Diferenciación e Integración

Isabela Acevedo García

Pontificia Universidad Javeriana Cali

Cali, Colombia

isa43461@javerianacali.edu.co

Abstract— En el presente documento se explicará los procesos que se llevaron a cabo para resolver el problema de diferenciación e integración numérica con el método de diferencias finitas adelante, atrás y centrada para la diferenciación y los métodos de punto medio, trapezoide y simpson.

I. INTRODUCCIÓN

Se desarrollaron dos archivos en Python de tal forma que cada uno cumpliera con los requisitos solicitados, el primer archivo resuelve el problema de diferenciación e integración, es decir que en este archivo se encuentra el método de diferencias finitas adelante, atrás y centrada para la diferenciación y los métodos de punto medio, trapezoide y simpson, también las funciones que necesitan para su desarrollo. El segundo archivo se hizo para realizar las simulaciones (main).

II. DESARROLLO DE CONTENIDO.

A. Materiales y métodos.

➔ **Materiales:**

- ***Python:*** Este fue el lenguaje utilizado para desarrollar los métodos de ecuaciones normales y ortogonalización. *"Python es un lenguaje de programación de código abierto, orientado a objetos, muy simple y fácil de entender. Tiene una sintaxis sencilla que cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único."* [1].
- ***Función round***: Esta es una función de Python utilizada para redondear los dígitos. *"La función* ***round*** *redondea un número a una precisión dada por el número de cifras decimales indicadas como argumento. Si no se indica el número de cifras decimales, se toma 0 como valor por defecto."* [2].
- ***Librería Statistics:*** Esta librería ayudó a sacar la media del error. ***"****El módulo statistics implementa muchas fórmulas estadísticas comunes para cálculos eficientes usando varios tipos numéricos de Python (int, float, Decimal, y Fracción)." [3].*
- ***Librería Numpy:*** Esta librería ayudó a hacer operaciones de álgebra lineal de forma eficiente. *"NumPy es un paquete de Python que significa "Numerical Python", es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales. Estas estructuras de datos garantizan cálculos eficientes con matrices." [4].*
- ***Librería Matplotlib:*** Esta librería ayudó a graficar el resultado de los dos métodos. *"Matplotlib es una librería de trazado utilizada para gráficos 2D en lenguaje de programación Python, es muy flexible y tiene muchos valores predeterminados incorporados que te ayudarán muchísimo en tu trabajo."* [5].
- ***Libreria time:*** Esta librería ayudó para sacar el tiempo de los algoritmos. *"El módulo time brinda acceso a varios tipos diferentes de relojes, cada uno útil para diferentes propósitos. "*[6]

:

➔ **Metodología**

La metodología estuvo dividida en dos partes. La primera fue la diferenciación numérica que constó de 3 métodos los cuales son diferencias finitas adelante, atrás y centrada. En cada uno de los métodos se

realizó la formula en lenguaje el cual Python entendiera cada método de diferenciación.

★ Diferencia finita hacia adelante:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

★ Diferencia finita hacia atrás:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

★ Diferencia finita centrada:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Se probó con 3 distintas h para ver cómo cambiaban los resultados y las gráficas, y también con 3 distintas funciones para ver en cuáles de los 3 métodos se podía ver mayor exactitud el valor real.

La segunda fue en la integración numérica ya que esta se compone de 3 reglas, las cuales son rectángulo, trapezoide y simpsons, con cada una de ellas se hizo la sumatoria respectiva en lenguaje el cual Python entendiera para cada regla.

★ Regla de rectángulo:

$$I(f) = M_c(f) = \sum_{i=1}^{n} (x_i - x_{i-1})\, f\left(\frac{x_{i-1} + x_i}{2}\right).$$

★ Regla del trapezoide:

$$I(f) = T_c(f) = \frac{1}{2} \sum_{i=1}^{n} (x_i - x_{i-1}) \left[f(x_{i-1}) + f(x_i)\right].$$

★ Regla de Simpsons:

$$I(f) = S_c(f) = \frac{1}{6} \sum_{i=1}^{n} (x_i - x_{i-1}) \left[f(x_{i-1}) + 4f\left(\frac{x_{i-1} + x_i}{2}\right) + f(x_i)\right].$$

Se probó con 3 distintas funciones para ver cómo cambiaban los resultados y en cuáles de las 3 reglas se podía ver mayor exactitud el valor real.

