EVALUACION 1 COMPUTACIÓN NUMÉRICA

Estudiantes: Mario González Galdames

Luis Inostroza Flores

Docente: Juan Cabezas Contardo

Fecha: 8/10/2020

Resumen

El problema planteado por el docente consistía en varias instrucciones a realizar en el lenguaje de programación Python. En primer lugar, debíamos generar dos listas con números aleatorios del tipo float64, los cuales utilizaríamos para transformarlos a números del tipo binario. Luego, con cada una de las tablas modificadas obtendríamos los valores de “suma binaria” que correspondería a la suma de ambas listas con valores binarios, y “suma real” que vendría siendo la suma de las dos primeras listas con valores reales. A continuación, pasaríamos los valores binarios a decimales para ir al siguiente paso. Finalmente, con esos valores obtenidos, debíamos generar resultados que mostraran los diferentes errores obtenidos o métricas, con sus respectivos gráficos para obtener una mejor observación visual de los datos.

Lo anterior corresponde a una explicación a grandes rasgos de que es lo que debíamos realizar; si nos vamos a un análisis más profundo paso a paso de los tópicos a completar, tendríamos lo siguiente: Lo principal sería crear dos variables tipo lista sobre las cuales trabajar (utilizando la biblioteca numpy para ayudarnos con esa tarea) de nombres “operando\_1” y “operando\_2”, respectivamente, de 80 o más valores, pasarle datos flotantes aleatorios y con ellos saltar al siguiente paso. A partir de los valores anteriores, generaremos su transformación a valores del tipo binario mediante una función que nos devolviera dos nuevas listas, a las cuales llamaremos “operando\_1\_binario” y “operando\_2\_binario”, respectivamente. El orden de los siguientes pasos variaría según el que nosotros le demos. Para el caso de la suma binaria, se seleccionan los datos presentes en las listas anteriormente mencionadas e iríamos recorriendo cada una según su índice, mientras las sumábamos, y demás procesos, dicha operación se realizaría en una función llamada “suma\_binaria”. Para el caso de la suma real, se realiza un proceso parecido al anterior, pero con algunos cambios en su estructura, dichos procesos se almacenarían en una función llamada “suma\_real”. Luego de realizar ambas sumas, se seguiría con la siguiente función, que se encargaría de tomar los valores binarios, para transformarlos en valores del tipo decimal y así poder realizar los siguientes pasos. Ahora comenzaríamos con el tópico de resultados, específicamente los errores solicitados, que tampoco requieren un orden específico y dependen del que nosotros como programadores le demos. Para el caso de los errores, procederíamos a crear su respectiva función e ir de a poco ejecutando las fórmulas dadas por el docente para obtener cada uno de los casos, además de una métrica adicional, que en nuestro caso fue el error cuadrático. Finalmente, mostraríamos cada uno de los errores comparados entre las dos listas anteriormente realizadas. Todos estos pasos se verán a mayor profundidad y con mejores explicaciones más adelante.

Nuestro objetivo en este caso es de, aparte de reforzar los contenidos del ramo en formación, darnos una visión de cómo es que interactúan los errores de medición de valores, en especifico la comparativa de valores reales con valores obtenidos de otro tipo de procedimiento, que en nuestro caso fue el tema de los números binarios.

Summary

The problem planted by the professor consisted of some instructions that we had to realize on Python, a programming language. First, we had to generate two lists, each them filled with random float64-type numbers, which we had to use later to turn them into binary numbers. Then, with each one of the modified lists we should obtain the values of “suma binaria”, that would correspond to the sum of both lists with binary numbers, and “suma real”, that would correspond to the sum of both lists with real numbers. After that, we would pass the binary numbers to decimal numbers to go through the next step. Finally, with that values obtained, we would generate results that show up the different obtained errors or metrics, with its respective graphics to have a better visual observation of the data.

