Interpolación

Historia de la Interpolación

La **interpolación** es una técnica matemática antigua, utilizada desde los tiempos de los primeros astrónomos griegos. **Hiparco de Nicea** (siglo II a.C.), conocido por ser uno de los pioneros de la trigonometría, fue de los primeros en utilizar técnicas de interpolación para estimar posiciones planetarias entre observaciones reales. Su obra fue continuada por **Ptolomeo**, que en el siglo II d.C. desarrolló más formalmente métodos interpolativos en sus tablas astronómicas, conocidas como el **Almagesto**.

Durante la Edad Media, los matemáticos islámicos como **Al-Biruni** (siglo X) y **Ulugh Beg** (siglo XV) utilizaron métodos de interpolación para construir tablas trigonométricas más precisas, lo cual fue crucial para la astronomía y la navegación de la época.

En Europa, la interpolación tomó un giro más formal con matemáticos como **Johannes Kepler** (siglo XVII), que aplicó la interpolación en sus leyes del movimiento planetario. Posteriormente, **Isaac Newton** desarrolló el **polinomio de interpolación de Newton**, que se convirtió en una herramienta esencial en análisis numérico. **Joseph Louis Lagrange** también contribuyó con su **polinomio de Lagrange**, un enfoque más general para la interpolación polinómica.

Con el tiempo, la interpolación ha evolucionado para abarcar métodos más avanzados, como los **splines** cúbicos, esenciales en el procesamiento de señales, gráficos por computadora y simulaciones físicas.

¿Qué es la Interpolación?

La interpolación es el proceso de **calcular valores intermedios** entre un conjunto de datos conocidos o medidos. Se aplica cuando tenemos un conjunto de puntos $\left(x_i, f\left(x_i\right)\right)$ y queremos estimar el valor de $f(x)$ en un punto intermedio donde no tenemos medición directa.

Definición Formal:

Sea $f(x)$ una función desconocida, y sean $\left(x_0, f\left(x_0\right)\right), \left(x_1, f\left(x_1\right)\right), \ldots, \left(x_n, f\left(x_n\right)\right)$ un conjunto de puntos conocidos. La interpolación busca construir una función $g(x)$ tal que:

$$g\left(x_i\right) = f\left(x_i\right) \forall\, i = 0, 1, \ldots, n$$

Una vez construida $g(x)$, se puede estimar el valor de $f(x)$ en cualquier punto dentro del rango $\left[x_0, x_n\right)$.

Los métodos más comunes son:

1. **Interpolación Lineal**: Utiliza líneas rectas entre puntos consecutivos.
2. **Interpolación Polinómica**: Usa un único polinomio que pasa por todos los puntos conocidos.

3. **Interpolación de Splines**: Utiliza polinomios de bajo grado ajustados por tramos, lo que permite suavidad en la interpolación.

Tipos de Interpolación

1. Interpolación Lineal

Es el método más sencillo, en el cual los puntos conocidos $(x_0, f(x_0))$ y $(x_1, f(x_1))$ se conectan mediante una línea recta. La fórmula para estimar el valor de $f(x)$ entre x_0 y x_1 es:

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

Este método es rápido y computacionalmente simple, aunque no siempre es muy preciso si la función subyacente no es lineal.

Ventajas:

- Fácil de implementar y computar.
- Muy eficiente para datos que se comportan de manera aproximadamente lineal entre puntos.

Desventajas:

- La función interpolada no es suave; presenta ángulos en los puntos conocidos.
- Puede ser muy impreciso si los puntos conocidos no están cercanos o si la función subyacente es no lineal.

Ejemplo: Dado los puntos $(2, 4)$ y $(5, 10)$, deseamos estimar el valor de $f(x)$ para $x = 3$. Usamos la fórmula:

$$f(3) = 4 + \frac{10 - 4}{5 - 2}(3 - 2) = 4 + 2 = 6$$

2. Interpolación Polinómica

Este método utiliza un **polinomio de grado** n para interpolar $n+1$ puntos conocidos. Por ejemplo, el **polinomio de Lagrange** es una forma de interpolación polinómica que asegura que el polinomio pase por todos los puntos dados.

El polinomio de Lagrange se define como:

$$L(x) = \sum_{i=0}^{n} f(x_i) \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j}$$

El grado del polinomio será n si se tienen $n+1$ puntos.

Ventajas:

- Produce una función suave y continua.
- Se ajusta perfectamente a todos los puntos de datos conocidos.

Desventajas:

- Con muchos puntos, el polinomio puede oscilar significativamente entre ellos, lo que genera el **fenómeno de Runge**, especialmente si los puntos no están distribuidos uniformemente.
- Para grandes conjuntos de datos, calcular los coeficientes del polinomio es costoso computacionalmente.

Ejemplo: Dado un conjunto de tres puntos $(1,1)$, $(2,4)$ y $(3,9)$, podemos construir el polinomio de Lagrange:

$$L(x)=1\cdot\frac{(x-2)(x-3)}{(1-2)(1-3)}+4\cdot\frac{(x-1)(x-3)}{(2-1)(2-3)}+9\cdot\frac{(x-1)(x-2)}{(3-1)(3-2)}$$

Que al simplificarse, dará un polinomio cúbico que pasa por los tres puntos dados.

3. Interpolación de Splines

Los **splines cúbicos** son un método más avanzado de interpolación que utiliza **polinomios de grado tres** (cúbicos) para cada intervalo entre puntos. La principal ventaja de este método es que garantiza que la función interpolada sea suave en todo el intervalo y que las primeras y segundas derivadas sean continuas en los puntos de interpolación.

Un spline cúbico $S(x)$ entre los puntos x_0 y x_1 tiene la forma:

$$S(x)=a(x-x_0)^3+b(x-x_0)^2+c(x-x_0)+d$$

Los coeficientes a, b, c y d se determinan imponiendo las condiciones de que el spline pase por ambos puntos y que las derivadas coincidan en los puntos donde se unen distintos polinomios.

Ventajas:

- Genera curvas suaves y continuas.
- Se evitan las oscilaciones no deseadas que pueden ocurrir en la interpolación polinómica de alto grado.
- Adecuado para grandes conjuntos de datos donde se busca precisión.

Desventajas:

- Más complejo de calcular que la interpolación lineal o polinómica.
- Puede requerir más esfuerzo computacional y memoria, sobre todo para muchos datos.

Ejemplo: Dado los puntos $(1,1)$, $(2,8)$, $(3,27)$ y $(4,64)$, que siguen la función cúbica $y=x^3$, ajustamos splines cúbicos entre cada par de puntos. Por ejemplo, para el intervalo entre $x_1=2$ y $x_2=3$, el spline cúbico tendrá la forma:

$$S(x)=a(x-2)^3+b(x-2)^2+c(x-2)+d$$

Aplicamos las condiciones de que $S(2)=8$ y $S(3)=27$, además de las condiciones de continuidad en las derivadas en $x=2$ y $x=3$, lo que permite calcular los coeficientes.

Métodos más Avanzados

1. **Interpolación por Fragmentos**: Divide los datos en fragmentos más pequeños y aplica interpolaciones diferentes en cada fragmento, a menudo mediante splines.
2. **Interpolación de Fourier**: Utiliza series de Fourier para interpolar funciones periódicas, muy utilizada en procesamiento de señales.

Aplicaciones de la Interpolación

1. **Astronomía**: Estimar posiciones de cuerpos celestes entre observaciones.
2. **Gráficos por Computadora**: Generación de transiciones suaves entre píxeles en imágenes de baja resolución.
3. **Procesamiento de Señales**: Reconstrucción de señales muestreadas, como audio o telecomunicaciones.
4. **Economía y Finanzas**: Predicción de valores intermedios en series temporales.
5. **Ciencia e Ingeniería**: Estimación de datos experimentales y condiciones climáticas en lugares donde no se toman mediciones directas.

Ventajas de la Interpolación

- **Precisión** dentro del rango de los datos conocidos.
- **Simplicidad** en métodos como la interpolación lineal.
- **Flexibilidad**: métodos que se adaptan a diversos tipos de datos.

Desventajas de la Interpolación

- **Extrapolación insegura**: puede generar errores significativos fuera del rango de datos conocidos.
- **Oscilaciones** en polinomios de alto grado.
- **Requiere conocimientos** profundos de los datos para métodos avanzados.

Interpolación Lineal en Varias Dimensiones

Historia y Contexto

La **interpolación** es un concepto matemático con una rica historia que se remonta a la antigüedad. Los matemáticos griegos, como Arquímedes, ya utilizaban métodos de aproximación para resolver problemas relacionados con áreas y volúmenes. Sin embargo, la interpolación moderna comenzó a consolidarse en el siglo XVIII con la introducción de métodos más sistemáticos para estimar valores entre puntos conocidos.

Contribuciones Clave

- **Isaac Newton y Joseph-Louis Lagrange**: En el siglo XVIII, ambos desarrollaron técnicas de interpolación polinómica. El polinomio de interpolación de Lagrange y los métodos de interpolación de Newton permitieron calcular polinomios que pasaban exactamente por conjuntos de puntos, estableciendo las bases para el uso de la interpolación en problemas prácticos.

- **Métodos Numéricos Avanzados**: En el siglo XX, con el auge de las computadoras, se comenzaron a desarrollar métodos numéricos más complejos, como la interpolación spline y la interpolación bilineal, que son fundamentales en la modelización de datos en múltiples dimensiones. Estos métodos son ampliamente utilizados en gráficos por computadora, análisis de datos y simulaciones científicas.

Aplicaciones en Inteligencia Artificial y Big Data

La interpolación lineal en varias dimensiones ha encontrado un papel fundamental en campos como la **inteligencia artificial (IA)** y el **big data**. Algunas de sus aplicaciones más notables incluyen:

1. **Modelado Predictivo**: La interpolación se utiliza en modelos de regresión, donde se estima el valor de una variable dependiente basándose en una o más variables independientes. Esto es esencial en el aprendizaje automático, donde se busca hacer predicciones basadas en datos históricos.

2. **Gráficos por Computadora**: En la renderización de gráficos, la interpolación permite suavizar transiciones entre diferentes estados visuales, asegurando que las animaciones y los modelos tridimensionales sean visualmente atractivos y coherentes.

3. **Reconocimiento de Patrones**: Los algoritmos de aprendizaje automático, como los k-vecinos más cercanos (KNN), utilizan interpolación para hacer predicciones basadas en la proximidad de puntos en un espacio multidimensional. Aquí, la interpolación ayuda a clasificar datos no vistos basándose en ejemplos conocidos.

4. **Análisis Espacial y Geoespacial**: La interpolación es fundamental en la geoinformática, donde se requiere estimar valores en un espacio geográfico basado en datos muestrales, como la temperatura o la elevación.