Resultados con H = 1

➢ ***Función número 1***

$$f(x) = 2x^4 + x^3 - x^2 + 4$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.34796***	***0.25231***	***0.04782***
Desviación Estándar	***0.19640***	***0.10047***	***0.04889***
Tiempo (sg)	***1.5578508377075194e-05***	***1.5620708465576172e-05***	***1.563096046447754e-05***

Fig. 1 Es el ejemplo #1 graficado en Python con la librería matplotlib.

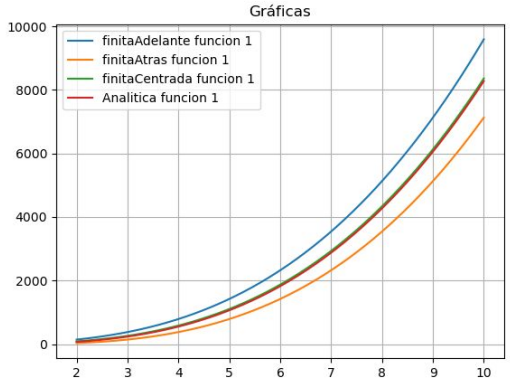

Fig. 1 función#1 con h =1.

- ***Análisis y discusión***

Se puede observar que las función finita centrada es la que se acerca más a la función real, la cual vendría siendo la función análitica, sin embargo la función finita adelante y atrás tienen la misma forma pero no es igual a la análitica, por lo cual los margenes de error de éstas 2 serán mayores ya que se alejan del resultado.

➢ ***Función número 2***

$$f(x) = \frac{3}{x^2}$$

	Adelante	*Atrás*	*Centrada*
Error Medio	***0.22852***	***0.46815***	***0.11981***
Desviación Estándar	***0.082118***	***0.36809***	***0.14559***
Tiempo (sg)	***8.976221084594727e-06***	***1.300549507141113 3e-05***	***2.193808555603027 4e-05***

Fig. 2 Es la función #2 graficado en Python con la librería matplotlib.

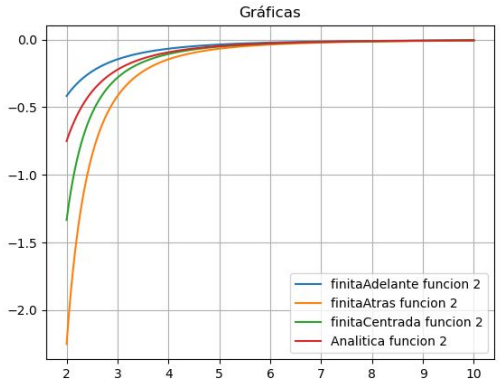

Fig. 2 función #2 con h = 1.

- ***Análisis y discusión***

En la gráfica podemos observar que ninguna de las 3 funciones se acerca a la analítica desde 2 hasta 5 (eje x), sin embargo todas están muy juntas a partir del 5 hasta el número 10.

➢ ***Ejemplo número 3***

$$f(x) = \sqrt{x^2 - 2x + 3}$$

	Adelante	*Atrás*	*Centrada*
Error Medio	***0.025309***	***0.047097***	***0.010894***
Desviación Estándar	***0.04415***	***0.08617***	***0.02103***
Tiempo (sg)	***3.123807907104492e-05***	***0.0***	***1.562690734863281 2e-05***

Fig. 3 Es el ejemplo #3 Graficado en Python con la librería matplotlib.