The above correspond to a rough explanation of what are we going to do; if we go to a deep step to step analysis of the topics that we are going to complete, we have the next: The main thing will be to create two list type variables on which to work (using the NumPy library to help us in the work) with names of “operando\_1” and “operando\_2”, respectively, of 80 or more values, give them random floating data and go on to the next step. From the previous values, we will generate its transformation to binary type numbers thanks to a function that will return two new lists, that we will name “operando\_1\_binario” and “operando\_2\_binario” respectively. The order of the next steps will vary according to the one we give to it. In the case of binary sum, we select the presented data on the previous lists and we would go through each one according to its index, while we sum it, and other processes, that operation would be realized on a function named “suma\_binaria”.   
In the case of the real sum, a process similar to the previous one is carried out, but with some changes in its structure, these processes would be stored in a function called "suma\_real". After performing both sums, we would continue with the following function, which would oversee taking the binary values, to transform them into decimal type values ​​and thus be able to carry out the following steps.   
Now we would start with the topic of results, specifically the requested errors, which also do not require a specific order and depend on what we as programmers give to it.   
In the case of errors, we would proceed to create their respective function and gradually execute the formulas given by the teacher to obtain each of the cases, as well as an additional metric, which in our case was the quadratic error. Finally, we would show each of the errors compared between the two lists previously made. All these steps will be seen in more depth and with better explanations later.

Our objective in this case is to, apart from reinforcing the contents of the class, give us a vision of how the errors of measurement of values interact, in specific the comparison of real values with values obtained from another type of procedure, which in our case was the subject of binary numbers.

Introducción

Lo propuesto en el informe expondrá los resultados de nuestro trabajo, hacienda uso de matemáticas aplicadas, manejo de listas, funciones, etc. Pretendemos que nuestra solución sea entendible en su totalidad, para ello es que pasaremos a explicar los puntos principales

Los ejes temáticos están ordenados de esta manera en el informe:

Resumen: Se da una explicación breve del problema en cuestión y de como se abordará en palabras simples

Introducción: Se presenta el contexto a grandes rasgos y la presentación de lo que se hablará a continuación

Marco teórico: Se buscará información de fuentes externas para ayudar a comprender las herramientas de software utilizadas en la solución de la problemática

Desarrollo: Una explicación mucho mas detallada de como se realizó el código y los métodos empleados para abarcar todos los puntos necesarios

Resultados: Se mostrarán los resultados obtenidos del trabajo, con sus respectivos gráficos

Conclusiones: A partir de los resultados, se ilustrarán los datos que se pueden rescatar del trabajo y las impresiones del trabajo realizado

Marco teórico

Para entender mejor el contexto del código e informe, definiremos ciertos conceptos a tener en cuenta al momento de la lectura de acuerdo a las herramientas utilizadas en el programa solución.

Python (<https://www.python.org>): Python es un lenguaje de programacióninterpretado de tipado dinámico cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas.

Dicho de otro modo, Python es:

* Interpretado: Se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
* Multiparadigma: Soporta programación funcional, programación imperativa y programación orientada a objetos.
* Tipado dinámico: Las variables se comprueban en tiempo de ejecución.
* Multiplataforma: disponible para plataformas de Windows, Linux o MAC.
* Gratuito: No dispone de licencia para programar. (Programo Ergo Sum, 2020)

Numpy (<https://numpy.org>):  Es la librería por excelencia para computación científica en Python. Trae integradas muchas funciones de cálculo matricial de N dimensiones, así como la transformada de Fourier, múltiples funciones de álgebra lineal y varias funciones de aleatoriedad. Puede ser utilizada como una herramienta estadística para realizar cálculos sobre, por ejemplo, varios archivos Excel a la vez. Además, otras muchas librerías científicas (por ejemplo, Pandas) se basan en Numpy para sus cálculos matemáticos. (Lopez, 2019)

Numpy nos proporciona un objeto ndarray que podemos usar para realizar una gran cantidad de operaciones en un arreglo de cualquier dimensión. En si, esta biblioteca tiene un gran manejo en lo que a listas se refiere, dándonos acceso a operaciones de alto rendimiento, como las que usaremos mas adelante.