Sistema Algebraico de la Forma $Ax=b$

La interpolación en varias dimensiones se puede formalizar mediante un sistema de ecuaciones lineales de la forma:

$$Ax=b$$

donde:

- A es una matriz que contiene los coeficientes asociados a los valores de función en los puntos conocidos.
- x es un vector que contiene los coeficientes que queremos determinar.
- b es un vector que representa los valores de la función en los puntos conocidos.

Construcción del Sistema

Para construir el sistema $Ax=b$, seguimos estos pasos:

1. **Elegir la Forma del Modelo**: Suponiendo que la función se puede aproximar como un polinomio lineal en tres dimensiones:

$$f(x,y)=a_0+a_1x+a_2y$$

En este caso, los coeficientes a_0, a_1 y a_2 son los valores que buscamos determinar.

2. **Establecer el Sistema de Ecuaciones**: Para un conjunto de puntos, cada punto proporciona una ecuación que se puede organizar en la forma $Ax=b$. Por ejemplo, consideremos los puntos:

   - $P_1(1,1,2)$, donde el valor de la función en $(1,1)$ es 2.
   - $P_2(1,2,3)$, donde el valor de la función en $(1,2)$ es 3.
   - $P_3(2,1,4)$, donde el valor de la función en $(2,1)$ es 4.

   Para cada punto, podemos escribir la ecuación correspondiente en función de los coeficientes a_0, a_1, y a_2. De esta manera, obtenemos el siguiente sistema de ecuaciones:

   - De P_1:
   $$a_0+a_1(1)+a_2(1)=2$$
   - De P_2:
   $$a_0+a_1(1)+a_2(2)=3$$
   - De P_3:
   $$a_0+a_1(2)+a_2(1)=4$$

   Estos se pueden reorganizar en forma matricial como:

$$\begin{bmatrix}1&1&1\\1&2&1\\2&1&1\end{bmatrix}\begin{bmatrix}a_0\\a_1\\a_2\end{bmatrix}=\begin{bmatrix}2\\3\\4\end{bmatrix}$$

   Aquí, la matriz A está compuesta por los coeficientes de las variables y la columna del vector b representa los valores observados de la función en los puntos específicos. Al resolver este sistema, se obtendrán los valores óptimos de los coeficientes a_0, a_1 y a_2 que mejor se ajustan a los puntos dados.

Determinante de la Matriz

Antes de abordar la clasificación de sistemas algebraicos, es crucial entender el concepto de **determinante**. El determinante de una matriz cuadrada es un número escalar que encapsula información fundamental sobre las propiedades de la matriz, particularmente en relación con su invertibilidad y la geometría de los vectores que representa. Se denota comúnmente como $det(A)$ o $|A|$.

Definición

El determinante se puede calcular para matrices cuadradas de cualquier tamaño, pero su interpretación es más sencilla en dimensiones menores. Para matrices de orden 2 y 3, los determinantes se definen de la siguiente manera:

- Para una matriz 2×2:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \text{entonces } det(A) = ad - bc.$$

- Para una matriz 3×3:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \text{entonces } det(A) = a(ei - fh) - b(di - fg) + c(dh - eg).$$

Propiedades Importantes del Determinante

1. **Criterio de Invertibilidad**:
   - **Definición**: El determinante de una matriz cuadrada proporciona una forma rápida de determinar si la matriz es invertible.
   - **Condición**: Si el determinante de una matriz A es diferente de cero ($det(A) \neq 0$), entonces la matriz es invertible. Esto significa que existe otra matriz A^{-1} tal que $AA^{-1} = A^{-1}A = I$, donde I es la matriz identidad.
   - **Implicaciones**: La invertibilidad de la matriz está relacionada con la existencia de una única solución para el sistema de ecuaciones asociado $Ax = b$. En este caso, los vectores columna (o fila) de la matriz son linealmente independientes. Esto implica que ninguna columna puede expresarse como una combinación lineal de las otras, lo que garantiza que no hay redundancias en las ecuaciones.
   - **Ejemplo**: Para la matriz

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix},$$

   tenemos que

$$det(A) = (2)(1) - (1)(1) = 1 \neq 0.$$

   Por lo tanto, A es invertible y el sistema tiene una solución única.

2. **Cálculo del Determinante**:
   - Existen diversas técnicas para calcular determinantes, que varían según el tamaño de la matriz. Para matrices pequeñas, se utilizan métodos directos, mientras que para matrices más grandes, se aplican métodos sistemáticos.
   - **Regla de Sarrus**: Aplicable solo para matrices 2×2 y 3×3, proporciona una manera sencilla de calcular determinantes.

- **Para matrices** 2×2:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Longrightarrow det(A) = ad - bc.$$

- **Para matrices** 3×3: Se pueden sumar las multiplicaciones de las diagonales hacia abajo menos las multiplicaciones de las diagonales hacia arriba.

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \Longrightarrow det(A) = a(ei - fh) - b(di - fg) + c(dh - eg).$$

– **Expansión por Cofactores**: Esta técnica se utiliza para matrices de orden superior. Permite descomponer el cálculo del determinante en submatrices de menor tamaño.

- Si se tiene una matriz A de tamaño $n \times n$, el determinante se puede calcular expandiendo por cualquier fila o columna. Por ejemplo, expandiendo por la primera fila:

$$det(A) = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} det(M_{1j}),$$

donde M_{1j} es la matriz que se obtiene al eliminar la primera fila y la j-ésima columna de A.

3. **Efectos de las Operaciones en el Determinante**:

– Las operaciones realizadas en las filas o columnas de la matriz afectan el valor del determinante de maneras específicas:
  - **Intercambio de Filas/Columnas**: Cambiar dos filas (o columnas) de una matriz cambia el signo del determinante. Por ejemplo, si B se obtiene de A al intercambiar dos filas, entonces $det(B) = -det(A)$.
  - **Multiplicación de Filas/Columnas por un Escalar**: Si se multiplica una fila (o columna) de una matriz por un escalar k, el determinante se multiplica por k. Por ejemplo, si C se obtiene de A multiplicando la primera fila por 3, entonces $det(C) = 3 \cdot det(A)$.
  - **Fila (o Columna) como Múltiplo de Otra**: Si una fila (o columna) es un múltiplo de otra, el determinante es cero. Esto indica que las filas (o columnas) son linealmente dependientes y que el sistema de ecuaciones asociado no tiene solución o tiene infinitas soluciones.
– **Ejemplo de Propiedad de Dependencia**: Para la matriz

$$D = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix},$$

el determinante es

$$det(D) = (1)(4) - (2)(2) = 0,$$

lo que indica que la segunda fila es un múltiplo de la primera, resultando en un sistema incompatible o indeterminado.

Rango de una Matriz

El **rango** de una matriz es un concepto fundamental en álgebra lineal que indica la dimensión del espacio vectorial generado por sus filas o columnas. En otras palabras, el rango de una matriz representa el número máximo de vectores linealmente independientes que se pueden extraer de la matriz. Este concepto es esencial para entender la estructura de los sistemas de ecuaciones lineales.

Definición

El rango de una matriz $ A $, denotado como $ r(A) $, se define como el número de filas o columnas linealmente independientes en la matriz. El rango puede calcularse de las siguientes maneras:

- **Rango de Filas**: Se determina considerando únicamente las filas de la matriz. Esto se logra mediante la transformación de la matriz a su forma escalonada o forma escalonada reducida (RREF) y contando el número de filas no nulas.
- **Rango de Columnas**: De manera similar, el rango se puede calcular considerando únicamente las columnas de la matriz.

Por el **teorema de la equivalencia de rangos**, el rango de filas y el rango de columnas de una matriz son iguales:

$$r(A) = r(A^T)$$

donde $ A^T $ es la transpuesta de la matriz $ A $.

Propiedades Importantes del Rango

1. **Número de Soluciones en Sistemas de Ecuaciones**: El rango de la matriz de coeficientes $ A $ en un sistema de ecuaciones lineales $ Ax = b $ proporciona información crucial sobre la naturaleza de las soluciones:
   - **Sistema Compatible Determinado**: Si $ r(A) = r([A|b]) $ y ambos son iguales al número de incógnitas (es decir, el número de columnas de $ A $), el sistema tiene una única solución.
   - **Sistema Compatible Indeterminado**: Si $ r(A) = r([A|b]) $ pero es menor que el número de incógnitas, el sistema tiene infinitas soluciones. Esto se da porque hay más variables que ecuaciones efectivamente independientes.
   - **Sistema Incompatible**: Si $ r(A) \neq r([A|b]) $, el sistema es inconsistente y no tiene solución. Esto indica que la combinación de las ecuaciones contradice las condiciones establecidas por el vector de términos independientes $ b $.
2. **Relación con el Determinante**:

- Para matrices cuadradas, si el determinante es diferente de cero ($ det(A) \neq 0 $), esto implica que el rango de la matriz es igual al número de filas (o columnas), lo que significa que las filas (o columnas) son linealmente independientes.
- Si el determinante es cero ($ det(A) = 0 $), el rango es menor que el número de filas (o columnas), indicando que hay dependencia lineal entre las filas (o columnas).

3. **Rango Máximo**: El rango de una matriz $ A $ no puede exceder el menor de las dimensiones de la matriz, es decir, $ r(A) \leq \min(m, n) $, donde $ m $ es el número de filas y $ n $ es el número de columnas.

Ejemplo Detallado

Consideremos la matriz:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

Para determinar el rango de esta matriz, seguiremos estos pasos:

1. **Transformación a la Forma Escalonada**: Usamos operaciones elementales para llevar la matriz a su forma escalonada:

   - Restamos dos veces la primera fila de la segunda fila y tres veces la primera fila de la tercera fila:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

2. **Contar las Filas No Nulas**: En esta forma, podemos ver que solo hay una fila no nula. Por lo tanto, el rango de la matriz es $ r(A) = 1 $.

3. **Interpretación del Resultado**: El resultado $ r(A) = 1 $ implica que todos los vectores fila de la matriz son linealmente dependientes. Esto significa que cualquiera de las filas se puede expresar como un múltiplo de la primera fila, lo que indica que la matriz no tiene suficiente información para cubrir un espacio de dimensión mayor que 1.

4. **Relación con Sistemas de Ecuaciones**: Si planteamos el sistema $ Ax = b $ donde $ b = \begin{bmatrix} 1 \ 2 \ 3 \end{bmatrix} $, podemos analizar el sistema:

   - La forma escalonada muestra que hay una ecuación no nula. Si resolvemos, encontraremos que esta ecuación admite múltiples soluciones para los valores de $ x_2 $ y $ x_3 $, lo que hace que el sistema tenga infinitas soluciones (Sistema Compatible Indeterminado).

