EXPLICACIÓN SIMULACRO DE R

ENUNCIADO

Un sistema de liberación controlada de un fármaco sigue una dinámica en la que la concentración $f(x)$ del fármaco en el plasma está influenciada por una función periódica modulada:

$$f(x) = \sum_{k=1}^{5} \frac{(-1)^k}{k!} x^k \sin(kx)$$

donde x representa el tiempo en horas y los primeros cinco términos de la serie definen el comportamiento del sistema. Queremos comprobar que nuestra hipótesis acerca del proceso es cierta evaluando la velocidad a la que varía la concentración. En el laboratorio en donde estamos monitorizando este proceso sólo tenemos acceso a la concentración de la muestra en ciertos instantes.

Vamos a simular el comportamiento de nuestro sistema de medición y de la muestra. Para ello:

(A) Define un vector que contenga los instantes de medición, sabiendo que: dispongo de 3 muestras equiespaciadas entre los instantes $x = [3, 3.5]$ y otras 3 (también equiespaciadas) en $x = [3.5, 4.5]$. En total dispongo de 5 muestras, ya que el punto de enganche para ambos intervalos, 3.5, es compartido. (1 PTO)

(B) Define la función $f(x)$, de tal forma que cuando ejecutes `f(x)` en cada uno de los puntos del soporte definidos en a) simule las medidas devueltas por el sistema de medición empleado en el laboratorio. (1 PTO)

(C) Evalúa $f(x)$ en el soporte calculado en el apartado a. Estos datos nos servirán como simulación de nuestro sistema de medición. (1 PTO)

(D) Define la función `f_first_analytical`. Ésta toma un valor de x y calcula $f'(x)$. Ten en cuenta que:

$$f'(x) = \sum_{k=1}^{5} \frac{(-1)^k}{(k-1)!} \left[x^{k-1} \sin(kx) + x^k \cos(kx)\right]$$

(1 PTO)

(E) Determina el valor de $f'(3.8)$ evaluando `f_first_analytical` y también empleando **el método de los coeficientes indeterminados**, teniendo en cuenta que el soporte disponible es el generado en el apartado a. Imprime ambos resultados por pantalla. (1.5 PTO)

(F) Genera un vector `x_vals` formado por una secuencia de 100 elementos equiespaciados entre 3.1 y 4.5. (1 PTO)

(G) Genera dos vectores del mismo tamaño que `x_vals`, tales que:

- El primero sea el resultado de evaluar la función de la expresión analítica del apartado b en `x_vals`. (1 PTO)
- El segundo sea la estimación de la derivada $f'(x)$ mediante el método de los coeficientes indeterminados. (1.5 PTO)

(H) Pinta en un mismo eje los dos vectores obtenidos en el apartado g. Añade una leyenda que explicite quién de las dos gráficas corresponde al resultado analítico y cuál a la aproximación numérica. (1 PTO)

SOLUCIÓN

Siempre ponemos estos comandos al principio de los códigos para limpiar la consola y las variables y funciones que teníamos almacenadas.

```
cat(" \014 ")

rm( list =ls ())
```

Las siguientes líneas de código son necesarias para el ejercicio. Es el código del método de derivación mediante diferencias indeterminadas y posee funciones que vamos a llamar a lo largo de los apartados del simulacro.

```
diff _ indeterminados <- function (s_points , x_0, orden _ diff ) {

        s_ points <- unique (s_ points )


        abs _ diff <- abs (s_ points - x_0)

        filtered _ diff <- abs _ diff [abs _ diff != 0]


        if ( length ( filtered _ diff ) == 0) {

                stop (" Todos los puntos son iguales a x_0, no se puede calcular la
                derivada.")

        }


        h <- min ( filtered_diff )

        n_points <- length (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 ,")."))

        }


        thetas <- (s_ points - x_0) / h

        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)
                }
        }

        diff _ equal <- numeric ( length = n_ points )
        diff _ equal [ orden _ diff + 1] <- factorial ( orden _ diff )

        coff _ indet _ diff <- solve ( vandermonde _matrix , diff _ equal * (1 / h ^ orden _diff))

        return ( coff _ indet _ diff )
}
```

 # APARTADO A

Como disponemos de 6 muestras equiespaciadas, definimos una variable n que contenga el número total de muestras.

```
n <- 6
```

Define que el orden de la derivada es 1 y se va a utilizar en el apartado e).

```
orden _diff <- 1
```

Genera una secuencia llamada aux con un total de n/2 muestras (en este caso son 3 porque 6/2 = 3) equiespaciadas en el intervalo [3, 3.5].

```
aux = seq (3, 3.5 , length .out = n/2)
```

Genera una secuencia llamada aux2 con un total de n/2 muestras (en este caso son 3 porque 6/2 = 3) equiespaciadas en el intervalo [3.5, 4.5].

```
aux2 = seq (3.5 ,4.5 , length .out=n/2)
```

Crea un vector llamado s_points cuyas componentes son las muestras de las secuencias generadas anteriormente (aux y aux2), eliminando la muestra 1 de aux2.

Es decir, como el valor 3.5 está repetido, coge todos los valores de aux2 excepto el primero. Ello se expresa así: aux2 [2: length (aux2)].

```
s_ points <- c(aux , aux2 [2: length ( aux2 )])
```

 # --- APARTADO B

Simplemente define la función f(x) del enunciado que la va a almacenar en la variable f.