Matplotlib (<https://matplotlib.org>): Matplotlib es una librería de Python que permite realizar gráficas 2D de excelente calidad. Es multiplataforma y puede ser usada desde scripts o desde la consola de Python. Matplotlib fue creado tomando como base a Matlab por lo que para las personas que han graficado alguna vez en su vida con Matlab, Matplotlib les va a resultar muy familiar. Podemos exportar nuestras gráficas en los formatos de imágenes más populares e incluso a formato látex para su inclusión en artículos científicos. (Raerpo, 2011)

En si esta librería nos permite como se explica arriba, dar un modelo gráfico de datos que hayamos obtenido en el desarrollo del código, en nuestro caso recibiría datos obtenidos de funciones anteriores y arrojará sus gráficas correspondientes

Desarrollo

El código implementado como solución constó de los siguientes pasos en su codificación, también hay comentarios en el programa para facilitar el entendimiento de este.

Para empezar, creamos una función llamada listas() que nos devolviera listas con números aleatorios de la biblioteca numpy, en específico el comando np.random.randn que fue facilitado por el profesor en clases

Luego, hicimos uso de la función floatbin() para transformar los números que están dentro de las primeras listas en números binarios, parte por borrar elementos “0b” y “.” que arroja la biblioteca numpy y después fue generando los números binarios. Seguidamente, con esos números, pasamos a llenar las listas con los números obtenidos mediante las funciones de binarizacion(), donde hicimos una por cada lista a llenar. Estas funciones reciben como parámetros las primeras dos listas con números aleatorios, las recorren, realizan su correspondiente trabajo y devuelven dos nuevas listas ya completadas con su conversión binaria.

Mas adelante, pasamos a la parte de las sumas. En primer lugar, realizamos la suma binaria, la cual tomaba ambas listas ya convertidas en binario y mediante un sistema de recorrido de listas las sumaba según su índice, y luego nos devuelve la lista con los datos dentro. Para la suma real, hicimos un método parecido, solo que, con un nivel menor de complejidad respecto a la anterior, ésta también nos devuelve una lista con los valores en ella.

Entrando al tema de los errores, lo primero que hicimos fue transformar los valores de la suma binaria a números decimales, para ello hicimos uso de la función conversion() que gracias al uso de bucles, nos devuelve una lista con valores anteriormente binarios transformados a decimales. Ahora pasando a los tres errores en cuestión que utilizamos (absoluto, relativo, cuadrático), el proceso fue casi el mismo para los 3, en lo que varían realmente es en la fórmula para llegar a ellos. El error absoluto resta la lista de valores reales en valor absoluto con la lista de valores obtenidos (binario a decimal). El error relativo hace lo mismo que arriba, pero en este caso al resultado se le divide por los valores de la lista real. En el caso del error cuadrático, este recibe el valor del error absoluto, lo recorre y lo aumenta hasta su potencia de dos, que sería la fórmula para calcularlo.

Luego de esto, tenemos la función plot, que hace uso de la biblioteca matplotlib y la cual usamos para graficar los resultados que se verán más adelante, dicha función fue facilitada por el docente en este caso.

