En este archivo PDF se nos presenta un problema típico de sistemas de ecuaciones lineales, donde el concesionario vende tres tipos de coches, llamados Tipo A, Tipo B y Tipo C. El objetivo es determinar cuántos coches de cada tipo se vendieron en un período específico basándonos en ciertas relaciones descritas por tres ecuaciones:

1. **Ecuación 1**: $2x + y + z = 9$, que describe cómo las ventas de coches del Tipo A tienen el doble de impacto en el total de ventas combinadas de los tres tipos. En esta ecuación, x, y, y z representan la cantidad de coches vendidos de los tipos A, B y C, respectivamente.
2. **Ecuación 2**: $3x + 2y + 3z = 21$, que refleja cómo los coches del Tipo A y C generan más ingresos que los del Tipo B, basándose en cómo sus ventas impactan las ganancias totales del concesionario.
3. **Ecuación 3**: $x + 4y + 9z = 23$, donde se refleja el uso de recursos, como el espacio de almacenamiento, con énfasis en que los coches del Tipo B ocupan más espacio que los del Tipo A, y los del Tipo C aún más.

Estas ecuaciones se pueden resolver utilizando una técnica estándar de álgebra lineal conocida como **método de matrices**. Aquí es donde entra en juego tu código.

Relación con el Código Proporcionado

El código que escribiste utiliza matrices para resolver un sistema de ecuaciones lineales de forma eficiente. Para hacerlo, primero define una **matriz de coeficientes**, que es análoga a las ecuaciones que describen las ventas de coches en el ejemplo. Luego, resuelve el sistema usando métodos numéricos. Vamos a descomponer las similitudes y el proceso del código en detalle.

1. Matriz de Coeficientes A

python
```
A = np.array([[1.5, 2, 1], [4, 3.5, 2], [2, 5, 10]])
B = np.array([15, 30, 50])
```

Esta matriz A es similar a la matriz que describiría el sistema de ecuaciones del concesionario:

$$A = \begin{pmatrix} 3 & 1 & 1 \\ 3 & 2 & 3 \\ 1 & 4 & 9 \end{pmatrix}$$

En este caso, las filas de la matriz A representan los coeficientes de las variables en cada una de las ecuaciones (x, y, z) para las ventas de los tres tipos de coches.

El vector B es el lado derecho del sistema de ecuaciones, que describe los valores totales que queremos alcanzar con nuestras ventas (en tu código serían las cantidades como 15, 30 y 50 en lugar de 9, 21 y 23).

2. Solución del Sistema

El código usa la función np.linalg.solve() para resolver el sistema:

python
```
X = np.linalg.solve(A, B)
```

Esto equivale a aplicar el método de **inversión de matrices** o **eliminación de Gauss** para encontrar la solución de las variables x, y, z, que representan la cantidad de coches de cada tipo vendidos. La solución nos da las respuestas para el número de coches vendidos, tal como lo haría si resolviéramos el problema del concesionario.

3. Determinante de la Matriz

El determinante de una matriz nos indica si un sistema de ecuaciones tiene una solución única. Si el determinante es cero, significa que el sistema no tiene una solución única o es indefinido (es decir, las ecuaciones son dependientes entre sí).

python
determinante = np.linalg.det(A)

Este cálculo se aplica a la matriz A para verificar si el sistema tiene una solución única. En el contexto del concesionario, el determinante nos diría si es posible calcular de manera precisa cuántos coches de cada tipo se vendieron sin ambigüedades.

4. Número Condicional

El **número condicional** de una matriz evalúa la **estabilidad numérica** de la solución. Si el número condicional es alto, significa que el sistema es muy sensible a pequeños cambios en los coeficientes de la matriz, lo cual podría llevar a errores en los resultados.

En el código, este número se calcula como:

python
condicional = np.linalg.cond(A)

En el ejemplo del concesionario, si el número condicional es alto, podría significar que pequeñas variaciones en los datos de ventas o ganancias harían que las soluciones cambien drásticamente, indicando que el modelo de ventas o de uso de recursos no es muy robusto.

5. Visualización en 3D

Tu código también incluye una visualización 3D de las ecuaciones del sistema, lo que permite observar los planos que representan cada ecuación:

python
ax.plot_surface(X, Y, Z1, color='red', alpha=0.6)
ax.plot_surface(X, Y, Z2, color='green', alpha=0.6)
ax.plot_surface(X, Y, Z3, color='blue', alpha=0.6)

En el contexto del concesionario, esta visualización tridimensional muestra cómo los tres planos (que representan cada una de las ecuaciones) se cruzan en un punto en el espacio. El punto de intersección es la solución del sistema, que nos dice cuántos coches de cada tipo se vendieron. Esta visualización es útil para entender gráficamente cómo interactúan las tres ecuaciones y qué significa su solución.

Conclusión

En resumen, tu código resuelve de manera numérica un sistema de ecuaciones lineales similar al ejemplo del concesionario. El uso de matrices en el código refleja cómo podemos modelar relaciones complejas entre varias variables (como las ventas de diferentes tipos de coches) y encontrar una solución exacta utilizando herramientas de álgebra lineal. Las operaciones que incluiste, como el cálculo del determinante y el número condicional, proporcionan información valiosa sobre la solvencia del sistema y su estabilidad numérica, mientras que la visualización en 3D da una representación gráfica de las soluciones.