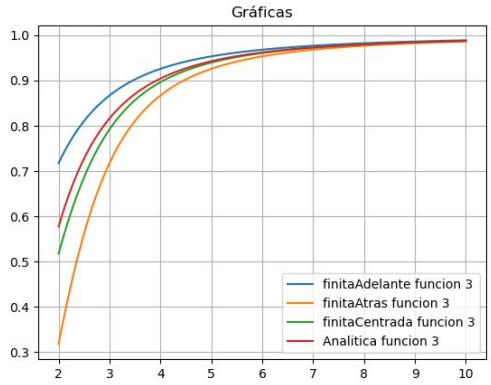

Fig. 3 función #3 con h=1.

- ***Análisis y discusión***

En comparación con la gráfica anterior podemos observar que ambas son similares, sin embargo, estas se empiezan a juntar a partir del punto 6 o 7 en el eje x, y también que tienen una curvatura menos pronunciada, podemos ver con facilidad que la finita centrada es la que trata de ser más similar a la analítica.

B. ***Resultados con H=0.1***

➢ ***Función número 1***

$$f(x) = 2x^4 + x^3 - x^2 + 4$$

	Adelante	*Atrás*	*Centrada*
Error Medio	***0.03015***	***0.02919***	***0.000478***

Desviación Estándar	***0.01480***	***0.01384***	***0.000488***
Tiempo (sg)	***0.0***	***1.562023162841797e-05***	***3.1242847442626954e-05***

Fig. 1 Es el ejemplo #1 graficado en Python con la librería matplotlib.

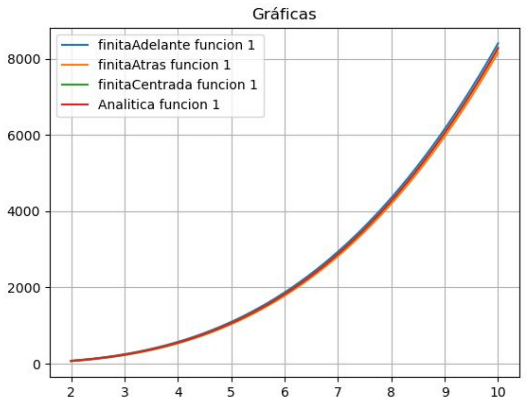

Fig. 1 función #1 con h =0.1

.

- ***Análisis y discusión***

En esta gráfica, al evaluarse con el h en 0.1 podemos observar que las funciones (adelante, atrás, centrada, analitica) son más difíciles de diferenciarse, en comparación a la misma función con h en 1, ya que ahí se podía ver la diferencia de manera más clara.

➢ ***Ejemplo número 2***

$$f(x) = \frac{3}{x^2}$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.0292263***	***0.03123***	***0.001003***
Desviación	***0.013709***	***0.015747***	***0.001039***
Estándar			
Tiempo (sg)	***3.124260902404785e-05***	***1.5621185302734374e-05***	***0.0***

Fig. 2 Es el ejemplo #2 graficado en Python con la librería matplotlib.

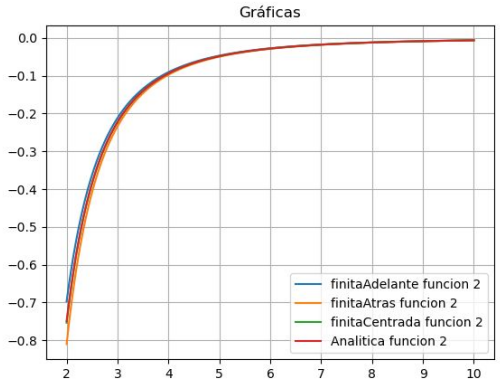

Fig. 2 función#2 con h=0.1.

- ***Análisis y discusión***

En esta gráfica no es muy evidente la diferencia entre cada función, lo más notorio que se puede evidenciar es que la función hacia atrás y adelante es mucho más fácil de ver que las demás, esto no pasaba tanto al tener un h mayor como h = 1.

➢ ***Ejemplo número 3.***

$$f(x) = \sqrt{x^2 - 2x + 3}$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.003266***	***0.00588***	***0.0001***
Desviación Estándar	***0.00588***	***0.006307***	***0.00021***
Tiempo (sg)	***1.562428474426269 6e-05***	***3.127574920654297e-05***	***1.562643051147461e-05***

Fig. 3 Es el ejemplo #3 Graficado en Python con la librería matplotlib.