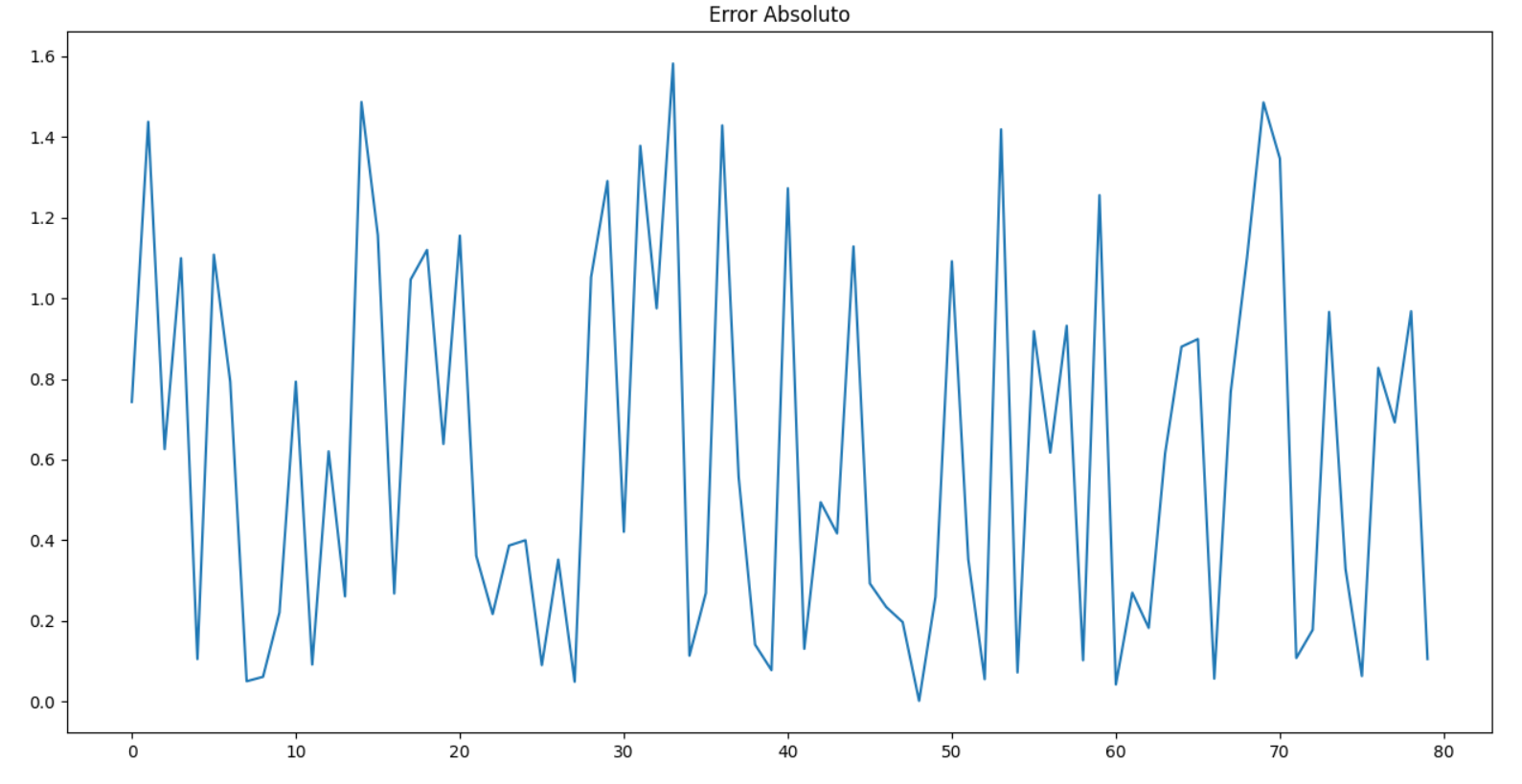
Finalmente, en la parte de ejecución del código, una vez que ya tenemos todas las funciones, simplemente le dimos los parámetros correspondientes para que el código funcionara con normalidad, mediante prints, fuimos mostrando la información en pantalla para evitar errores.

