Instituto Tecnológico Superior Progreso

Tarea: 1.1

Tema: Introducción a los métodos numéricos

Asignatura: Métodos Numéricos **Calificación:** _____

Profesor: Dr. Iván de Jesús May Cen **Fecha:** 5 de febrero del 2025

Estudiante: Antonio Josue Rodriguez Falcon **Carrera:** ISIC

Reporte de la Tarea 1.1: Calculo de errores.

Ejercicio 1: Precisión de la representación numérica

Explicación de los Resultados

Este código tiene como objetivo determinar la precisión de máquina, es decir, el mínimo valor positivo que al sumarse con 1.0 en la computadora sigue produciendo un resultado distinto de 1.0. y los puntos son:

- Primer punto, se inicializa una variable épsilon con el valor de 1.0.
- Segundo punto, en cada iteración, épsilon se divide entre 2 y se comprueba si 1.0 + épsilon sigue siendo distinto de 1.0.
- Tercer punto, es cuando 1.0 + épsilon es igual a 1.0, significa que épsilon ha alcanzado el límite de precisión de la computadora y, por lo tanto, se revierte la última división para obtener el valor correcto.

El resultado final representa la precisión más pequeña con la que los números pueden ser representados en punto flotante en la computadora.

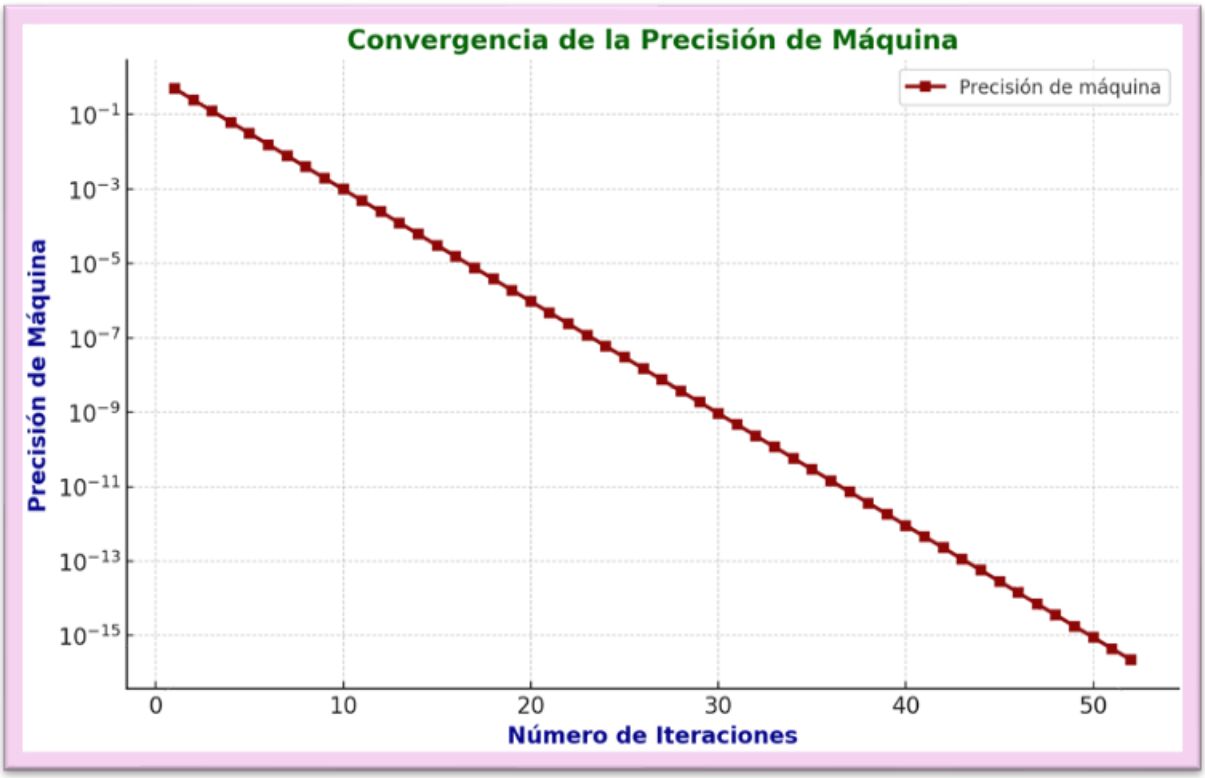
Tablas y Gráficas Obtenidas

Iteración	Precisión de máquina

1	0.5
2	0.25
3	0.125
4	0.0625
5	0.03125
6	0.015625
7	0.0078125
8	0.00390625
9	0.001953125
10	0.0009765625
11	0.00048828125
12	0.000244140625
13	0.0001220703125
14	6.103515625e-05
15	3.0517578125e-05
16	1.52587890625e-05
17	7.62939453125e-06
18	3.814697265625e-06
19	1.9073486328125e-06
20	9.5367431640625e-07
21	4.76837158203125e-07
22	2.384185791015625e-07
23	1.1920928955078125e-07
24	5.960464477539063e-08
25	2.9802322387695312e-08
26	1.4901161193847656e-08
27	7.450580596923828e-09
28	3.725290298461914e-09

29	1.862645149230957e-09
30	9.313225746154785e-10
31	4.656612873077393e-10
32	2.3283064365386963e-10
33	1.1641532182693481e-10
34	5.820766091346741e-11
35	2.9103830456733704e-11
36	1.4551915228366852e-11
37	7.275957614183426e-12
38	3.637978807091713e-12
39	1.8189894035458565e-12
40	9.094947017729282e-13
41	4.547473508864641e-13