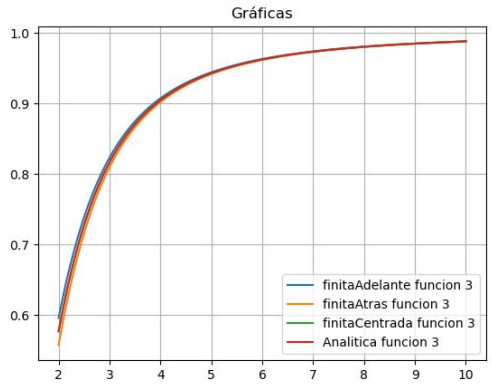

Fig. 3 función #3 con h=0.1.

- ***Análisis y discusión***

En la gráfica se puede observar que solo se ve de manera notable la finita adelante y atrás, la análitica y la centrada, más que todo la finita centrada no se logra ver por ningún lado, esto quiere decir que esta es la que más trata de precisar a la análitica, sin embargo con un h tan pequeño, no se puede ver de manera clara.

C. Resultados con H = 1.5

➢ ***Función número 1***

$$f(x) = 2x^4 + x^3 - x^2 + 4$$

	Adelante	*Atrás*	*Centrada*
Error Medio	***0.5642442***	***0.34903***	***0.107606***
Desviación Estándar	***0.340502***	***0.12454***	***0.109952***
Tiempo (sg)	***0.0***	***1.5622377 395629884 e-05***	***3.1279563 90380859 e-05***

Fig. 1 Es el ejemplo #1 graficado en Python con la librería matplotlib.

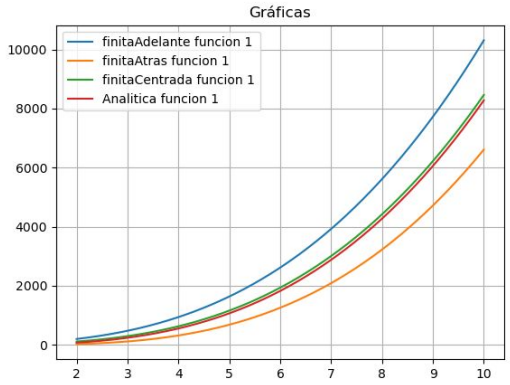

Fig. 1 función#1 con h=1.5

- ***Análisis y discusión***

En esta gráfica se puede evidenciar más fácil que la finita adelante y la atrás están alejadas de la analitica, pero la finita centrada es la más cercana a la original, quiere decir que la centrada es la que tiene menor margen de error.

➢ ***Ejemplo número 2***

$$f(x) = \frac{3}{x^2}$$

	Adelante	*Atrás*	*Centrada*
Error Medio	***0.30668***	***1.06514***	***0.37923***
Desviación Estándar	***0.09770***	***1.333292***	***0.62645***
Tiempo (sg)	***0.0***	***3.1241893 76831055e -05***	***1.5620708 465576172 e-05***

Fig. 2 Es el ejemplo #2 graficado en Python con la librería matplotlib.

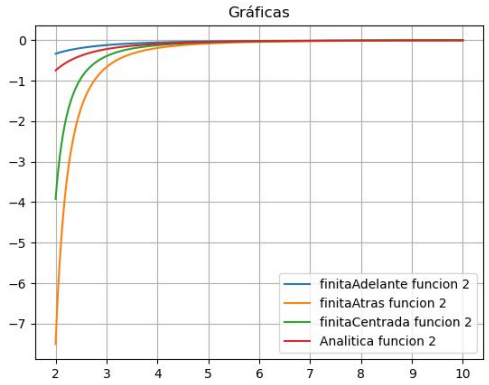

Fig. 2 función#2 con h= 1.5

- ***Análisis y discusión***

Podemos observar que desde 2 a 5 en el eje x las funciones están todas separadas de cada una, sin embargo en este caso, se puede ver que la función adelante y analitica son las más juntas desde el principio.