Resultados

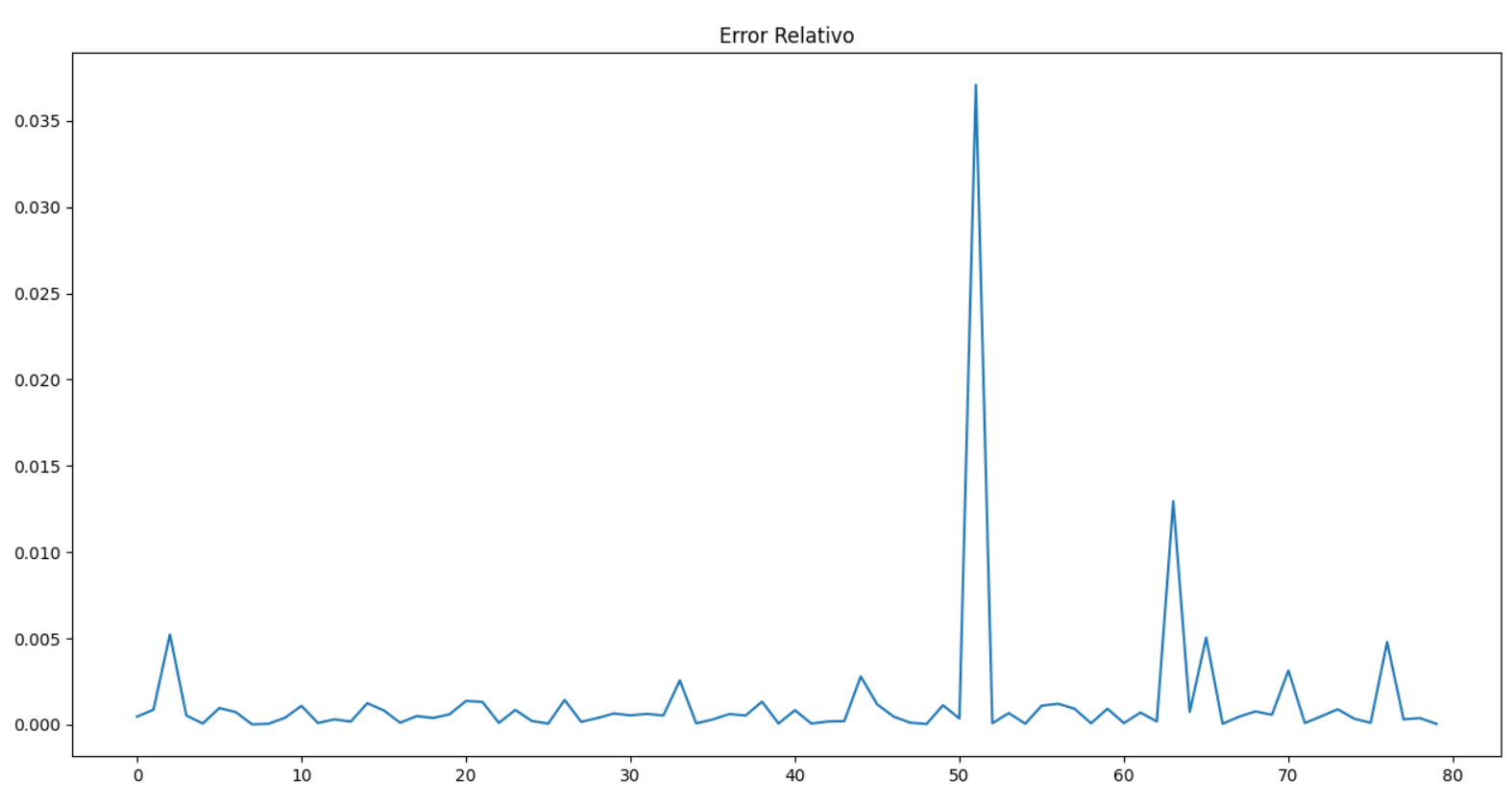
A continuación, se mostrarán los distintos gráficos generados gracias a la biblioteca matplotlib, que nos muestran de manera visual como es que se comportan las diferentes métricas que se emplearon en el desarrollo del problema.

Los datos fueron tomados directamente de las listas que se fueron generando a medida que realizamos el código, y se implementaron en la función plot() facilitada por el profesor

