ALGORITMOS DE DERIVACIÓN

La derivación polinómica tiene múltiples aplicaciones y utilidades en diversos campos de las matemáticas y las ciencias aplicadas. Algunas de las principales utilidades de la derivada de un polinomio incluyen:

1. **Tasa de cambio**: La derivada de un polinomio en un punto dado nos dice cómo cambia el valor de la función con respecto a la variable independiente (generalmente x) en ese punto. Esto es fundamental para entender el comportamiento de la función, como la velocidad de un objeto en movimiento o el ritmo de crecimiento de una población.
2. **Análisis de comportamiento asintótico**: La derivada también puede ayudar a entender el comportamiento a gran escala de la función, especialmente cuando se analiza el comportamiento de un polinomio cuando x tiende a infinito. Esto es útil en el estudio de las funciones y sus límites, especialmente en la teoría de la aproximación y el cálculo de límites.
3. **Solución de problemas físicos**: En física, la derivación de polinomios es fundamental para modelar y resolver problemas relacionados con el movimiento, la aceleración y otras variables que cambian con respecto al tiempo. Por ejemplo, la velocidad es la derivada de la posición con respecto al tiempo, y la aceleración es la derivada de la velocidad.

1. DERIVACIÓN DEL POLINOMIO DE LAGRANGE

La derivación del **polinomio de Lagrange** sirve para obtener una aproximación de la **tasa de cambio** de la función interpolada entre los puntos dados. Al derivar el polinomio de Lagrange, se puede calcular la **derivada numérica** de la función en cualquier punto dentro del intervalo de interpolación, lo que es útil en varios contextos:

→ Polinomio de Lagrange

El **polinomio de Lagrange** es una fórmula utilizada para interpolar un conjunto de puntos (x_0, y_0), (x_1, y_1), ..., ($x\square$, $y\square$). Este polinomio pasa por todos los puntos dados, y se construye como una combinación lineal de polinomios básicos $L_i(x)$.

→ Derivación del Polinomio de Lagrange

Ahora que sabemos cómo se define el polinomio de Lagrange, vamos a derivarlo. El objetivo es encontrar la **derivada** P'(x) del polinomio P(x). Para hacerlo, aplicamos la regla de la derivación a la expresión anterior.

$P(x) = \Sigma_{i=0}^{n} f(si)* L_i(x)$

Por lo tanto, el problema se reduce a encontrar la derivada de cada uno de los polinomios base $L_i(x)$. Esta derivada se denomina c(i), por lo que todo se reduce a la expresión

$$f^{(k}(x^*) \approx \sum_{i=0}^{n} c_i \cdot f(s_i)$$

donde c(i) es igual a la derivada de las bases de lagrange y f(si) son las imágenes de cada uno de los puntos del soporte.

→ CÓDIGO EN R

```
# Función para calcular el polinomio de Lagrange
lagrange <- function(x, X, Y) {
  n <- length(X)
  L <- 0
  for (i in 1:n) {
    li <- 1
    for (j in 1:n) {
      if (i != j) {
        li <- li * (x - X[j]) / (X[i] - X[j])
      }
    }
    L <- L + Y[i] * li
  }
  return(L)
}

# Función para calcular la derivada del polinomio de Lagrange
derivada_lagrange <- function(x, X, Y) {
  n <- length(X)
  dL <- 0
  for (i in 1:n) {
    dli <- 0
```

```
    for (j in 1:n) {
      if (i != j) {
        # Derivada del término Li (polinomio de Lagrange)
        term <- 1
        for (k in 1:n) {
          if (k != i && k != j) {
            term <- term * (x - X[k]) / (X[i] - X[k])
          }
        }
        dli <- dli + term / (X[i] - X[j])
      }
    }
    dL <- dL + Y[i] * dli
  }
  return(dL)
}
```

Función lagrange:

- Esta función calcula el polinomio de Lagrange en un punto x, usando un conjunto de puntos de interpolación X y sus valores correspondientes Y.
- La fórmula de Lagrange se construye sumando los productos de Yi y los polinomios li(x), que son fracciones diseñadas para pasar por cada punto (Xi,Yi).

Función derivada_lagrange:

- Calcula la derivada del polinomio de Lagrange en un punto x.
- Para ello, toma la derivada de cada uno de los términos li(x) (polinomios de Lagrange), utilizando la regla de la cadena y las reglas de derivación de fracciones.
- La derivada total se obtiene sumando las derivadas ponderadas de cada término li(x), multiplicadas por los valores correspondientes Yi.

2. DERIVACIÓN UTILIZANDO NEWTON (2 / 3 SOPORTES)

El **algoritmo de derivación de Newton** se basa en el uso de polinomios interpoladores, y puede ser utilizado para calcular la derivada de un polinomio interpolador en puntos determinados. En particular, la **fórmula de Newton** puede aplicarse con 2 o 3 puntos de soporte para aproximar la derivada de la función interpolada en un punto dado.

→ Fórmula de Newton con 2 Puntos de Soporte

Teniendo 2 puntos de soporte se obtiene un polinomio de grado 1 con el método de Newton. Es por ello que únicamente se podría obtener la primera derivada de dicho polinomio, que sería igual a esta expresión:

$$p(x) = f[s_0] + f[s_0, s_1](x - s_0)$$

$$f'(x^*) \approx p'(x^*) = f[s_0, s_1] = \frac{f[s_1] - f[s_0]}{s_1 - s_0}$$

Donde f[s0,s1] es la diferencia dividida del polinomio de grado 1.
En este caso podrían darse tres casuísticas:

- Diferencia finita hacia delante: la muestra está por detrás de mi incógnita.
- Diferencia finita hacia atrás: la muestra está por delante de mi incógnita.
- Diferencia finita centrada: la incógnita está equiespaciada entre dos muestras.