➢ ***Ejemplo número 3.***

$$f(x) = \sqrt{x^2 - 2x + 3}$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.03362***	***0.08290***	***0.024638***
Desviación Estándar	***0.05770***	***0.14835***	***0.04544***
Tiempo (sg)	***1.4959096908569336e-05***	***1.4959096908569336e-05***	***2.1942377090454102e-05***

Fig. 3 Es el ejemplo #3 Graficado en Python con la librería matplotlib.

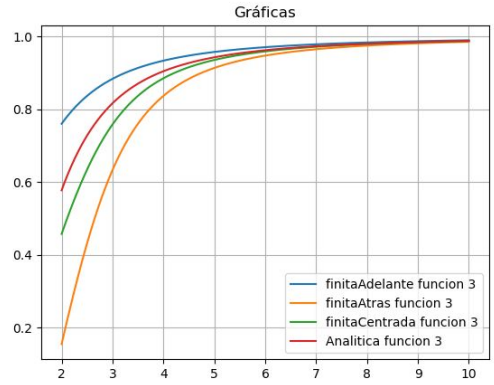

Fig. 3 función #3 con h=1.5

- ***Análisis y discusión***

Podemos observar que desde 2 a 7 en el eje x las funciones están todas separadas de cada una, sin embargo en este caso, se puede ver que la función centrada y analitica son las más juntas desde el principio.

D. ***Resultados con H=0.5***

➢ ***Función número 1***

$$f(x) = 2x^4 + x^3 - x^2 + 4$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.160737***	***0.1368248***	***0.0119562***
Desviación Estándar	***0.084228***	***0.060252***	***0.0122169***
Tiempo (sg)	***1.1036872863769532e-05***	***3.445577621459961e-05***	***0.0***

Fig. 1 Es el ejemplo #1 graficado en Python con la librería matplotlib.

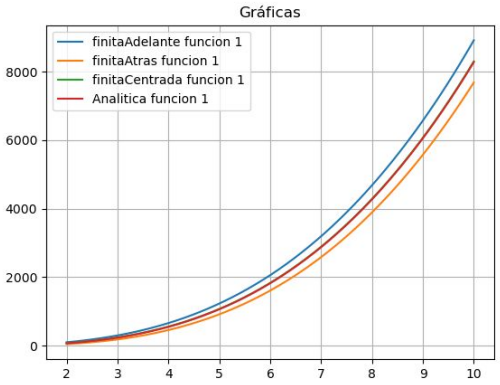

Fig. 1 función #1 con h =0.5

- ***Análisis y discusión***

En esta gráfica, al evaluarse con el h en 0.5 podemos observar que las funciones (centrada, analitica) son más difíciles de diferenciarse, en comparación a la misma función con h en 1, ya que ahí se podía ver la diferencia de manera más clara. La función finita centrada no se logra ver ya que tapa la función análitica, por lo que en este caso el error de la centrada es poco.

➢ ***Ejemplo número 2***

$$f(x) = \frac{3}{x^2}$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.12982***	***0.181954***	***0.0260659***
Desviación Estándar	***0.053563***	***0.108397***	***0.0279578***
Tiempo (sg)	***1.0003089 904785157 e-05***	***9.1776847 83935547e -06***	***2.6431560 51635742e -05***

Fig. 2 Es el ejemplo #2 graficado en Python con la librería matplotlib.

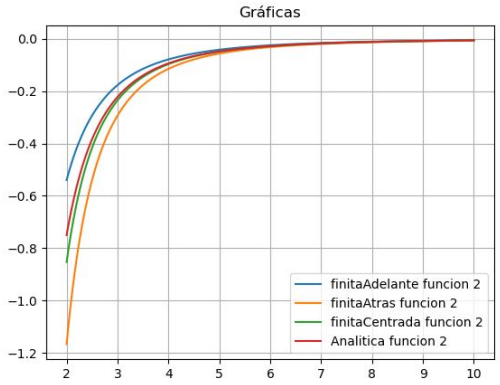

Fig. 2 función#2 con h=0.5

- ***Análisis y discusión***

En esta gráfica, lo más notorio que se puede evidenciar es que la función hacia atrás y adelante están más alejadas de las demás, y se puede ver de manera leve la finita centrada, sin embargo sigue estando muy junta a la análitica.