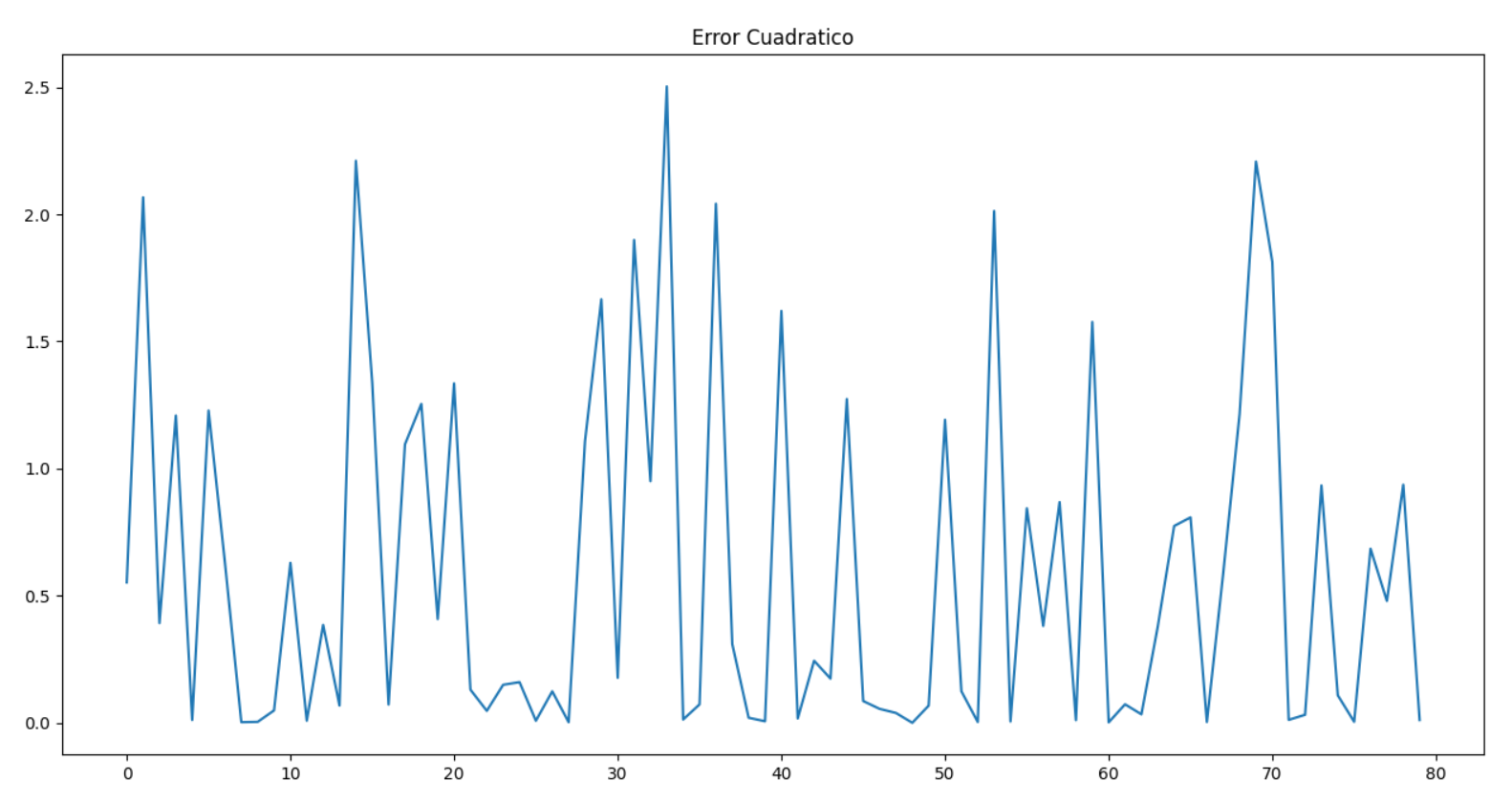
**Error absoluto**



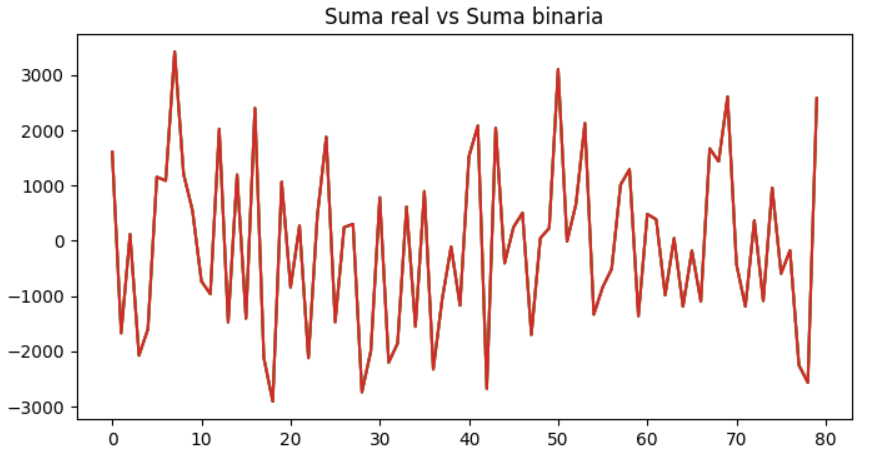
**Error relativo**



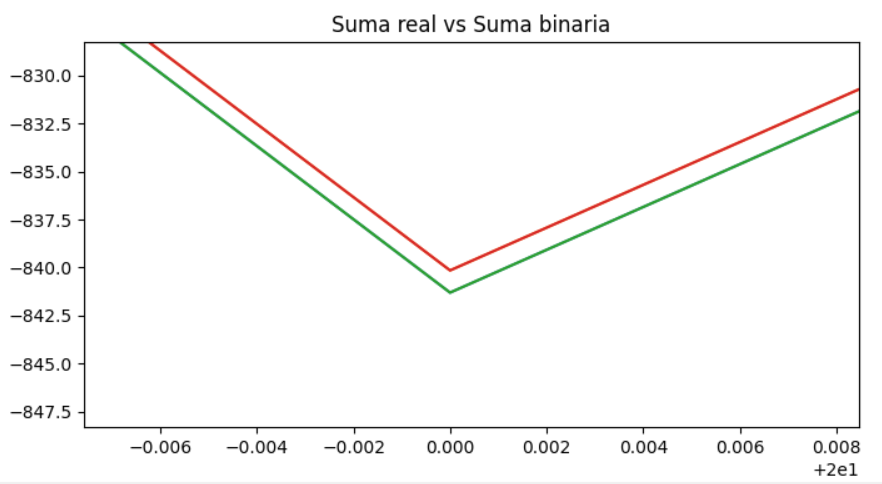
**Error cuadrático**



**Suma real v/s suma binaria**



En este caso es necesario hacer un zoom profundo para notar las diferencias entre cada suma, al realizarlo tenemos por ejemplo un caso como este



Conclusiones

El desarrollo del algoritmo si fue idéntico al trabajo en Python, utilizamos todas las herramientas facilitadas por el profesor para poder llegar a la solución, además de otras que habíamos usado con anterioridad.

En cuanto a los resultados, son los que se esperaban, podemos verlo reflejado en los gráficos, donde cada vez que ejecutamos el código daba resultados similares, lo que nos da confianza de que si era lo que nosotros creíamos desde el principio.

Personalmente, la parte que mas nos costó realizar fue el caso de la suma binaria, donde tuvimos que investigar mas a fondo para dar respuesta a esa parte, y también al momento de crear la lista binaria, tuvimos que indagar un poco con respecto a los caracteres binarios especiales (como el “0b”). Fuera de eso no tuvimos mayores inconvenientes para realizar el trabajo gracias a las herramientas del propio lenguaje