El rango de una matriz es un concepto esencial que ayuda a comprender la estructura de los sistemas de ecuaciones lineales. Su relación con el determinante y el número de soluciones proporciona una comprensión más profunda sobre la dependencia e independencia de los

vectores en el contexto de las ecuaciones lineales. Al analizar el rango, se pueden inferir características sobre la existencia y unicidad de soluciones en sistemas algebraicos, facilitando la resolución de problemas en diversas aplicaciones de las matemáticas.

Clasificación de Sistemas Algebraicos

La resolución de sistemas de la forma $A\,x=b$ puede resultar en diferentes clasificaciones según la relación entre el número de ecuaciones y el número de incógnitas. Los sistemas pueden clasificarse en:

1. **Sistema Compatible Determinado (SCD)**:
   - **Definición**: Un sistema es compatible determinado cuando tiene exactamente una solución. Esto se da cuando el número de ecuaciones es igual al número de incógnitas, y el rango de la matriz A coincide con el número de incógnitas.
   - **Condiciones**:
     - **Determinante**: $det(A)\neq 0$, lo que implica que las filas o columnas de la matriz A son linealmente independientes.
     - **Rango**: El rango de A debe ser igual al número de incógnitas, es decir, $r(A)=n$, donde n es el número de incógnitas. Esto significa que cada ecuación es esencial para encontrar la solución única.
   - **Resolución**: Para resolver un SCD, se puede utilizar la matriz inversa. Si A es invertible, la solución única se obtiene mediante:
     $$x=A^{-1}b\,.$$
   - **Ejemplo**: Para el sistema:
     $$\begin{bmatrix}1 & 1\\ 2 & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}2\\ 5\end{bmatrix}$$
     Este sistema es compatible y tiene una única solución. Aquí, el determinante es $1(3)-2(1)=1\neq 0$, lo que garantiza que la solución es única. Resolviendo el sistema, encontramos que $x_1=1$ y $x_2=1$.
2. **Sistema Compatible Indeterminado (SCI)**:
   - **Definición**: Un sistema es compatible indeterminado cuando tiene infinitas soluciones. Esto ocurre en situaciones donde hay más incógnitas que ecuaciones, o cuando las ecuaciones son linealmente dependientes.
   - **Condiciones**:
     - **Determinante**: $det(A)=0$, lo que indica que al menos una de las filas o columnas de A es una combinación lineal de las otras.
     - **Rango**: El rango de la matriz de coeficientes A es menor que el número de incógnitas, es decir, $r(A)<n$, donde n es el número de incógnitas. Además, se cumple que el rango de la matriz aumentada $[A\vee b)$ es igual al rango de la matriz de coeficientes, $r(A)=r([A\vee b))$.
   - **Resolución**: En el caso de un SCI, dado que no existe una matriz inversa, se puede utilizar la pseudoinversa para encontrar una solución general. La pseudoinversa se denota como $A^{+¿¿}$ y la solución se puede expresar como:
     $$x=A^{+¿b.¿}$$
   - **Ejemplo**: Considere el sistema:

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

En este caso, las ecuaciones son linealmente dependientes (la segunda ecuación es un múltiplo de la primera), lo que resulta en infinitas soluciones. Aquí, el determinante es 0, lo que indica que el sistema no tiene una solución única. Todas las soluciones están a lo largo de una línea en el espacio de soluciones.

3. **Sistema Incompatible (SI)**:
   - **Definición**: Un sistema es considerado incompatible cuando no tiene solución. Esto sucede cuando las ecuaciones son contradictorias, es decir, no hay valores de las incógnitas que satisfagan simultáneamente todas las ecuaciones.
   - **Condiciones**:
     - **Determinante**: $det(A) = 0$, lo que indica que las filas o columnas de la matriz A son linealmente dependientes.
     - **Rango**: El rango de la matriz de coeficientes A es menor que el rango de la matriz aumentada $[A \vee b)$, es decir, $r(A) < r([A \vee b))$. Esto sugiere que el vector b no puede ser representado como una combinación lineal de las columnas de A.
   - **Ejemplo**: Considere el sistema:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Este sistema es incompatible porque la primera ecuación implica que $x_1 + x_2 = 2$, mientras que la segunda sugiere que $x_1 - x_2 = 0$. Estas dos ecuaciones no pueden ser verdaderas simultáneamente, resultando en un sistema sin solución.

Relación entre el Rango y la Clasificación de Sistemas

El rango de la matriz juega un papel crucial en la clasificación de los sistemas algebraicos.

En este sistema de ecuaciones, cada fila de la matriz A representa una relación entre las variables dependientes e independientes. Resolver el sistema nos permitirá encontrar los coeficientes del polinomio que mejor se ajusta a los datos. Esta relación se vuelve fundamental al aplicar métodos de optimización en aprendizaje automático y en el análisis de datos.

La interpolación lineal en varias dimensiones no solo es una herramienta matemática esencial, sino que también juega un papel crucial en el desarrollo de tecnologías modernas en inteligencia artificial y big data. Al combinar la teoría de interpolación con sistemas algebraicos de la forma $Ax = b$, se pueden abordar problemas complejos en múltiples disciplinas, mejorando nuestra capacidad para modelar, predecir y entender fenómenos en el mundo real.

Dependencia e Independencia Lineal

Definición de Independencia Lineal

Un conjunto de vectores es **linealmente independiente** si ninguno de los vectores en el conjunto puede ser escrito como una combinación lineal de los demás. Formalmente, se dice que los vectores $v_1, v_2, \ldots, v_n$ son linealmente independientes si la única solución a la ecuación:

$$c_1 v_1 + c_2 v_2 + \cdots + c_n v_n = 0$$

es la trivial, donde todos los coeficientes $c_1, c_2, \ldots, c_n$ son cero. Esto implica que cada vector aporta información única y no redundante al espacio vectorial que abarca.

Ejemplo de Independencia Lineal:

Consideremos los vectores en R^3:

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Para determinar si son linealmente independientes, consideremos la combinación lineal:

$$c_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + c_2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + c_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Esto nos da el siguiente sistema de ecuaciones:

1. $c_1 = 0$
2. $c_2 = 0$
3. $c_3 = 0$

Dado que la única solución es $c_1 = c_2 = c_3 = 0$, los vectores v_1, v_2 y v_3 son linealmente independientes.

Definición de Dependencia Lineal

En contraste, un conjunto de vectores se dice que es **linealmente dependiente** si al menos uno de los vectores puede ser expresado como una combinación lineal de otros. Es decir, existe un conjunto de coeficientes no todos cero, tal que:

$$c_1 v_1 + c_2 v_2 + \cdots + c_n v_n = 0$$

Esto significa que hay redundancia en el conjunto de vectores, lo que puede llevar a una falta de soluciones únicas en sistemas de ecuaciones asociadas.

Ejemplo de Dependencia Lineal:

Consideremos los vectores en R^3:

$$w_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, w_2 = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}, w_3 = \begin{bmatrix} 3 \\ 6 \\ 9 \end{bmatrix}$$

Para verificar la dependencia, observemos que el vector w_2 es el doble de w_1 y el vector w_3 es el triple de w_1:

$$w_2 = 2 \cdot w_1, w_3 = 3 \cdot w_1$$

Si intentamos escribir una combinación lineal que iguale a cero:

$$c_1 w_1 + c_2 w_2 + c_3 w_3 = 0$$

Podemos elegir $c_1 = 1$, $c_2 = -\frac{1}{2}$ y $c_3 = -\frac{1}{3}$, lo que resulta en una solución no trivial:

$$1\begin{bmatrix}1\\2\\3\end{bmatrix} - \frac{1}{2}\begin{bmatrix}2\\4\\6\end{bmatrix} - \frac{1}{3}\begin{bmatrix}3\\6\\9\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$$

Dado que hemos encontrado una combinación no trivial que resulta en el vector cero, los vectores w_1, w_2 y w_3 son linealmente dependientes.

La independencia y dependencia lineal son conceptos fundamentales en álgebra lineal que ayudan a entender la estructura de los espacios vectoriales. La capacidad de un conjunto de vectores para ser expresado como combinaciones lineales es clave para analizar sistemas de ecuaciones y determinar su naturaleza y soluciones.

Relación con el Determinante

El determinante de una matriz tiene un papel crucial en la evaluación de la independencia o dependencia lineal de sus columnas o filas:

- **Determinante y Dependencia**:
  - Si A es una matriz cuyas columnas están formadas por los vectores $v_1, v_2, \ldots, v_n$, el determinante de A se utiliza como un indicador de la independencia lineal:
    - **Independencia Lineal**: Si el determinante $det(A) \neq 0$, los vectores son linealmente independientes. Esto implica que la matriz es invertible y que el sistema de ecuaciones asociado $Ax = b$ tiene una única solución.
    - **Dependencia Lineal**: Si $det(A) = 0$, al menos uno de los vectores es linealmente dependiente de los otros. Esto sugiere que la matriz no es invertible y el sistema puede ser compatible indeterminado (infinitas soluciones) o incompatible (sin solución).

Clasificación de Sistemas Algebraicos de la Forma $Ax = b$

La clasificación de los sistemas de ecuaciones de la forma $Ax = b$ está directamente relacionada con la independencia y dependencia lineal, así como con el determinante de la matriz de coeficientes A:

1. **Sistema Compatible Determinado (SCD)**:
   - **Condición**: Este sistema tiene una única solución cuando el determinante de la matriz de coeficientes A es diferente de cero ($det(A) \neq 0$). En este caso, el rango de la matriz coincide con el número de incógnitas.
   - **Interpretación**: Cada ecuación aporta información nueva y relevante al sistema, garantizando que se pueda encontrar una única solución. En aplicaciones prácticas, esto es crucial en situaciones donde se requiere un resultado preciso y específico, como en la ingeniería o la economía.

2. **Sistema Compatible Indeterminado (SCI)**:
   - **Condición**: Este sistema tiene infinitas soluciones cuando $det(A)=0$ y el número de ecuaciones es menor que el número de incógnitas, o las ecuaciones son linealmente dependientes. En este caso, el rango de la matriz es menor que el número de incógnitas.
   - **Interpretación**: La presencia de redundancia en las ecuaciones permite múltiples soluciones, lo que puede ser útil en situaciones de modelado donde se busca una gama de resultados posibles en lugar de un único valor. Por ejemplo, en modelos de predicción, un SCI puede indicar un rango de escenarios posibles.
3. **Sistema Incompatible**:
   - **Condición**: Este sistema no tiene solución cuando $det(A)=0$ y el vector b no se encuentra en el espacio generado por las columnas de A. Esto significa que las ecuaciones son contradictorias, indicando que no hay manera de que todas las ecuaciones sean verdaderas al mismo tiempo.
   - **Interpretación**: En situaciones prácticas, un sistema incompatible puede surgir en problemas de optimización donde se intentan satisfacer restricciones conflictivas, lo que resulta en un resultado imposible.