➢ ***Ejemplo número 3.***

$$f(x) = \sqrt{x^2 - 2x + 3}$$

	Adelante	***Atrás***	***Centrada***
Error Medio	***0.014491***	***0.019867***	***0.002688***
Desviación Estándar	***0.02573***	***0.036359***	***0.005318***
Tiempo (sg)	***1.1289834 97619629e -05***	***1.8030166 62597656e -05***	***8.0010890 96069335e -06***

Fig. 3 Es el ejemplo #3 Graficado en Python con la librería matplotlib.

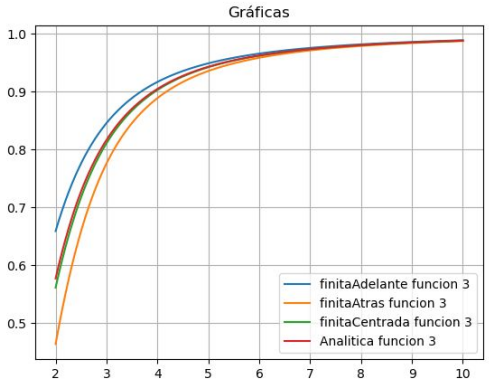

Fig. 3 función #3 con h=0.5.

- ***Análisis y discusión***

En la gráfica se puede observar que solo se ve de manera notable la finita adelante y atrás, la analítica y la centrada, más que todo la finita centrada no se logra ver mucho, esto quiere decir que esta es la que más trata de precisar a la analítica.

Resultados para integración

	$\int x\,sin(x)dx$	$\int e^{x}cos(x)dx$	$\int x\sqrt{1+x}\,dx$
Rectángulo	0.30059256746846086	1.3787817190027822	0.6434704472388555
Trapezoide	0.3023205824939365	1.3765098282342558	0.6444300126935935
Simpsons	0.3011685724769528	1.3780244220794	0.6437903023904349
Integral normal	0.30116	1.37802	0.64379

III. CONCLUSIONES

Finalmente, podemos concluir que la función para diferenciación que más se acerca a la función analitica es la función finita centrada ya que en la mayoría de las gráficas era la más similar o casi identica a la analitica, se puede observar también como cambian las funciones respecto al h ya que entre más pequeño sea, menos se puede diferenciar cómo actúan las gráficas para poder compararlas, en cambio en los h como 1 o 1.5 se podía diferenciar con facilidad cuál era la gráfica con menos margen de error.

También podemos observar que con las reglas de simpson, trapezoide y rectángulo, se nota una diferencia entre cada una de ellas, pero siempre habrá una que se aproxima más al valor real.

IV. REFERENCIAS

[1] A. Soloaga (2018, Octubre 19) “Principales Usos de Python”.[Online]. Disponible en: https://www.akademus.es/blog/programacion/principales-usos-python/

[2] D. Barrueco. (2019, Enero 19). “ROUND”. [Online]. Disponible en: https://www.interactivechaos.com/python/function/round

[3]. E. Rico (2018). “Statistics-Cálculos estadísticos” [Online]. Disponible en: https://rico-schmidt.name/pymotw-3/statistics/index.html

[4]. L. González. (2018, Septiembre 21) “Introducción a la librería NumPy de Python – Parte 1 ``.[Online]. Disponible en: https://ligdigonzalez.com/introduccion-a-numpy-python-1/

[5]. L. González. (2018, Octubre 19). “Introducción a la Librería Matplotlib de Python – Parte 1 ``.[Online]. Disponible en:

https://ligdigonzalez.com/libreria-pandas-de-matplotlib-tutorial/

[6]. E. Rico (2018). “time- Hora del reloj” [Online]. Disponible en: https://rico-schmidt.name/pymotw-3/time/