PARTE 1: Importaciones

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import TransferFunction, tf2zpk, step
from pymoo.core.problem import ElementwiseProblem
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.termination import get_termination
from pymoo.optimize import minimize
import os
```

Estas libreras se usan para:

- numpy: operaciones matemticas y vectores.
- matplotlib: grficas (respuesta al escaln y frente de Pareto).
- scipy.signal: simulacin de funciones de transferencia.
- pymoo: optimizacin evolutiva multiobjetivo con NSGA-II.
- os: crear carpetas para guardar imgenes.

PARTE 2: Modelo de la Aeronave

```
class AircraftTransferFunction:
    def __init__(self, k, c, zeta, omega_n):
```

Define la funcin de transferencia del sistema longitudinal de la aeronave. Los parmetros de entrada son:
- k: ganancia del sistema.
- c: ubicacin del cero.
- zeta: amortiguamiento.
- omega_n: frecuencia natural.

```
self.num = [self.k, self.k * self.c]      # k*s + kc
self.den = [1, 2 * self.zeta * self.omega_n, self.omega_n ** 2]  # s + 2s +
self.system = TransferFunction(self.num, self.den)
```

Aqu se crea el modelo simblico con scipy.signal.TransferFunction.

Mtodo 1: analyze_poles_zeros

```
def analyze_poles_zeros(self):
    zeros, poles, _ = tf2zpk(self.num, self.den)
```

Convierte la funcin de transferencia a polos y ceros.

```
for p in poles:
    damping_ratio = -np.real(p) / np.abs(p)
    natural_freq = np.abs(p)
```

Calcula:

- Amortiguamiento = -Re(p) / |p|
- Frecuencia natural n = |p|

Clasifica el tipo de modo dinmico segn la frecuencia:

- Fugoide si n < 0.5
- Periodo corto si n > 1.5
- Indeterminado en caso contrario

Mtodo 2: plot_step_response

```
def plot_step_response(self, label):
    t, y = step(self.system)
```

Simula la respuesta al escaln.

```
output_dir = "respuestas_escalon"
os.makedirs(output_dir, exist_ok=True)
plt.savefig(f"{output_dir}/step_response_{label}.png")
```

Guarda la figura con nombre personalizado dentro de una carpeta respuestas_escalon.

PARTE 3: Problema de Optimizacin

```
class AircraftOptimizationProblem(ElementwiseProblem):
```

Esta clase define el problema de optimizacin multiobjetivo.

xl=np.array([0.5, 0.1, 0.01, 0.1])

xu=np.array([5.0, 2.0, 1.0, 5.0])

Lmites inferiores y superiores de bsqueda:

- k: 0.5 a 5
- c: 0.1 a 2
- zeta: 0.01 a 1
- omega_n: 0.1 a 5

Mtodo _evaluate: funcin objetivo

```
def _evaluate(self, x, out, *args, **kwargs):
    k, c, zeta, omega_n = x
```

Se toma un vector x con los 4 parmetros.

```
model = AircraftTransferFunction(k, c, zeta, omega_n)
poles_data = model.analyze_poles_zeros()
avg_freq = np.mean([p["natural_freq"] for p in poles_data])
avg_damping = np.mean([p["damping_ratio"] for p in poles_data])
```

Se calcula:

- avg_freq: promedio de frecuencia natural.
- avg_damping: promedio de amortiguamiento.

```
out["F"] = [avg_freq, -avg_damping]
```

Los objetivos son:

- Minimizar la frecuencia natural (modo rpido es deseable).
- Maximizar el amortiguamiento (pero se minimiza el negativo porque pymoo minimiza todo).

PARTE 4: Ejecutar el algoritmo

```
problem = AircraftOptimizationProblem()
```

```
algorithm = NSGA2(pop_size=40)
termination = get_termination("n_gen", 25)
```

Configura:

- 40 individuos por generacin
- 25 generaciones

```
res = minimize(problem,
               algorithm,
               termination,
               seed=1,
               verbose=True)
```

Corre el algoritmo NSGA-II.

Frente de Pareto

```
F = res.F
plt.scatter(F[:, 0], -F[:, 1], c="blue")
```

Grafica las soluciones ptimas (compromiso entre frecuencia baja y amortiguamiento alto).

PARTE 5: Evaluacin de las mejores soluciones

```
top_indices = np.argsort(F[:, 0])[:5]
```

Selecciona las 5 soluciones con menor frecuencia natural.

```
for i, idx in enumerate(top_indices):
    ...
    model = AircraftTransferFunction(k, c, zeta, omega_n)
    analysis = model.analyze_poles_zeros()
    ...
    model.plot_step_response(label)
```

Para cada una:

- Imprime sus polos y clasificacin.
- Genera y guarda la respuesta al escaln.

Conclusin

Este cdigo simula, analiza y optimiza automticamente configuraciones dinmicas de un sistema aeronutico longitudinal y clasifica sus modos de respuesta usando NSGA-II. Puedes adaptarlo fcilmente para distintas funciones objetivo o condiciones dinmicas.