La dependencia e independencia lineal son conceptos fundamentales en álgebra lineal que afectan la naturaleza de los sistemas de ecuaciones lineales. El determinante proporciona un criterio efectivo para evaluar la independencia lineal y clasificar los sistemas de la forma $Ax=b$. Estos conceptos son esenciales en diversas aplicaciones, desde la resolución de problemas matemáticos hasta el modelado en ingeniería y ciencia de datos. Comprender estas relaciones permite abordar problemas complejos y encontrar soluciones efectivas en múltiples disciplinas.

Explicación detallada de la función `calcular_matriz_a_y_b`

La función `calcular_matriz_a_y_b` tiene como propósito calcular la **matriz A** y el **vector b** a partir de una lista de puntos conocidos. Estos puntos suelen ser representados como tuplas, donde los primeros elementos corresponden a las variables independientes (x1, x2, x3, x4) y el último elemento es el valor de la función (f_val).

Desglose de la función:

1. **Inicialización de listas vacías**:
   - Se crean dos listas vacías, `A` y `b`, que almacenarán la matriz de coeficientes y el vector de resultados, respectivamente.
2. **Recorrido de los puntos conocidos**:
   - Se utiliza un bucle `for` para iterar a través de cada punto en `data_points`. Cada punto es una tupla con el formato `(x1, x2, x3, x4, f_val)`, donde:
     - `x1`, `x2`, `x3`, y `x4` son las variables independientes.
     - `f_val` es el valor de la función en ese punto.
3. **Construcción de la matriz A**:
   - En cada iteración del bucle, se agrega una lista `[1, x1, x2, x3, x4]` a la matriz `A`. Esta lista corresponde a los coeficientes del polinomio en una ecuación lineal. El primer elemento es siempre `1`, lo que permite considerar el término independiente del polinomio.

4. **Construcción del vector b**:
   - En cada iteración, se agrega el valor de la función `f_val` a la lista `b`. Esto representa el lado derecho de la ecuación (el resultado de aplicar la función a los valores `(x1, x2, x3, x4)`).
5. **Retorno de la matriz A y el vector b**:
   - Al finalizar el bucle, las listas `A` y `b` son convertidas a arrays de NumPy para facilitar su uso en cálculos matriciales posteriores, y luego se retornan.

Resultados:

- **Matriz A**: Es una matriz de coeficientes del sistema de ecuaciones. Cada fila de la matriz corresponde a una ecuación basada en uno de los puntos conocidos.
- **Vector b**: Es un vector columna que contiene los valores de la función (f_val) para cada uno de los puntos conocidos.

Utilidad:

Esta función es útil para construir sistemas de ecuaciones lineales a partir de puntos de datos y resolverlos utilizando técnicas algebraicas como la inversión de matrices o la factorización.

```
import numpy as np

def calcular_matriz_a_y_b(data_points):
    """Calcula la matriz A y el vector b a partir de los puntos
conocidos."""
    A = []
    b = []
    for point in data_points:
        x1, x2, x3, x4, f_val = point
        A.append([1, x1, x2, x3, x4])  # Coeficientes del polinomio
        b.append(f_val)                 # Valor de la función
    return np.array(A), np.array(b)
```

Explicación detallada de la función `verificar_determinante`

La función `verificar_determinante` tiene como propósito calcular el determinante de una matriz `A` y utilizar ese valor para clasificar el sistema de ecuaciones asociado. Si el determinante es distinto de cero, el sistema es **compatible determinado (SCD)**. Si el determinante es cero, el sistema es **compatible indeterminado (SCI)** o **incompatible (SI)**, dependiendo del rango.

Desglose de la función:

1. **Verificación de las dimensiones de la matriz `A`**:
   - La función primero comprueba si la matriz `A` es cuadrada, es decir, si el número de filas es igual al número de columnas (`A.shape[0] == A.shape[1]`).
   - Si la matriz no es cuadrada, la función retorna `None` y el valor `False`, indicando que no es posible calcular el determinante y que no se puede clasificar el sistema.
2. **Cálculo del determinante**:

- Si la matriz es cuadrada, se procede a calcular el determinante de `A` usando la función `np.linalg.det(A)` de NumPy.
- El valor del determinante es almacenado en la variable `determinante_A`.

3. **Clasificación del sistema**:
   - Se retorna el valor del determinante junto con un valor booleano. Este booleano es `True` si el determinante es distinto de cero, y `False` si el determinante es igual a cero.
   - Si el determinante es distinto de cero (`determinante_A != 0`), el sistema es **compatible determinado (SCD)**, es decir, tiene una única solución.
   - Si el determinante es igual a cero, el sistema puede ser **compatible indeterminado (SCI)** o **incompatible (SI)**, lo que requiere un análisis adicional del rango de la matriz.

Resultados:

- **Determinante**: El determinante de la matriz `A`. Si el determinante es distinto de cero, el sistema tiene una solución única.
- **Booleano**: Indica si el determinante es distinto de cero (`True`) o no (`False`).

Utilidad:

Esta función es útil para verificar rápidamente si un sistema de ecuaciones lineales tiene una solución única, es indeterminado o incompatible, lo que es clave para muchos problemas en álgebra lineal.

```
def verificar_determinante(A):
    """Verifica el determinante de A y clasifica el sistema."""
    if A.shape[0] == A.shape[1]:
        determinante_A = np.linalg.det(A)
        return determinante_A, determinante_A != 0
    return None, False
```

Explicación detallada de la función `calcular_solucion`

La función `calcular_solucion` está diseñada para resolver un sistema de ecuaciones lineales de la forma $Ax=b$, donde `A` es la matriz de coeficientes y `b` es el vector de términos independientes. Además, la función verifica la precisión de la solución calculada.

Desglose de la función:

1. **Cálculo de la solución base (`x_base`)**:
   - La función utiliza la inversa de la matriz A para calcular la solución del sistema. Esto se logra con la expresión `np.linalg.inv(A) @ b`, que es equivalente a $ \mathbf{x} = A^{-1} \mathbf{b} $.
   - `x_base` contiene el vector solución para el sistema $Ax=b$.
2. **Cálculo de `b_base` (recalculación del vector `b`)**:

- Una vez obtenida la solución base `x_base`, se recalcula el vector `b_base` mediante la multiplicación de la matriz A por el vector `x_base` (`A @ x_base`). El propósito es verificar si la solución satisface la ecuación $Ax \approx b$.
- `b_base` representa el valor de b obtenido al volver a aplicar la solución al sistema.

3. **Cálculo de la norma de la diferencia**:
   - Para evaluar la precisión de la solución, la función calcula la norma de la diferencia entre el vector b original y el vector `b_base` obtenido. Esto se hace usando la función `np.linalg.norm(b - b_base)`.
   - La norma de la diferencia (`norma_diferencia`) mide la magnitud del error entre el valor esperado de b y el valor recalculado con la solución obtenida. Un valor cercano a cero indica que la solución es precisa.
4. **Retorno de valores**:
   - La función retorna tres elementos:
     i. `x_base`: El vector solución base del sistema.
     ii. `b_base`: El vector b recalculado usando la solución.
     iii. `norma_diferencia`: La norma de la diferencia entre el vector b original y `b_base`, que mide la precisión de la solución.

Utilidad:

Esta función es útil en la resolución de sistemas de ecuaciones lineales, proporcionando tanto la solución como una verificación de la precisión de esa solución. Se puede aplicar en contextos donde es fundamental evaluar si los resultados obtenidos son lo suficientemente precisos, como en simulaciones, modelos matemáticos o problemas de ingeniería.

Nota:

- El uso de `np.linalg.inv(A)` implica que la matriz A debe ser invertible (su determinante no debe ser cero). Si A es singular, esta operación fallará.

```
def calcular_solucion(A, b):
    """Calcula la solución base y su verificación."""
    x_base = np.linalg.inv(A) @ b
    b_base = A @ x_base
    norma_diferencia = np.linalg.norm(b - b_base)
    return x_base, b_base, norma_diferencia
```

Explicación detallada de la función `calcular_solucion_perturbada`

La función `calcular_solucion_perturbada` tiene como objetivo calcular la solución perturbada de un sistema de ecuaciones lineales y evaluar las diferencias entre la solución original y la perturbada. Este tipo de análisis es útil para entender cómo pequeñas variaciones en los datos de entrada pueden afectar la solución del sistema.

Desglose de la función:

1. **Parámetros de entrada**:
   - `A`: Matriz de coeficientes del sistema de ecuaciones lineales.

- `x_base`: Vector de la solución base previamente calculada del sistema.
- `b`: Vector de términos independientes original del sistema.
- `perturbacion_factor`: Factor que determina la magnitud de la perturbación aplicada a la solución base (por defecto es 0.01).

2. **Cálculo de la perturbación**:
   - La función calcula una perturbación basada en la solución base `x_base`. Esta perturbación se calcula como el producto del `perturbacion_factor` y `x_base`:
     $$\text{perturbacion} = \text{perturbacion\_factor} \times \text{x\_base}$$
   - Esto permite modificar ligeramente cada componente de la solución base para simular cambios en el sistema.
3. **Cálculo de la solución perturbada (`x_perturbada`)**:
   - Se obtiene la nueva solución perturbada sumando la perturbación a la solución base:
     $$\text{x\_perturbada} = \text{x\_base} + \text{perturbacion}$$
   - Este vector representa una posible variación en la solución original, permitiendo analizar cómo afectan las pequeñas perturbaciones en los datos.
4. **Cálculo del vector `b_perturbada`**:
   - La función recalcula el vector de términos independientes perturbado multiplicando la matriz de coeficientes `A` por la solución perturbada:
     $$\text{b\_perturbada} = A \cdot \text{x\_perturbada}$$
   - Este nuevo vector `b_perturbada` representa la salida del sistema con la solución perturbada.
5. **Cálculo de la norma de la diferencia**:
   - Para evaluar el impacto de la perturbación, se calcula la norma de la diferencia entre el vector original `b` y el vector perturbado `b_perturbada`:
     $$\text{norma\_diferencia\_perturbada} = \| b - \text{b\_perturbada} \|$$
   - Esta norma mide la magnitud del error entre el valor original de `b` y el recalculado `b_perturbada`, proporcionando una indicación de cómo la perturbación ha afectado los resultados.
6. **Retorno de valores**:
   - La función retorna tres elementos:
     i. `x_perturbada`: El vector solución perturbada del sistema.
     ii. `b_perturbada`: El vector de términos independientes recalculado con la solución perturbada.
     iii. `norma_diferencia_perturbada`: La norma de la diferencia entre el vector original `b` y `b_perturbada`, que mide el efecto de la perturbación.

