Examen

```
%load_ext autoreload
```

Cambios

- Se utiliza el metodo eliminación gaussiana modificado, solamente para escalonar la matriz.
- Se considera el intercambio de fila, que cambia el signo del determinante.
- Después se multiplica los numeros que haya en la diagonal.

```
import numpy as np


def eliminacion_gaussiana_det(A: np.ndarray) -> tuple[np.ndarray, int]:
    """Realiza la eliminación Gaussiana para triangular una matriz y calcula el número de int

    ## Parameters

    ``A``: Matriz cuadrada de tamaño n x n.

    ## Return

    ``A``: Matriz triangular superior obtenida después de la eliminación Gaussiana.
    ``num_intercambios``: Número de intercambios de filas realizados (para ajustar el signo d
    """
    if not isinstance(A, np.ndarray):
        A = np.array(A, dtype=float)
    assert A.shape[0] == A.shape[1], "La matriz A debe ser cuadrada."

    n = A.shape[0]
    num_intercambios = 0  # Contador de intercambios de filas
```

```
    for i in range(n - 1):
        p = None
        for pi in range(i, n):
            if A[pi, i] != 0:
                p = pi
                break

        if p is None:
            raise ValueError("La matriz es singular. Det = 0.")

        if p != i:
            A[[i, p]] = A[[p, i]]
            num_intercambios += 1

        for j in range(i + 1, n):
            m = A[j, i] / A[i, i]
            A[j, i:] -= m * A[i, i:]

    return A, num_intercambios


def calc_determinante(A: list[list[float]]) -> float:
    """Calcula el determinante de una matriz cuadrada usando eliminación Gaussiana.

    ## Parameters

    ``A``: Matriz cuadrada de tamaño n x n.

    ## Return

    ``detA``: Determinante de la matriz A.
    """
    A_triang, num_intercambios = eliminacion_gaussiana_det(A)

    detA = 1
    for i in range(A_triang.shape[0]):
        detA *= A_triang[i, i]

    detA *= (-1) ** num_intercambios

    return detA
```

Ejercicio 1

```
A1 = [
    [-4, 2, -4, -4, 1, 2, 5, 3, 5, 1],
    [1, 0, 4, 3, 0, -2, 3, 0, 1, 5],
    [5, 5, -4, 5, -4, 2, 2, 2, 4, 4],
    [-1, 3, 4, -1, -4, 0, 5, 0, 0, 5],
    [4, 1, 4, 2, 0, 0, 3, -1, 0, 2],
    [2, -2, 1, -1, -2, -3, 2, -2, 4, -1],
    [3, -2, -3, -2, -1, -3, 5, -1, 5, 0],
    [3, 4, -3, 3, -2, 2, -4, -4, 1, 5],
    [-4, 0, 3, 3, -3, -2, -2, 0, 5, -4],
    [-2, 4, 4, -2, -1, 1, 5, -1, 3, -3],
]
calc_determinante(A1)
```

```
np.float64(9912776.000000015)
```

Ejercicio 2

```
A2 = [
    [2, 2, 4, 5, -2, -3, 2, -2],
    [-1, -1, 3, 2, 1, 1, -4, 4],
    [2, 5, -3, -3, -2, 2, 5, 3],
    [-2, -4, 0, 1, -1, 5, -4, -1],
    [1, -2, -1, 5, 5, 2, 1, -2],
    [5, 4, 0, 3, 4, -1, -3, -2],
    [4, -4, 1, 2, 3, 3, -1, 3],
    [-2, 1, -3, 0, 5, 4, 4, -4],
]
calc_determinante(A2)
```

```
np.float64(2341546.000000001)
```

GitHub: Prueba IIB: Determinante - @mateobtw18