Método de diferencias divididas de Newton para interpolación polinómica en R

Vamos a ver como realizar mediante las diferencias divididas de Newton un ejercicio de interpolación polinómica, programando en R.

Para empezar, creamos la función diferencias_divididas(x, y), que calcula la tabla de diferencias divididas. Inicialmente, se determina el número de puntos n y se crea una matriz llamada tabla para almacenar los resultados. La primera columna de esta matriz se llena con los valores de y. Luego, se utilizan dos bucles anidados para calcular las diferencias divididas, llenando las columnas restantes de la tabla. Al finalizar, se nombran las filas y columnas y se devuelve la tabla completa.

```
diferencias_divididas <- function(x, y) {
 n <- length(x) - 1
 # Inicializar la matriz para almacenar las diferencias divididas
 tabla <- matrix(0, nrow = n + 1, ncol = n + 1)
 # La primera columna son los valores de y
 tabla[, 1] <- y
 # Calcular las diferencias divididas
 for (j in 2:(n + 1)) {
  for (i in 1:(n + 2 - j)) {
   tabla[i, j] <- (tabla[i + 1, j - 1] - tabla[i, j - 1]) /
    (x[i + j - 1] - x[i])
  }
 }
 # Nombrar las filas y columnas
 rownames(tabla) <- as.character(x)
 colnames(tabla) <- paste0("Orden ", 0:n)
 return(tabla)
}
```

En segundo lugar, solicitamos al usuario que introduzca los valores de x e y, separados por comas. Estos valores se leen como cadenas y luego se convierten en vectores numéricos utilizando strsplit() y as.numeric(). A continuación, se verifica que ambos vectores tengan la misma longitud; si no es así, se genera un error con stop(), indicando que los vectores deben ser compatibles.

```
# Pedir valores de entrada

x_input <- readline(prompt = "Introduce los valores de x separados por comas: ")

y_input <- readline(prompt = "Introduce los valores de y separados por comas: ")

# Convertir las entradas a vectores numéricos

x <- as.numeric(unlist(strsplit(x_input, ",")))

y <- as.numeric(unlist(strsplit(y_input, ",")))

# Verificar que las entradas sean válidas

if (length(x) != length(y)) {

stop("Los vectores x e y deben tener la misma longitud.")

}
```

Después de obtener los vectores de entrada, el código llama a la función diferencias_divididas() con los valores introducidos por el usuario. El resultado, que es la tabla de diferencias divididas, se almacena en la variable tabla, y luego se imprime en la consola para que el usuario pueda verla.

```
# Llamar a la función y mostrar la tabla

tabla <- diferencias_divididas(x, y)

print(tabla)
```

Posteriormente, la función polinomio_newton_canonico(tabla_dif_div, x_orig) toma la tabla de diferencias divididas y los valores originales de x para construir el polinomio en su forma canónica. Extrae los coeficientes desde la primera fila de la tabla y comienza a construir el polinomio inicializando con el término constante. Utiliza un bucle para calcular cada término del polinomio mediante productos acumulativos y suma los resultados correspondientes. Finalmente, imprime la ecuación del polinomio en un formato legible y devuelve los coeficientes del polinomio.

```
# Función para convertir el polinomio de Newton a forma canónica

polinomio_newton_canonico <- function(tabla_dif_div, x_orig) { n <- length(tabla_dif_div[1, ]) - 1

# Extraer los coeficientes de Newton desde la primera fila de la tabla de diferencias divididas

coef_newton <- as.numeric(tabla_dif_div[1, ])

# Inicializar el polinomio con el término constante

pol <- c(coef_newton[1]) prod_pol <- 1

for (i in 2:(n + 1)) { prod_pol <- c(0, prod_pol) - c(prod_pol, 0) * x_orig[i - 1] pol <- c(pol, 0) + coef_newton[i] * prod_pol }

# Imprimir la ecuación del polinomio

cat("Ecuación del polinomio:\n")
```

```
cat("p(x) = ")
for (i in 1:length(pol)) {
 if (i > 1 && pol[i] > 0) cat(" + ")
if (pol[i] != 0) {
cat(sprintf("%.4f", pol[i]))
if (i > 1) cat(sprintf("x^%d", i - 1))
}
}
cat("\n")
return(pol)
}
#  Obtener el polinomio en forma canónica
polinomio_canonico <- polinomio_newton_canonico(tabla, x)
print(polinomio_canonico)
```

Además, también podemos evaluar el polinomio en un punto. La función evaluar_polinomio(coeficientes, x) se define para calcular el valor del polinomio en forma canónica para un valor específico de x. Esta función toma dos argumentos: coeficientes, que es un vector con los coeficientes del polinomio en orden ascendente de potencia, y x, que es el valor en el que se evaluará el polinomio. La función utiliza un bucle para sumar cada término del polinomio, multiplicando cada coeficiente por x elevado a la potencia correspondiente. El resultado final es la suma de todos estos términos.

```
# Función para evaluar el polinomio en forma canónica
evaluar_polinomio <- function(coeficientes, x) {
resultado <- 0
for (i in 1:length(coeficientes)) {
resultado <- resultado + coeficientes[i] * x^(i - 1)
 }
return(resultado)
}
```

Al igual que antes, tenemos que solicitar el valor de x en el que queremos evaluar el polinomio. Esto se hace mediante la función readline(), que muestra un prompt al usuario y espera su entrada. El valor introducido se almacena inicialmente como una cadena de texto, pero lo convertimos a número porque las operaciones matemáticas requieren valores numéricos, no cadenas de texto.

```
# Punto donde queremos evaluar el polinomio

x_eval <- readline(prompt = "Introduce valor de x para evaluar polinomio: ")

x_eval <- as.numeric(x_eval) # Convertir a número
```

Por último, si la conversión del valor introducido fue exitosa (es decir, se introdujo un número válido), el código evalúa el polinomio en el punto x proporcionado utilizando la función evaluar_polinomio(), que toma los coeficientes del polinomio canónico y el valor de x. El resultado de esta evaluación se almacena en la variable resultado.

Finalmente, se imprime un mensaje formateado mediante la función cat(), que muestra el valor de x introducido junto con el resultado calculado del polinomio en ese valor.

```
# Verificar si la conversión fue exitosa

if (is.na(x_eval)) {

cat("Error: Por favor, introduce un número válido.\n")

 } else {

resultado <- evaluar_polinomio(polinomio_canonico, x_eval)

cat("\nResultado del polinomio en x =", x_eval, "es:", resultado, "\n")

}
```