Utilidad:

Esta función es útil en el análisis de sensibilidad y estabilidad de soluciones en sistemas de ecuaciones lineales. Permite evaluar cómo cambios pequeños en la solución inicial pueden afectar los resultados, lo que es crucial en muchas aplicaciones de ingeniería y ciencias aplicadas.

Nota:

- Es importante tener en cuenta que la perturbación se aplica proporcionalmente a la solución base, lo que significa que la magnitud del cambio dependerá de la escala de `x_base`. Un factor de perturbación mayor resultará en cambios más significativos en la solución perturbada.

```
def calcular_solucion_perturbada(A, x_base, b,
perturbacion_factor=0.01):
    """Calcula la solución perturbada y sus diferencias."""
    perturbacion = perturbacion_factor * x_base
    x_perturbada = x_base + perturbacion
    b_perturbada = A @ x_perturbada
    norma_diferencia_perturbada = np.linalg.norm(b - b_perturbada)
    return x_perturbada, b_perturbada, norma_diferencia_perturbada
```

Explicación detallada de la función `imprimir_resultados_solucion`

La función `imprimir_resultados_solucion` se encarga de presentar los resultados de la solución de un sistema de ecuaciones lineales de manera estructurada y clara. Esta función es útil para mostrar la información relacionada con las soluciones base y perturbadas, así como las diferencias entre ellas.

Desglose de la función:

1. **Parámetros de entrada**:
   - `solver_num`: Número identificador de la solución, que permite distinguir entre diferentes soluciones en caso de que se estén analizando múltiples sistemas.
   - `x_base`: Vector que contiene la solución base del sistema de ecuaciones.
   - `b_base`: Vector de términos independientes calculado a partir de la matriz de coeficientes `A` y la solución base `x_base`.
   - `x_perturbada`: Vector que representa la solución perturbada, obtenida tras aplicar una perturbación a la solución base.
   - `b_perturbada`: Vector de términos independientes recalculado a partir de la solución perturbada `x_perturbada`.
   - `norma_diferencia_base`: Norma de la diferencia entre el vector `b_base` y el vector `b` original, que mide el error en la aproximación.
   - `norma_diferencia_perturbada`: Norma de la diferencia entre el vector `b_perturbada` y el vector `b` original, que evalúa el impacto de la perturbación en los resultados.
2. **Impresión de resultados**:
   - La función utiliza `print` para mostrar de forma estructurada la información relacionada con cada solución. A continuación se describen los pasos de impresión:
   - **Impresión de la solución base**:
     - Se muestra el número de la solución seguido del vector de la solución base `x_base`.

- **Impresión del vector `b_base`**:
    - Se imprime el vector de términos independientes calculado `b_base`, mostrando el resultado de la evaluación del sistema con la solución base.
- **Impresión de la solución perturbada**:
    - Se muestra el vector de la solución perturbada `x_perturbada`, que resulta de aplicar una pequeña perturbación a la solución base.
- **Impresión del vector `b_perturbada`**:
    - Se imprime el vector de términos independientes recalculado `b_perturbada`, que es el resultado de evaluar el sistema con la solución perturbada.

3. **Cálculo y presentación de las normas de diferencia**:
    - La función imprime varias normas que permiten evaluar la calidad de la solución:
        - **Diferencia entre la solución base y la perturbada**:
            - Se calcula y se imprime la norma de la diferencia entre `x_base` y `x_perturbada`, lo que indica cómo ha cambiado la solución debido a la perturbación.
        - **Diferencia entre `b_base` y `b_perturbada`**:
            - Se calcula y se muestra la norma de la diferencia entre `b_base` y `b_perturbada`, proporcionando información sobre el efecto de la perturbación en los términos independientes.
        - **Diferencia entre `b_base` y `b` real**:
            - Se imprime la norma de la diferencia entre `b_base` y el vector `b` original, que evalúa la precisión de la solución base.
        - **Diferencia entre `b_perturbada` y `b` real**:
            - Finalmente, se muestra la norma de la diferencia entre `b_perturbada` y el vector `b` original, que evalúa cómo la perturbación ha afectado el resultado.

Utilidad:

Esta función es esencial para la evaluación de la robustez y sensibilidad de la solución de sistemas de ecuaciones lineales. Permite a los usuarios observar las diferencias entre soluciones originales y perturbadas, así como evaluar la precisión de las soluciones en relación con los datos originales.

Consideraciones:

- La impresión estructurada de los resultados facilita la comparación y el análisis de diferentes soluciones, lo que es crucial en aplicaciones donde la precisión es fundamental.
- Se recomienda utilizar esta función en contextos donde se estén realizando análisis de sensibilidad o en situaciones donde se necesite validar la estabilidad de la solución ante perturbaciones.

```
def imprimir_resultados_solucion(solver_num, x_base, b_base,
x_perturbada, b_perturbada, norma_diferencia_base,
norma_diferencia_perturbada):
    """Imprime los resultados de cada solución de manera
estructurada."""
    print(f"\nSolución {solver_num}:")
    print(x_base)
    print("\nb_base calculada:")
    print(b_base)
    print("\nSolución perturbada:")
    print(x_perturbada)
    print("\nb_perturbada calculada:")
    print(b_perturbada)
    print(f"\nNorma de la diferencia entre x_base y x_perturbada (||
x_base - x_perturbada||): {np.linalg.norm(x_perturbada -
x_base):.2e}")
    print(f"Norma de la diferencia entre b_base y b_perturbada (||
b_base - b_perturbada||): {np.linalg.norm(b_base -
b_perturbada):.2e}")
    print(f"Norma de la diferencia entre b_base y b real (||b_base -
b||): {norma_diferencia_base:.2e}")
    print(f"Norma de la diferencia entre b_perturbada y b real (||
b_perturbada - b||): {norma_diferencia_perturbada:.2e}")
```

Explicación detallada de la función `procesar_sistema_determinado`

La función `procesar_sistema_determinado` se encarga de manejar el caso en el que un sistema de ecuaciones lineales es Compatible Determinado (SCD). Esto significa que el sistema tiene una única solución. A continuación se detalla el funcionamiento de la función.

Desglose de la función:

1. **Parámetros de entrada**:
   - `A`: Matriz de coeficientes del sistema de ecuaciones.
   - `b`: Vector de términos independientes que representan el resultado del sistema.
2. **Determinación del tipo de sistema**:
   - La función imprime un mensaje indicando que el determinante de la matriz `A` es distinto de cero, lo que significa que el sistema es Compatible Determinado (SCD).
3. **Cálculo de la solución base**:
   - Se llama a la función `calcular_solucion` con la matriz `A` y el vector `b` para obtener:
     - `x_base`: La solución base del sistema.
     - `b_base`: El vector de términos independientes recalculado a partir de la solución base.
     - `norma_diferencia`: La norma de la diferencia entre `b` y `b_base`, que mide la precisión de la solución.
4. **Cálculo del número de condición**:

- Se calcula la inversa de la matriz `A` utilizando `np.linalg.inv(A)`.
- Se calcula el número de condición de la matriz `A` usando la norma de la matriz y la norma de su inversa:

$$\text{Número de condición} = ||A|| \times ||A^{-1}||$$

- Se imprime el número de condición, lo que proporciona información sobre la estabilidad de la solución en relación a pequeñas perturbaciones en `A`.

5. **Cálculo y presentación de la solución perturbada**:
   - Se llama a la función `calcular_solucion_perturbada` con la matriz `A`, la solución base `x_base`, y el vector `b` para obtener:
     - `x_perturbada`: La solución perturbada, que se obtiene aplicando una pequeña perturbación a `x_base`.
     - `b_perturbada`: El vector de términos independientes recalculado a partir de `x_perturbada`.
     - `norma_diferencia_perturbada`: La norma de la diferencia entre `b` y `b_perturbada`, que mide el impacto de la perturbación.
6. **Impresión de resultados**:
   - Se llama a la función `imprimir_resultados_solucion` para mostrar de manera estructurada todos los resultados, que incluyen la solución base, el vector de términos independientes calculado, la solución perturbada y sus respectivas normas de diferencia.

Utilidad:

Esta función es esencial para procesar sistemas de ecuaciones que son Compatibles Determinados, brindando un análisis detallado de la solución y la estabilidad de la misma. Al calcular y mostrar la solución base y perturbada, se ayuda a entender cómo afectan las perturbaciones a la solución del sistema.

Consideraciones:

- La función asume que se ha verificado previamente que el determinante de `A` es distinto de cero antes de ser llamada, asegurando que el sistema es efectivamente SCD.
- El número de condición calculado es una métrica importante en análisis numérico, ya que indica la sensibilidad del sistema a cambios en los coeficientes.

```
def procesar_sistema_determinado(A, b):
    """Procesa el caso en el que el sistema es Compatible Determinado
(SCD)."""
    print("\nEl determinante de A es distinto de cero. El sistema es
Compatible Determinado (SCD).")

    # Calcular solución base
    x_base, b_base, norma_diferencia = calcular_solucion(A, b)

    # Calcular el número de condición
    inversa_A = np.linalg.inv(A)
    numero_condicion = np.linalg.norm(A) * np.linalg.norm(inversa_A)
    print(f"\nNúmero de condición de A: {numero_condicion:.2e}")
```

```
    # Calcular y mostrar solución perturbada
    x_perturbada, b_perturbada, norma_diferencia_perturbada =
calcular_solucion_perturbada(A, x_base, b)

    imprimir_resultados_solucion("base", x_base, b_base, x_perturbada,
b_perturbada, norma_diferencia, norma_diferencia_perturbada)
```

Explicación detallada de la función `procesar_sistema_indeterminado_o_incompatible`

La función `procesar_sistema_indeterminado_o_incompatible` se encarga de procesar un sistema de ecuaciones lineales que puede ser Incompatible o Compatible Indeterminado. Esto significa que el sistema no tiene solución o tiene infinitas soluciones, respectivamente. A continuación, se detalla el funcionamiento de la función.