1. Inicializa la variable de salida denominada result a 0.

2. Genera un bucle for para llevar a cabo el sumatorio y dentro escribe escribe la fórmula.

3. Devuelve result mediante el comando return.

```
f <- function (x) {
        result <- 0
        for (k in 1:5) {
                result <- result + (( -1)^k / factorial (k)) * x^k * sin (k * x)
        }
        return(result)
}
```

 # --- APARTADO C

Para evaluar los puntos de soporte del vector generado en el apartado a) en la función definida en el apartado b) utiliza un bucle for.

1. Define la variable que va a almacenar los valores y se va a llamar f_values. Esta, como depende de la longitud del vector s_points, se define así: f_ values <- numeric (length (s_points)).

2. Creamos un bucle for desde que i toma el valor 1 hasta la longitud de s_points (5). Dentro del bucle se escribe f_values [i] <- f(s_points [i]) para que en f_values se alamcenen elemento a elemento los valores de los puntos de soporte evaluados en la función.

```
# Evaluar f(x) en los puntos de soporte
f_ values <- numeric ( length (s_points ))
for (i in seq_ along (s_ points )) {
        f_values [i] <- f(s_points [i])
}
```

 # --- APARTADO D

Igual que el apartado b), solo que en este caso en vez de definir f(x) define su derivada primera, llamando a la función f_first _analytical.

```
# Derivada analítica primera de f(x)

f_ first _ analytical <- function (x) {

        result <- 0

        for (k in 1:5) {

                result <- result + (( -1)^k / factorial (k -1)) * (x^(k - 1) * sin (k * x) + x^k *
                cos(k*x))

        }

        return ( result )

}
```

 # --- APARTADO E

--- E.1.

Para hallar el valor de la derivada de 3.8 en la función definida en el apartado d) hay que crear una variable (x_0) a la cual se le asigna dicho valor (3.8). Después para mostrar el resultado utiliza print de la función y entre paréntesis la variable a evaluar, en este caso, x_0.

```
x_0 <- 3.8

print (f_ first _ analytical (x_0))
```

--- E.2.

Para poder emplear el método de los coeficientes indeterminados hay que utilizar las funciones del código de derivación mediante diferencias indeterminadas que hay antes de empezar a resolver el ejercicio.

La función que hay que usar es la de diff_indeterminados (s_points , x_0, orden_diff) que se encarga de devolver los coeficientes numéricos. Siendo s_points el vector con los puntos de soporte, x_0 el valor a aproximar y orden_diff el orden de la derivada. En nuestro caso se nombra a la función como coff_indet_diff.

Después, siguiendo con la fórmula del método de derivación por diferencias indeterminadas hay que hacer un sumatorio de la multiplicación entre dichos coeficientes y los valores de la función f(x) evaluada en el soporte (f_values calculados en el apartado c)). Ello es lo que printea.

```
#numéricamente

coff_indet_diff <- diff_indeterminados (s_points , x_0, orden _ diff )

print (sum( coff_indet_diff * f_values ))
```

 # --- APARTADO F

Aquí genera un vector denominado x_vals formado por una secuencia de 100 muestras equiespaciadas en el intervalo [3.1, 4.5].

```
x_vals <- seq (3.1 , 4.5 , length .out = 100)
```

 # ---- APARTADO G

--- G.1.

Hay que evaluar la función f_first_analytical (definida en el apartado d)) en x_vals. Para ello, sigue el mismo procedimiento que el apartado c) ya explicado. En este caso, los valores los va a almacenar en un vector llamado first_derivative_analytical.

```
# Valores analíticos y numéricos de la derivada

first_derivative_analytical <- numeric ( length (x_vals ))

for (i in seq_ along (x_ vals )) {

        first_derivative_analytical [i] <- f_ first _ analytical (x_ vals [i])

}
```

--- G.2.

Mismo procedimiento que en el apartado e), pero esta vez al no ser solo un valor a aproximar hay que utilizar un bucle for. Entonces, cambia que x_0 ahora va a ser uno a uno los elementos del vector x_vals según la iteración i en del bucle for. Es decir, cuando i=1 se cogerá el valor en la posición 1 de x_vals y así sucesivamente.

```
estimated_first_derivative <- numeric ( length (x_vals ))

for (i in seq_along (x_vals )) {

        x_0 <- x_ vals [i]

        coff_indet_diff <- diff_indeterminados (s_points , x_0, orden _ diff )

        estimated_first_derivative [i] <- sum ( coff_indet_diff * f_values )

}
```

 # --- APARTADO H

Por último, dibuja la gráfica comparando los resultados de ambas derivadas. Coloca en el eje X x_vals porque es la secuencia sobre la que se apoya first_derivative_analytical, que lo coloca en el eje Y (los vectores, secuencias… que se colocan en los ejes deben ser de la misma longitud para que funcione). Luego, especifica los parámetros de la gráfica. Además, añade una línea con x_vals sobre el eje X de nuevo y estimated_first_derivative en el eje Y, consiguiendo comparar ambos valores de las derivadas. Finalmente, añade una leyenda.

(más información de cómo dibujar en este recurso:

https://cooperativo.liti.digym.upm.es/2024/12/23/dibujar-en-r/)

Pinta los resultados

plot (x_vals , first_derivative_analytical , type = "l", col = " blue ", lwd = 2, xlab = "x", ylab = " Derivada primera ", main = " Derivada primera : Analítica vs Numérica ")

lines (x_vals , estimated_first_derivative , col = "red", lwd = 2)

legend (" topright ", legend = c(" Analítica ", "Numérica "), col = c(" blue ", "red"), lty =1, lwd= 2)

Se vería así: