DERIVACIÓN EN R POR EL MÉTODO DE LOS COEFICIENTES INDETERMINADOS Y LA FUNCIÓN diff_indeterminados

El método de coeficientes indeterminados es una técnica general para aproximar derivadas de una función en un punto específico utilizando una combinación lineal de los valores de la función en varios puntos cercanos. Este método es fundamental en el cálculo numérico y sirve para derivadas de cualquier orden, incluso cuando no se conoce la forma explícita de la función.

Idea general

La idea central es construir una fórmula de derivación que exprese una derivada de f(x) (de cualquier orden n) en un punto x0 como una combinación ponderada de los valores de f(x) en puntos circundantes. Es decir:

$$f^{(n)}(x_0) \approx \sum_{i=1}^{k} c_i \cdot f(x_i),$$

donde:

- f(n)(x0) es la n-ésima derivada que queremos aproximar en x0
- ci son los coeficientes que queremos determinar,
- f(xi) son los valores de la función en puntos cercanos x1,x2,…,xk.

Para poder aplicar esta formula debemos conocer los ci, por lo que emplearemos la función **diff_indeterminados,** detallada a continuación:

Definición de la función y validaciones iniciales

```
diff_indeterminados <- function(s_points, x_0, orden_diff) {

  # s_points: Vector de puntos de soporte alrededor del punto de interés.

  # x_0: Punto en el que se desea calcular la derivada.

  # orden_diff: Orden de la derivada que se desea calcular.
```

Validación de puntos distintos:

```
abs_diff <- abs(s_points - x_0)

filtered_diff <- abs_diff[abs_diff != 0]
```

- **abs_diff:** Calcula las diferencias absolutas entre cada punto en s_pointsx0. Si x0 es igual a uno de los puntos de s_points, la diferencia será 0.
- **filtered_diff:** Filtra las diferencias distintas de 0, eliminando los puntos que coinciden con x0.

Verificación de x0 único:

```
if (length(filtered_diff) == 0) {
  stop()
}
```

- Si todos los puntos en s_points son iguales a x0, la función termina con un error porque no es posible calcular derivadas.

Determinación de hhh y validación del orden

```
h <- min(filtered_diff)
n_points = length(s_points)
```

- h: Calcula el paso más pequeño (distancia mínima) entre x0 y los puntos de soporte. Esto es esencial para normalizar las distancias.
- n_points: Número de puntos en el vector s_points.

```
if (orden_diff >= n_points) {
  stop(
    paste(
      "El orden de la derivada (",
      orden_diff,
      ") debe ser menor que el número de puntos (",
      n_points,  ").  ))
}
```

- Asegura que el orden de la derivada (n) sea menor que el número de puntos (k), ya que no se puede calcular una derivada de orden superior al número de puntos disponibles.

Cálculo de los valores normalizados (θ\thetaθ)

```
thetas <- (s_points - x_0) / h
```

- `thetas`: Normaliza los puntos de soporte s_points dividiendo la distancia de cada punto a x0 por el paso h. Esto facilita la construcción de la matriz de Vandermonde.

Construcción de la matriz de Vandermonde

```
vandermonde_matrix <- matrix(0, nrow = n_points, ncol = n_points)

for (i in 1:n_points) {

  for (j in 1:n_points) {

    vandermonde_matrix[i, j] <- thetas[j] ^ (i - 1)  }}
```

- vandermonde_matrix: Una matriz donde cada fila i contiene las potencias de los valores de θjhasta i−1.

Vector del lado derecho

```
diff_equal <- numeric(length = n_points)

diff_equal[orden_diff + 1] <- factorial(orden_diff)
```

- diff_equal: Un vector donde todos los elementos son 0, excepto el índice correspondiente a la derivad, que contiene n! (el factorial del orden de derivada). Este vector establece las condiciones para la derivada deseada.

Resolución del sistema de ecuaciones

```
coff_indet_diff <- solve(vandermonde_matrix, diff_equal * (1 / h ^ orden_diff))
```

- solve(vandermonde_matrix, ...): Resuelve el sistema lineal A · c=b
- coff_indet_diff: Los coeficientes.

Devolución de resultados

```
return(coff_indet_diff)
```

- Devuelve un vector de coeficientes

CÁLCULO DE LA APROXIMACIÓN A LA DERIVADA FINAL.

Simplemente realizamos el sumatorio de la fórmula inicial con la imagen de los spoint, y.

```
funcion_derivada <- function(coff_indet_diff, y) {
  sum = 0  # Inicializamos el resultado en 0
  n = length(coff_indet_diff)  # Determinamos el número de coeficientes

  # Realizamos el sumatorio
  for (i in 1:n) {
    sum = sum + (coff_indet_diff[i] * y[i])  # Suma de coeficientes * valores de y
  }

  # Retornar el valor calculado
  return(sum)
}
```

Finalmente, a la hora de llamar a la función_derivada, obtendremos el valor de la aproximación deseada.