Desglose de la función:

1. **Parámetros de entrada**:
   - `A`: Matriz de coeficientes del sistema de ecuaciones.
   - `b`: Vector de términos independientes que representan el resultado del sistema.
2. **Determinación del tipo de sistema**:
   - La función imprime un mensaje indicando que el determinante de la matriz `A` es cero, lo que sugiere que el sistema es Incompatible o Compatible Indeterminado.
3. **Verificación del rango**:
   - La función imprime un mensaje indicando que se verificarán los rangos para clasificar el sistema.
4. **Cálculo de los rangos**:
   - Se calcula el rango de la matriz `A` utilizando `np.linalg.matrix_rank(A)`.
   - Se crea la matriz aumentada `[A|b]` utilizando `np.column_stack((A, b))`, que combina la matriz `A` con el vector `b` como una nueva columna.
   - Se calcula el rango de la matriz aumentada usando `np.linalg.matrix_rank(matriz_ampliada)`.
5. **Impresión de los rangos**:
   - Se imprimen los rangos de `A` y de la matriz aumentada, lo que proporciona información crucial para clasificar el sistema.
6. **Clasificación del sistema**:
   - Se verifica la relación entre los rangos:
     - Si el rango de `A` es igual al rango de la matriz aumentada y el rango de `A` es menor que el número de columnas de `A` (es decir, hay más variables que ecuaciones), entonces se considera que el sistema es Compatible Indeterminado, y se llama a la función `procesar_sistema_indeterminado` para manejar este caso.
     - Si no se cumple la condición anterior, se concluye que el sistema es Incompatible y se imprime un mensaje correspondiente.

Utilidad:

Esta función es esencial para analizar sistemas de ecuaciones que pueden no tener solución o tener infinitas soluciones. Al calcular los rangos y clasificar el sistema, se proporciona información valiosa sobre la naturaleza del mismo.

Consideraciones:

- La función asume que se ha verificado previamente que el determinante de `A` es cero antes de ser llamada, lo que es un requisito para determinar si el sistema es Incompatible o Compatible Indeterminado.
- La correcta identificación del tipo de sistema es crucial en la resolución de sistemas de ecuaciones, ya que determina el enfoque a seguir para encontrar soluciones, si es que las hay.

```
def procesar_sistema_indeterminado_o_incompatible(A, b):
    """Procesa el caso en el que el sistema es Incompatible o
Compatible Indeterminado."""
    print("\nEl determinante de A es cero. El sistema es Incompatible
o Compatible Indeterminado.")
    print("Verificamos el rango para clasificar el sistema.")

    # Calcular rangos
    rango_A = np.linalg.matrix_rank(A)
    matriz_ampliada = np.column_stack((A, b))  # Crea la matriz
aumentada [A|b]
    rango_ampliado = np.linalg.matrix_rank(matriz_ampliada)

    print(f"\nRango de A: {rango_A}")
    print(f"Rango de la matriz ampliada [A|b]: {rango_ampliado}")

    if rango_A == rango_ampliado and rango_A < A.shape[1]:
        procesar_sistema_indeterminado(rango_A, A, b)
    else:
        print("\nEl sistema es Incompatible.")
```

Explicación detallada de la función `procesar_sistema_indeterminado`

La función `procesar_sistema_indeterminado` maneja el caso en el que un sistema de ecuaciones es Compatible Indeterminado (SCI). Este tipo de sistema tiene infinitas soluciones, lo que significa que hay más variables que ecuaciones. A continuación, se explica en detalle cómo funciona esta función.

Desglose de la función:

1. **Parámetros de entrada**:
   - `rango_A`: El rango de la matriz de coeficientes `A`.
   - `A`: Matriz de coeficientes del sistema de ecuaciones.
   - `b`: Vector de términos independientes que representa el resultado del sistema.

2. **Identificación del sistema**:
   - La función imprime un mensaje indicando que el sistema es Compatible Indeterminado (SCI).
3. **Cálculo de la solución particular**:
   - Se utiliza la pseudo-inversa de `A` para calcular una solución particular del sistema. La pseudo-inversa permite encontrar una solución incluso si el sistema tiene más incógnitas que ecuaciones.
   - La pseudo-inversa se calcula con `np.linalg.pinv(A)`.
   - Se calcula la solución base `x_base` multiplicando la pseudo-inversa por el vector `b`:

$$x_{\text{base}} = A^{+¿ \cdot b ¿}$$

Pseudo-inversa

La pseudo-inversa de una matriz, denotada como $A^+ $, es una generalización de la inversa de una matriz que se aplica a matrices que no son cuadradas o que son singulares (es decir, su determinante es cero). La pseudo-inversa se utiliza para encontrar la mejor solución en el sentido de mínimos cuadrados para sistemas de ecuaciones lineales que pueden no tener soluciones únicas. Esto se logra minimizando la suma de los cuadrados de los errores.

4. **Verificación de la solución base**:
   - Se calcula el vector `b_base` mediante la multiplicación de `A` por la solución base:

$$b_{\text{base}} = A \cdot x_{\text{base}}$$

   - Se calcula la norma de la diferencia entre `b` y `b_base`, que mide cuán cerca está la solución calculada de la solución original.
5. **Prueba de estabilidad**:
   - Se realiza una prueba de estabilidad introduciendo una perturbación en la solución base. La función `calcular_solucion_perturbada` se utiliza para calcular una solución perturbada y sus diferencias:
     - `x_perturbada_base`: La solución base perturbada.
     - `b_perturbada_base`: El vector `b` correspondiente a la solución perturbada.
     - `norma_diferencia_perturbada_base`: La norma de la diferencia entre `b` y `b_perturbada_base`.
6. **Impresión de resultados**:
   - La función `imprimir_resultados_solucion` se llama para mostrar los resultados de la solución base, la verificación, y las soluciones perturbadas de manera estructurada.
7. **Generación de múltiples soluciones**:

- Se llama a la función `generar_soluciones_diferentes`, que se encarga de generar y mostrar tres soluciones diferentes del sistema indeterminado. Esto se hace pasando la matriz `A`, el rango de `A`, la solución base y el vector `b`.

Utilidad:

Esta función es fundamental para trabajar con sistemas de ecuaciones que tienen infinitas soluciones. Al calcular una solución particular y evaluar su estabilidad, se proporciona información útil sobre el comportamiento del sistema bajo pequeñas perturbaciones.

Consideraciones:

- La función asume que se ha verificado previamente que el sistema es Compatible Indeterminado y que se han calculado los rangos correspondientes.
- La utilización de la pseudo-inversa es clave para resolver sistemas que no tienen una solución única, lo que la convierte en una herramienta poderosa en álgebra lineal.

```
def procesar_sistema_indeterminado(rango_A, A, b):
    """Procesa el caso donde el sistema es Compatible Indeterminado
(SCI)."""
    print("\nEl sistema es Compatible Indeterminado (SCI).")

    # Utilizamos la pseudo-inversa para encontrar una solución
particular
    pseudo_inversa_A = np.linalg.pinv(A)
    x_base = pseudo_inversa_A @ b

    # Verificación de la solución base
    b_base = A @ x_base  # Definimos b_base como A * x_base
    norma_diferencia_base = np.linalg.norm(b - b_base)

    # Prueba de estabilidad: perturbación del 1% para la solución base
    x_perturbada_base, b_perturbada_base,
norma_diferencia_perturbada_base = calcular_solucion_perturbada(A,
x_base, b)

    # Mostrar resultados de la perturbación
    imprimir_resultados_solucion("base", x_base, b_base,
x_perturbada_base, b_perturbada_base, norma_diferencia_base,
norma_diferencia_perturbada_base)

    # Generar 3 soluciones diferentes
    generar_soluciones_diferentes(A, rango_A, x_base, b)
```

Explicación detallada de la función `generar_soluciones_diferentes`

La función `generar_soluciones_diferentes` se utiliza para crear y mostrar tres soluciones distintas para un sistema de ecuaciones lineales que tiene infinitas soluciones (Compatible Indeterminado). Esto se logra utilizando el espacio nulo de la matriz de coeficientes $ A $.

Desglose de la función:

1. **Parámetros de entrada**:
   - `A`: La matriz de coeficientes del sistema.
   - `rango_A`: El rango de la matriz $ A $.
   - `x_base`: La solución base calculada previamente.
   - `b`: El vector de términos independientes.
2. **Cálculo del espacio nulo**:
   - La función comienza calculando el espacio nulo de la matriz $ A $ utilizando la descomposición en valores singulares (SVD):

$$\text{null\_space} = \text{SVD}(A)[2][rango_A:].T$$

   - Aquí, $ \text{SVD}(A) $ devuelve tres matrices, y el espacio nulo se obtiene de los vectores singulares correspondientes a los valores singulares que son cero (que indican la dirección de la degeneración de la matriz).

     ### Kernel o Espacio Nulo El **kernel** o **espacio nulo** de una matriz $ A $ es el conjunto de todos los vectores $ x $ que satisfacen la ecuación $ A \cdot x = 0 $. En otras palabras, es el espacio de soluciones a la ecuación homogénea asociada a $ A $. Este espacio es fundamental en la teoría de sistemas de ecuaciones lineales, ya que proporciona información sobre la existencia de soluciones libres y la dimensionalidad de las soluciones de un sistema.

3. **Generación de soluciones**:
   - La función verifica si el espacio nulo tiene columnas (lo que indica la existencia de soluciones libres). Si hay al menos una dimensión en el espacio nulo, se generan tres soluciones diferentes:
     - Para cada solución, se generan parámetros libres aleatorios utilizando `np.random.randn(null_space.shape[1])`.
     - La solución particular se calcula sumando la solución base $ x_{\text{base}} $ al producto del espacio nulo y los parámetros libres:

$$\text{solucion\_particular} = x_{\text{base}} + \text{null\_space} \cdot \text{parametros\_libres}$$

4. **Cálculo de $ b $ y diferencias**:
   - Se calcula el vector $ b $ correspondiente a la solución particular:

$$b_{\text{base\_particular}} = A \cdot \text{solucion\_particular}$$

   - Se mide la diferencia entre $ b $ y $ b_{\text{base_particular}} $ utilizando la norma:

$$\text{norma\_diferencia\_base\_particular} = \| b - b_{\text{base\_particular}} \|$$

   - También se calcula la solución perturbada y sus diferencias utilizando la función `calcular_solucion_perturbada`:
     - `x_perturbada_particular`: La solución perturbada.
     - `b_perturbada_particular`: El vector $ b $ correspondiente a la solución perturbada.
     - `norma_diferencia_perturbada_particular`: La norma de la diferencia entre $ b $ y $ b_{\text{perturbada_particular}} $.

5. **Impresión de resultados**:
   - La función `imprimir_resultados_solucion` se llama para mostrar los resultados de cada solución particular, incluyendo:
     - La solución particular $ x_{\text{particular}} $.
     - El vector $ b_{\text{base_particular}} $ calculado.
     - La solución perturbada y su correspondiente $ b $ perturbado.
     - Las normas de las diferencias.