Si algo pudiéramos mejorar del modelo, sería su diseño quizás, en su estética se ve algo pobre, pero cumple totalmente con lo que se solicitó, así que no sería mas que un detalle. O bien, reducir un poco de líneas de código, que nos permita un procesamiento mas rápido, pero en este tipo de casos (con relativamente pocas líneas de código) no lo vemos como algo muy necesario.

El trabajo en sí cumplió con todos los objetivos requeridos, pasando desde el manejo de listas hasta la muestra de resultados, cosas que son imprescindibles dentro del manejo del lenguaje, pero sobre todo, se logró afianzar los conceptos de errores absoluto, relativo, cuadrático, y como es que los valores punto flotante actúan sobre ellos y como se comportan.

# Referencias

Lopez, R. (07 de 06 de 2019). *raullg.com*. Obtenido de https://raullg.com/wiki/que-es-numpy/

Programo Ergo Sum. (08 de 10 de 2020). *programoergosum.com*. Obtenido de https://www.programoergosum.com/cursos-online/raspberry-pi/244-iniciacion-a-python-en-raspberry-pi/que-es-python

Raerpo. (27 de 12 de 2011). *raerpo2.blogspot.com*. Obtenido de http://raerpo2.blogspot.com/2011/12/que-es-matplotlib.html