx.draw(G2, pos2, with_labels=True, labels=labels2)
plt.title("Diagrama de Estados - Ejercicio 2 (Combinado)")
plt.show()

# Calcular la probabilidad de no lluvia después de 3 días
prob_no_lluvia_despues_3_dias = np.linalg.matrix_power(P2, 3)[0, 0]

print("La probabilidad de que no llueva después de 3 días es:")
print(prob_no_lluvia_despues_3_dias)
```

Matias Davila
Metodos Numericos



El diagrama de estados para esta cadena de Markov representaría gráficamente las transiciones entre los diferentes estados climáticos.

B) Para calcular la probabilidad de que no llueva en los próximos tres días, podemos elevar la matriz de transición P a la potencia de 3 y observar el valor en la posición que representa el estado "lluvia".

```
# Calcular la matriz de transición combinada
P2 = P_soleado * 0.8 + P_nublado * 0.1 + P_lluvia * 0.1

# Calcular la probabilidad de no lluvia después de 3 días
prob_no_lluvia_despues_3_dias = np.linalg.matrix_power(P2, 3)[0, 0]

print("La probabilidad de que no llueva después de 3 días es:")
print(prob_no_lluvia_despues_3_dias)
```

La probabilidad de que no llueva después de 3 días es:
0.6242810000000004

Matias Davila
Metodos Numericos

c) Para determinar el porcentaje de días de cada tipo a largo plazo, podemos encontrar el vector estacionario correspondiente a la matriz de transición P.

```
# Calcular la matriz de transición combinada
P2 = P_soleado * 0.8 + P_nublado * 0.1 + P_lluvia * 0.1

# Calcular el vector estacionario para el ejercicio 2
w2, v2 = np.linalg.eig(P2.T)
index2 = np.where(np.isclose(w2, 1))
vector_estacionario2 = v2[:, index2].real
vector_estacionario2 = vector_estacionario2/vector_estacionario2.sum()

print("Vector Estacionario para el ejercicio 2:")
print(vector_estacionario2)
```

El vector estacionario indica que, a largo plazo, se espera que la proporción de días con climas soleados sea de aproximadamente 60.52%, la proporción de días nublados sea de aproximadamente 21.44%, y la proporción de días lluviosos sea de aproximadamente 18.04%.
Vector Estacionario para el ejercicio 2:
[[[0.60517652]] = Soleado

[[0.21443978]] = Nublados

[[0.1803837]]] = Lluviosos