Utilidad:

Esta función es esencial para explorar la naturaleza de los sistemas indeterminados, proporcionando múltiples soluciones que permiten entender mejor el comportamiento del sistema bajo variaciones. Además, resalta la importancia del espacio nulo en la teoría de sistemas lineales.

Consideraciones:

- La función asume que ya se ha verificado que el sistema es Compatible Indeterminado y que se han calculado previamente la matriz $ A $ y su rango.
- La generación de soluciones aleatorias permite obtener un conjunto diverso de posibles resultados, que pueden ser útiles para análisis adicionales o para ilustrar la indeterminación del sistema.

```
def generar_soluciones_diferentes(A, rango_A, x_base, b):
    """Genera y muestra tres soluciones diferentes basadas en el
espacio nulo de A."""
    null_space = np.linalg.svd(A)[2][rango_A:].T  # Espacio nulo de A
    if null_space.shape[1] > 0:
        for i in range(3):
            parametros_libres = np.random.randn(null_space.shape[1])
            solucion_particular = x_base + null_space @
parametros_libres

            # Calcular b_base y b perturbada para la solución
particular
            b_base_particular = A @ solucion_particular
            norma_diferencia_base_particular = np.linalg.norm(b -
b_base_particular)
            x_perturbada_particular, b_perturbada_particular,
norma_diferencia_perturbada_particular =
calcular_solucion_perturbada(A, solucion_particular, b)

            # Imprimir resultados para la solución particular
            imprimir_resultados_solucion(i + 1, solucion_particular,
b_base_particular, x_perturbada_particular, b_perturbada_particular,
norma_diferencia_base_particular,
norma_diferencia_perturbada_particular)
```

Explicación detallada de la función `analizar_puntos`

La función `analizar_puntos` es la función principal que coordina el análisis de un conjunto de puntos conocidos y realiza cálculos relacionados con un sistema de ecuaciones lineales.

Desglose de la función:

1. **Propósito**:
   - La función tiene como objetivo analizar un conjunto de puntos que representan una relación matemática y determinar las características del sistema de ecuaciones asociado.
2. **Parámetros de entrada**:
   - `data_points`: Una lista de puntos conocidos en el formato `(x1, x2, x3, x4, f(x1, x2, x3, x4))`, donde cada punto contiene las coordenadas y el valor de la función en ese punto.
3. **Preparación de matrices**:
   - Se llama a la función `calcular_matriz_a_y_b` para generar la matriz de coeficientes $ A $ y el vector de términos independientes $ b $ a partir de los puntos dados.
   - Esto se hace para establecer la estructura del sistema de ecuaciones lineales que se va a analizar.
4. **Impresión de matrices**:
   - Se imprimen la matriz $ A $ y el vector $ b $ en la consola para que el usuario pueda ver la configuración del sistema.
   - Esta información es crucial para la comprensión de las relaciones lineales entre los puntos.
5. **Verificación del determinante**:
   - La función `verificar_determinante` se llama para calcular el determinante de la matriz $ A $ y determinar si el sistema es Compatible Determinado (SCD) o Incompatible/Compatible Indeterminado.
   - Se imprime el valor del determinante en la consola.
6. **Procesamiento del sistema**:
   - Si el determinante de $ A $ es distinto de cero (`sistema_es_determinado` es verdadero), se llama a la función `procesar_sistema_determinado` para analizar el sistema como Compatible Determinado (SCD).
   - Si el determinante es cero, se llama a la función `procesar_sistema_indeterminado_o_incompatible` para manejar el caso de un sistema Incompatible o Compatible Indeterminado.

Consideraciones:

- La función asume que las funciones auxiliares, como `calcular_matriz_a_y_b`, `verificar_determinante`, `procesar_sistema_determinado`, y `procesar_sistema_indeterminado_o_incompatible`, están implementadas y funcionan correctamente.
- La estructura modular de la función permite una fácil comprensión y mantenimiento del código, al dividir las tareas en funciones más pequeñas y específicas.

Conclusión:

La función `analizar_puntos` es esencial para iniciar el análisis de sistemas de ecuaciones lineales basados en datos conocidos. Proporciona una interfaz clara para preparar los datos, verificar las propiedades del sistema, y decidir el camino adecuado para su procesamiento posterior.

```
def analizar_puntos(data_points):
    """Función principal para analizar los puntos y realizar
cálculos."""
    # Preparar matrices A y b
    A, b = calcular_matriz_a_y_b(data_points)

    # Imprimir las matrices A y b
    print("Matriz A:")
    print(A)
    print("\nVector b:")
    print(b)

    # Verificar el determinante de A
    determinante_A, sistema_es_determinado = verificar_determinante(A)
    print(f"\nDeterminante de A: {determinante_A:.2f}")

    if sistema_es_determinado:
        procesar_sistema_determinado(A, b)
    else:
        procesar_sistema_indeterminado_o_incompatible(A, b)
```

Sistema Compatible Determinado

```
# Definir los puntos conocidos
# Formato: (x1, x2, x3, x4, f(x1, x2, x3, x4))
data_points = [
    (1, 1, 2, 3, 25),
    (1, 3, 2, 4, 35),
    (2, 3, 4, 5, 45),
    (2, 1, 2, 3, 30),
    (3, 4, 5, 5, 60)
]

# Llamar a la función principal con los puntos
analizar_puntos(data_points)
```

```
Matriz A:
[[1 1 1 2 3]
 [1 1 3 2 4]
 [1 2 3 4 5]
 [1 2 1 2 3]
```

```
 [1 3 4 5 5]]

Vector b:
[25 35 45 30 60]

Determinante de A: -4.00

El determinante de A es distinto de cero. El sistema es Compatible
Determinado (SCD).

Número de condición de A: 7.99e+01

Solución base:
[13.75  5.    6.25  3.75 -2.5 ]

b_base calculada:
[25. 35. 45. 30. 60.]

Solución perturbada:
[13.8875  5.05    6.3125  3.7875 -2.525 ]

b_perturbada calculada:
[25.25 35.35 45.45 30.3  60.6 ]

Norma de la diferencia entre x_base y x_perturbada (||x_base -
x_perturbada||): 1.65e-01
Norma de la diferencia entre b_base y b_perturbada (||b_base -
b_perturbada||): 9.15e-01
Norma de la diferencia entre b_base y b real (||b_base - b||): 4.55e-
14
Norma de la diferencia entre b_perturbada y b real (||b_perturbada -
b||): 9.15e-01
```

Sistema Compatible Indeterminado

```python
# Definir los puntos conocidos
# Formato: (x1, x2, x3, x4, f(x1, x2, x3, x4))
data_points = [
    (1, 1, 2, 3, 25),
    (2, 2, 4, 6, 50),
    (3, 3, 6, 9, 75),
    (8, 5, 2, 2, 40),
    (1, 3, 2, 5, 40)
]

# Llamar a la función principal con los puntos
analizar_puntos(data_points)
```

```
Matriz A:
[[1 1 1 2 3]
 [1 2 2 4 6]
 [1 3 3 6 9]
 [1 8 5 2 2]
 [1 1 3 2 5]]

Vector b:
[25 50 75 40 40]

Determinante de A: 0.00

El determinante de A es cero. El sistema es Incompatible o Compatible Indeterminado.
Verificamos el rango para clasificar el sistema.

Rango de A: 4
Rango de la matriz ampliada [A|b]: 4

El sistema es Compatible Indeterminado (SCI).

Solución base:
[2.56153867e-14 1.52696366e+00 2.36225088e+00 2.84876905e+00
 5.13774912e+00]

b_base calculada:
[25. 50. 75. 40. 40.]

Solución perturbada:
[2.58715406e-14 1.54223329e+00 2.38587339e+00 2.87725674e+00
 5.18912661e+00]

b_perturbada calculada:
[25.25 50.5  75.75 40.4  40.4 ]

Norma de la diferencia entre x_base y x_perturbada (||x_base - x_perturbada||): 6.51e-02
Norma de la diferencia entre b_base y b_perturbada (||b_base - b_perturbada||): 1.09e+00
Norma de la diferencia entre b_base y b real (||b_base - b||): 4.55e-14
Norma de la diferencia entre b_perturbada y b real (||b_perturbada - b||): 1.09e+00

Solución 1:
[2.56153867e-14 1.76766792e+00 2.02526491e+00 2.39143096e+00
 5.47473509e+00]

b_base calculada:
[25. 50. 75. 40. 40.]
```

```
Solución perturbada:
[2.58715406e-14 1.78534460e+00 2.04551756e+00 2.41534527e+00
 5.52948244e+00]

b_perturbada calculada:
[25.25 50.5  75.75 40.4  40.4 ]

Norma de la diferencia entre x_base y x_perturbada (||x_base -
x_perturbada||): 6.55e-02
Norma de la diferencia entre b_base y b_perturbada (||b_base -
b_perturbada||): 1.09e+00
Norma de la diferencia entre b_base y b real (||b_base - b||): 4.14e-
14
Norma de la diferencia entre b_perturbada y b real (||b_perturbada -
b||): 1.09e+00

Solución 2:
[2.56153867e-14 1.34671201e+00 2.61460319e+00 3.19124718e+00
 4.88539681e+00]

b_base calculada:
[25. 50. 75. 40. 40.]

Solución perturbada:
[2.58715406e-14 1.36017913e+00 2.64074922e+00 3.22315965e+00
 4.93425078e+00]

b_perturbada calculada:
[25.25 50.5  75.75 40.4  40.4 ]

Norma de la diferencia entre x_base y x_perturbada (||x_base -
x_perturbada||): 6.53e-02
Norma de la diferencia entre b_base y b_perturbada (||b_base -
b_perturbada||): 1.09e+00
Norma de la diferencia entre b_base y b real (||b_base - b||): 3.89e-
14
Norma de la diferencia entre b_perturbada y b real (||b_perturbada -
b||): 1.09e+00

Solución 3:
[2.56153867e-14 1.08816606e+00 2.97656752e+00 3.68248449e+00
 4.52343248e+00]

b_base calculada:
[25. 50. 75. 40. 40.]

Solución perturbada:
[2.58715406e-14 1.09904772e+00 3.00633320e+00 3.71930934e+00
 4.56866680e+00]
```

```
b_perturbada calculada:
[25.25 50.5  75.75 40.4  40.4 ]

Norma de la diferencia entre x_base y x_perturbada (||x_base - x_perturbada||): 6.64e-02
Norma de la diferencia entre b_base y b_perturbada (||b_base - b_perturbada||): 1.09e+00
Norma de la diferencia entre b_base y b real (||b_base - b||): 4.08e-14
Norma de la diferencia entre b_perturbada y b real (||b_perturbada - b||): 1.09e+00
```