→ Fórmula de Newton con 3 Puntos de Soporte

En cambio, cuando tenemos 3 puntos en el soporte, el grado del polinomio obtenido es 2, por lo que se podría obtener hasta la segunda derivada de dicho polinomio.

La primera derivada sería la derivada primera del polinomio, pero al calcular la segunda derivada, esta se puede realizar de una manera mucho más cómoda y eficiente mediante la fórmula: $$f''(x^*) \approx p''(x^*) = f[s_0, s_1, s_2] = \frac{f(x^* + h) - 2f(x^*) + f(x^* - h)}{h^2}$$

→ CÓDIGO EN R

```r
# Función para calcular las diferencias divididas
diferencias_divididas <- function(X, Y) {
  n <- length(X)
  F <- matrix(0, nrow = n, ncol = n)
  F[, 1] <- Y
  for (j in 2:n) {
    for (i in 1:(n - j + 1)) {
      F[i, j] <- (F[i + 1, j - 1] - F[i, j - 1]) / (X[i + j - 1] - X[i])
    }
  }
  return(F)
}

# Función para calcular el polinomio de Newton en un punto x
newton_polynomial <- function(x, X, F) {
  n <- length(X)
  P <- F[1, 1]
  prod <- 1
  for (i in 2:n) {
    prod <- prod * (x - X[i - 1])
    P <- P + F[1, i] * prod
  }
  return(P)
}

# Función para calcular la derivada del polinomio de Newton en un punto x
derivada_newton <- function(x, X, Y) {
  F <- diferencias_divididas(X, Y)  # Calcular las diferencias divididas
  n <- length(X)

  # La derivada del polinomio de Newton se obtiene derivando término por término
  derivada <- 0
  prod <- 1
  for (i in 2:n) {
    derivada <- derivada + F[1, i] * prod
    prod <- prod * (x - X[i - 1])
  }

  return(derivada)
}
```

1. Función diferencias_divididas:
    - Esta función calcula las diferencias divididas, que son necesarias para construir el polinomio de Newton.
    - La matriz F es la tabla de diferencias divididas, donde F[i,j] representa la j-ésima diferencia dividida de los puntos X[i],X[i+1],…
2. Función newton_polynomial:
    - Esta función calcula el polinomio de Newton en un punto x, usando las diferencias divididas y los valores de X y Y.

- Utiliza la forma estándar del polinomio de Newton, sumando los términos correspondientes.

3. Función derivada_newton:
   - Esta función calcula la derivada del polinomio de Newton en un punto xxx.
   - La derivada se calcula derivando cada término del polinomio de Newton, lo cual es sencillo porque el término general tiene la forma f[x0,x1,…]∏i=1k(x−Xi), y su derivada es simplemente la suma de términos de la forma f[x0,x1,…]∏i=1k(x−Xi) con los productos sucesivos.

3. EXPANSIÓN DE TAYLOR

La expansión de Taylor es una aplicación de la serie de Taylor en la derivación de tipo interpolatorio.

Para obtener los c(i) se aplica la fórmula:

$$\sum_{i=0}^{n} c_i(\theta_i h)^k = k!$$

En esta fórmula la letra theta se deduce de la expresión

$$s_i = x^* + \theta_i h.$$

Para obtener los c(i) con los datos de theta, k(orden de la derivada), y h(longitud característica) se puede aplicar una expresión matricial

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ \theta_0 & \theta_1 & \theta_2 & \dots & \theta_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_0^{k-1} & \theta_1^{k-1} & \theta_2^{k-1} & \dots & \theta_n^{k-1} \\ \theta_0^{k} & \theta_1^{k} & \theta_2^{k} & \dots & \theta_n^{k} \\ \theta_0^{k+1} & \theta_1^{k+1} & \theta_2^{k+1} & \dots & \theta_n^{k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta_0^{n} & \theta_1^{n} & \theta_2^{n} & \theta_n^{n} & \theta_n^{n} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{k-1} \\ c_k \\ c_{k+1} \\ \vdots \\ c_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ k! \\ 0 \\ \vdots \\ 0 \end{bmatrix} \frac{1}{h^k}$$

Se despeja la columnas de las c(i).

Para calcular la derivada se utilizan los coeficientes obtenidos en la expresión

$$f^{(k}(x^*) \approx \sum_{i=0}^{n} c_i \cdot f(s_i)$$

→ CÓDIGO EN R

```r
# Función para calcular la derivada usando la expansión de Taylor
derivada_taylor <- function(f, a, h = 1e-5) {
  # f: la función de la que se quiere calcular la derivada
  # a: el punto alrededor del cual expandimos (punto de derivación)
  # h: un valor pequeño para la aproximación
  return((f(a + h) - f(a)) / h)
}

# Ejemplo de uso: derivada de f(x) = x^2 en x = 2
f <- function(x) { x^2 }
a <- 2  # El punto donde queremos la derivada

# Calcular la derivada en a utilizando la expansión de Taylor
derivada_resultado <- derivada_taylor(f, a)
cat("La derivada de f(x) = x^2 en x =", a, "es:", derivada_resultado, "\n")
```

Función derivada_taylor:

- Esta función calcula la derivada de la función f(x) en el punto aaa utilizando la fórmula de la expansión de Taylor de primer orden.
- La fórmula utilizada es: $f'(a) \approx (f(a+h) - f(a))/h$. Donde h es un valor pequeño que se utiliza para aproximar la derivada. Cuanto más pequeño sea h, más precisa será la aproximación, pero se debe elegir un valor suficientemente grande para evitar problemas de redondeo.

Parámetros:

- f: La función f(x) que deseas derivar.
- a: El punto alrededor del cual queremos calcular la derivada.
- h: Un valor pequeño que determina la precisión de la aproximación