INFORME TAREA 1

Tomás Rojas C.
RUT:19.688.339-8
Github: @tomasrojasc

1. Pregunta 1

Para la primera parte de la tares, teníamos que comparar dos maneras de calcular derivadas numéricamente.

para esto, la función a evaluar fue $f(x) = \sin(x/2)$ y el punto a evaluar (dado mi RUT) fue $x = 1.339$. El primer método a evaluar fue el siguiente:

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{1}$$

Mientras que el método propuesto es el del enunciado.

Para lograr la derivada tenemos dos funciones que nos permiten usar ambos métodos a la vez de especificar el grado de precisión deseado.

A continuación podemos ver un gráfico donde la línea verde punteada es el valor que nos da la derivada real de la función importada desde el paquete math de python.

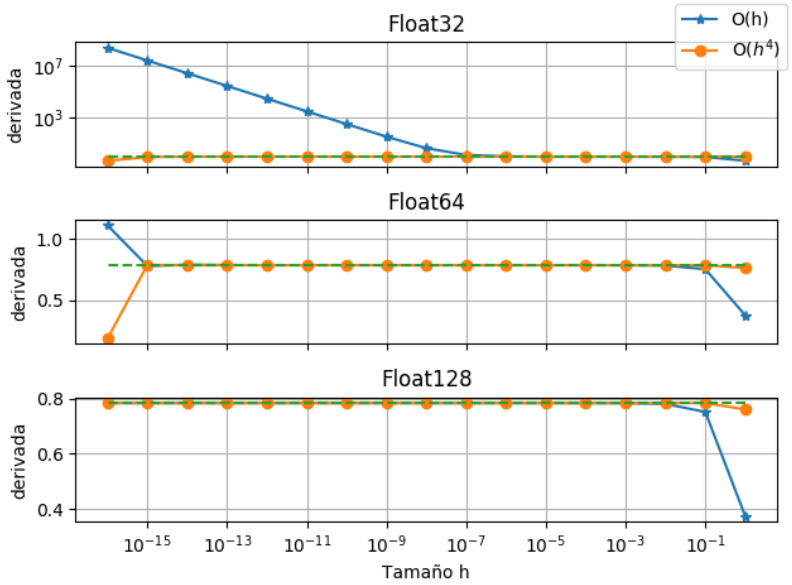

Figura 1: Comparación de los dos métodos con distintos grados de precisión.

Es importante notar que el primer gráfico, a diferencia de los otros dos, tiene escala logarítmica y no lineal. Notamos que el método depende en gran medida del nivel precisión que usemos, por ejemplo, para float32, la precisión es tan baja, que al truncar para valores de h pequeños, obtenemos divisiones por cero, lo que hace que nuestra aproximación esté muy lejos del valor real, pero vemos que para h del órden de 10^{-5} se comporta mejor, pero el rango donde esta aproximación es fiable

es muy pequeño. Por otro lado, vemos que con la aproximación propuesta en el enunciado esta convergencia es mucho más rápida al valor de la línea punteada. También vemos que el intervalo de h para el cual la aproximación de (1) es fiable, es siempre menor que el de la aproximación del enunciado, esto es un punto a considerar ya que nos dice cuánto cambio de resolución acepta nuestro algoritmo, lo cual es muy útil ya que si tenemos una función errática y una que se comporta mejor en el sentido que no tiene cambios de concavidad brusco y tiene derivadas pequeñas, podemos usar la misma función con distintos h y esperar un buen resultado sin tener que hacer otra implementación.

En síntesis, si usamos un número de mayor presición, logramos que el rango donde nuestra aproximación es útil, en términos de h sea mucho más amplio.

2. Pregunta 2:

Lo primero es regularizar la integral. Para ello usamos el siguiente cambio de variable $\sin\theta = \frac{\sin(\phi/2)}{\sin(\phi_/2)}$ y $k = \sin(\phi_0/2)$, con lo que el problema se transforma en:

$$\frac{T}{T_0} = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1-k^2\sin^2\theta}} \tag{2}$$

Al integrar para un muestreo de los ϕ_0 indicados en el enunciado, tenemos el siguiente gráfico:

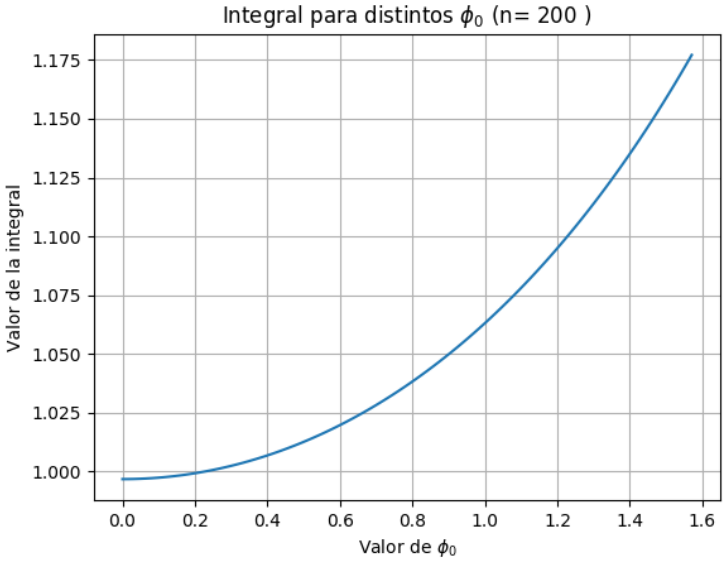

Figura 2: Resultado de la integral para distintos ϕ_0

Para lograr esto hubo que restar un poco el dominio original y restarle a $\pi/2$ un poco con la intención de evitar divisiones por cero, de todas maneras, lo que se restó es despreciable con respecto al dominio de integración, por lo que es posible argumentar que sigue siendo una buena aproximación para nuestro problema.

En cuanto a la velocidad de nuestra implementación, al usar %timeit obtivimos que para la función "Evalua" una vuelta tarda en promedio $11.1ms \pm 286\mu s$ en cambio, (teniendo en cuenta

que la aproximación implementada cumplió el criterio de convergencia con $n = 200$, por lo que usaremos el mismo n en las implementaciones de scipy) tenemos que *scipy.integrate.trapz*

3. Discusión y Conclusiones

Podemos concluir que por un lado la presición es muy importante al resolver problemas numéricos, pero hay que tener en cuenta que hay consecuencias en la precisión y esto es la demanda computacional, ya que siempre es una aproximación y por lo mismo nos podemos dar libertades como las hechas en la pregunta 2, donde cortamos el dominio de integración para evitar problemas al evaluar la función.