Sistema Incompatible

```
# Definir los puntos conocidos
# Formato: (x1, x2, x3, x4, f(x1, x2, x3, x4))
data_points = [
    (1, 1, 2, 3, 25),
    (2, 2, 4, 6, 50),
    (3, 3, 6, 9, 75),
    (4, 4, 8, 12, 99),
    (5, 5, 10, 15, 100)
]

# Llamar a la función principal con los puntos
analizar_puntos(data_points)
```

```
Matriz A:
[[ 1  1  1  2  3]
 [ 1  2  2  4  6]
 [ 1  3  3  6  9]
 [ 1  4  4  8 12]
 [ 1  5  5 10 15]]

Vector b:
[ 25  50  75  99 100]

Determinante de A: 0.00

El determinante de A es cero. El sistema es Incompatible o Compatible Indeterminado.
Verificamos el rango para clasificar el sistema.

Rango de A: 2
Rango de la matriz ampliada [A|b]: 3

El sistema es Incompatible.
```

Sistema Compatible Determinado (Inestable)

```
# Definir los puntos conocidos
# Formato: (x1, x2, x3, x4, f(x1, x2, x3, x4))
data_points = [
    (0.001, 10, 200, 30, 250),
    (10, 300, 0.002, 4, 350),
    (20, 300, 40, 50, 450),
    (20, 10, 0.002, 30, 300),
    (30, 0.04, 5, 0.0005, 650)
]

# Llamar a la función principal con los puntos
analizar_puntos(data_points)
```

```
Matriz A:
[[1.e+00 1.e-03 1.e+01 2.e+02 3.e+01]
 [1.e+00 1.e+01 3.e+02 2.e-03 4.e+00]
 [1.e+00 2.e+01 3.e+02 4.e+01 5.e+01]
 [1.e+00 2.e+01 1.e+01 2.e-03 3.e+01]
 [1.e+00 3.e+01 4.e-02 5.e+00 5.e-04]]

Vector b:
[250 350 450 300 650]

Determinante de A: -52180608.23

El determinante de A es distinto de cero. El sistema es Compatible
Determinado (SCD).

Número de condición de A: 1.00e+03

Solución base:
[-14.40558244  21.82414432   0.54387644   1.93232463  -4.25066443]

b_base calculada:
[250. 350. 450. 300. 650.]

Solución perturbada:
[-14.54963827  22.04238576   0.54931521   1.95164788  -4.29317108]

b_perturbada calculada:
[252.5 353.5 454.5 303.  656.5]

Norma de la diferencia entre x_base y x_perturbada (||x_base -
x_perturbada||): 2.66e-01
Norma de la diferencia entre b_base y b_perturbada (||b_base -
b_perturbada||): 9.49e+00
Norma de la diferencia entre b_base y b real (||b_base - b||): 8.53e-
14
```

```
Norma de la diferencia entre b_perturbada y b real (||b_perturbada - b||): 9.49e+00
```

Extrapolación

La extrapolación es el proceso de **estimar valores fuera del rango de datos conocidos**. Dado un conjunto de datos $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$, la extrapolación busca predecir el valor de $f(x)$ para $x < x_0$ o $x > x_n$. A diferencia de la interpolación, donde se supone que los datos entre puntos están bien representados por una función conocida, la extrapolación conlleva un mayor riesgo de error, ya que se basa en supuestos sobre el comportamiento de la función fuera del rango observado.

Definición Formal

Sea $f(x)$ una función desconocida definida por puntos conocidos $(x_0, f(x_0)), \ldots, (x_n, f(x_n))$. La extrapolación busca construir una función $g(x)$ tal que $g(x) = f(x)$ para $x < x_0$ o $x > x_n$. Al extrapolar, se asume que la función $f(x)$ sigue el mismo comportamiento que dentro del rango de datos observados, lo cual no siempre es cierto, especialmente en contextos donde se presentan cambios abruptos o no lineales.

Tipos de Extrapolación

Existen varios métodos para realizar la extrapolación, muchos de los cuales son extensiones de los métodos de interpolación:

1. **Extrapolación Lineal**: Utiliza una línea recta para proyectar el comportamiento de los datos fuera del rango observado.
2. **Extrapolación Polinómica**: Utiliza un polinomio ajustado a los puntos conocidos para extrapolar más allá del rango observado.
3. **Extrapolación mediante Modelos**: Utiliza modelos matemáticos o físicos específicos que describen el comportamiento de los datos para hacer predicciones más precisas.

Métodos Comunes de Extrapolación

1. Extrapolación Lineal

Este es el método más sencillo y común de extrapolación. Se basa en la suposición de que la relación entre los datos sigue siendo lineal más allá del rango observado. Dado un conjunto de puntos $(x_0, f(x_0))$ y $(x_1, f(x_1))$, la fórmula para extrapolar más allá de x_1 sería:

$$f(x) = f(x_1) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1)$$

Este método es útil cuando los datos conocidos tienen una relación aproximadamente lineal, pero puede ser inexacto si los datos son no lineales fuera del rango observado.

Ventajas:

- Muy sencillo de implementar.
- Útil cuando los datos tienen una relación casi lineal.

Desventajas:

- Poco preciso si la función subyacente es no lineal.
- Rápidamente pierde precisión a medida que nos alejamos del rango de los datos conocidos.

Ejemplo: Dado los puntos $(2,4)$ y $(5,10)$, si deseamos extrapolar el valor de $f(x)$ para $x=6$, usamos la fórmula:

$$f(6)=10+\frac{10-4}{5-2}(6-5)=10+2=12$$

2. Extrapolación Polinómica

La extrapolación polinómica se basa en ajustar un polinomio a los datos conocidos y utilizar este polinomio para estimar valores fuera del rango de datos. Este método es más flexible que la extrapolación lineal, ya que puede capturar relaciones no lineales entre los datos.

Para extrapolar utilizando un polinomio de grado n, se utiliza el polinomio interpolador de Lagrange o el método de mínimos cuadrados. La fórmula general de un polinomio de grado n es:

$$P(x)=a_n x^n+a_{n-1}x^{n-1}+\ldots+a_1 x+a_0$$

donde los coeficientes a_i se determinan a partir de los puntos conocidos.

Ventajas:

- Puede modelar relaciones no lineales en los datos.
- Proporciona una mejor aproximación en comparación con la extrapolación lineal si los datos tienen un comportamiento polinómico.

Desventajas:

- Puede sobreajustar los datos si se utilizan polinomios de grado demasiado alto.
- Aún presenta riesgos de inexactitud al extrapolar fuera del rango de datos, especialmente si los datos no siguen un patrón polinómico.

Ejemplo: Dado los puntos $(1,1)$, $(2,4)$ y $(3,9)$, podemos ajustar un polinomio cuadrático $P(x)=a x^2+b x+c$. Si ajustamos el polinomio y encontramos que $P(x)=x^2$, podemos extrapolar para $x=4$:

$$P(4)=4^2=16$$

3. Extrapolación Mediante Modelos

En algunos casos, el comportamiento de los datos puede ser modelado mediante ecuaciones físicas o matemáticas que describen con precisión la naturaleza del fenómeno subyacente. Por

ejemplo, las **leyes de Newton** en física o las **curvas de crecimiento** en biología y economía son modelos que permiten extrapolaciones precisas en función de los parámetros del modelo.

Este tipo de extrapolación es mucho más fiable que la lineal o polinómica cuando se tiene un buen conocimiento del fenómeno subyacente, pero requiere suposiciones más estrictas sobre la forma de la función que describe los datos.

Ventajas:

- Muy preciso si se conoce bien el modelo subyacente.
- Captura relaciones no lineales y comportamientos complejos.

Desventajas:

- Requiere un buen conocimiento del fenómeno que se está modelando.
- Depende de que el modelo subyacente sea correcto, lo cual no siempre es posible.

Ejemplo: En física, la extrapolación de la velocidad de un objeto en caída libre se puede modelar con la ecuación:

$$v = v_0 + g t$$

donde v_0 es la velocidad inicial, g es la aceleración debida a la gravedad, y t es el tiempo. Si conocemos la velocidad en dos momentos de tiempo, podemos extrapolar su valor en momentos posteriores, confiando en que el modelo es correcto.

Métodos más Avanzados

Además de estos métodos básicos, hay técnicas más avanzadas como:

1. **Extrapolación Logarítmica**: Utiliza transformaciones logarítmicas para datos que muestran un crecimiento exponencial o comportamiento logarítmico.
2. **Extrapolación de Fourier**: Se utiliza para extrapolar funciones periódicas o cíclicas utilizando series de Fourier.

Aplicaciones de la Extrapolación

1. **Climatología y Meteorología**: Extrapolar tendencias de temperatura o precipitaciones futuras basándose en datos históricos.
2. **Economía y Finanzas**: Predicciones del crecimiento económico o de precios futuros basados en modelos y datos históricos.
3. **Física**: Predicción del comportamiento de sistemas físicos más allá de los límites observados, como la extrapolación de trayectorias de partículas.
4. **Biología**: Extrapolación de curvas de crecimiento o poblacionales para estimar el tamaño de una población en el futuro.
5. **Ingeniería**: Predicción del comportamiento de materiales bajo condiciones extremas (más allá de las condiciones de prueba).

Ventajas de la Extrapolación

- **Predicción fuera del rango de datos**: Permite estimar valores futuros o pasados en casos donde no hay datos disponibles.
- **Flexibilidad de métodos**: Hay diferentes métodos (lineales, polinómicos, modelos) que pueden adaptarse a la naturaleza del problema.
- **Uso en modelos complejos**: En situaciones donde se entiende bien el fenómeno subyacente, la extrapolación puede proporcionar predicciones útiles y precisas.

Desventajas de la Extrapolación

- **Incertidumbre**: Aumenta el riesgo de error ya que se hace una suposición sobre el comportamiento futuro que puede no ser válida.
- **Dependencia del modelo**: La precisión de la extrapolación depende en gran medida de la calidad del modelo utilizado y de su ajuste a los datos.
- **Límites de confianza**: Las extrapolaciones muy alejadas de los datos observados pueden ser extremadamente inexactas.

Principales Diferencias entre Extrapolación e Interpolación

Característica	Interpolación	Extrapolación
Rango de Datos	Dentro del rango de datos conocidos	Fuera del rango de datos conocidos
Precisión	Generalmente más precisa	Mayor riesgo de error
Métodos Utilizados	Suele usar métodos de ajuste de curva	Puede usar métodos de ajuste o modelos
Suposiciones	Menos suposiciones sobre el comportamiento	Más suposiciones sobre el comportamiento
Aplicaciones Comunes	Rellenar datos faltantes	Predecir tendencias futuras