42	2.2737367544323206e-13
43	1.1368683772161603e-13
44	5.684341886080802e-14
45	2.842170943040401e-14
46	1.4210854715202004e-14
47	7.105427357601002e-15
48	3.552713678800501e-15
49	1.7763568394002505e-15
50	8.881784197001252e-16
51	4.440892098500626e-16
52	2.220446049250313e-16
53	1.1102230246251565e-16

Tabla 1.1 Tabla de Precisión de la máquina.



Grafica 1.2 Grafica de la Precisión de Maquina vs la Iteración.

Análisis de los Errores Encontrados

En este ejercicio ilustra las limitaciones en la representación numérica de punto flotante en el código. Algunos de los errores más comunes son:

- Error de Redondeo: Los números en punto flotante tienen una cantidad finita de bits, lo que puede llevar a redondeos imprecisos en cálculos matemáticos.

- Error de Cancelación: Cuando dos números muy cercanos se restan, la precisión puede perderse debido a la representación finita de los números.

- Error de Acumulación: En iteraciones sucesivas, los errores de redondeo pueden acumularse y afectar la precisión final de los cálculos.

En conclusión, este código permite visualizar las limitaciones inherentes a los sistemas de punto flotante y es clave para el diseño de algoritmos numéricos robustos.

Código: Tarea 1.1 Calculo de Errores GitHub

Ejercicio 2: Cálculo del error absoluto, relativo y cuadrático en una aproximación

Explicación de los Resultados

El ejercicio 2 implementa la serie de Leibniz para aproximar el valor de π mediante una suma alternante. La expresión que se utiliza es:

$$\pi = 4\sum_{k=0}^{n}\frac{(-1)^k}{2k+1}$$

Donde n representa el número de términos considerados en la aproximación. A medida que aumenta, la aproximación se vuelve más precisa. Sin embargo, la convergencia de esta serie es relativamente lenta.

El código compara el valor obtenido con el valor real de la biblioteca NumPy y calcula los siguientes errores:

- Error absoluto: Diferencia entre el valor real y el aproximado.
- Error relativo: Proporción del error absoluto con respecto al valor real.
- Error cuadrático: Cuadrado del error absoluto, lo que da mayor peso a errores mayores.

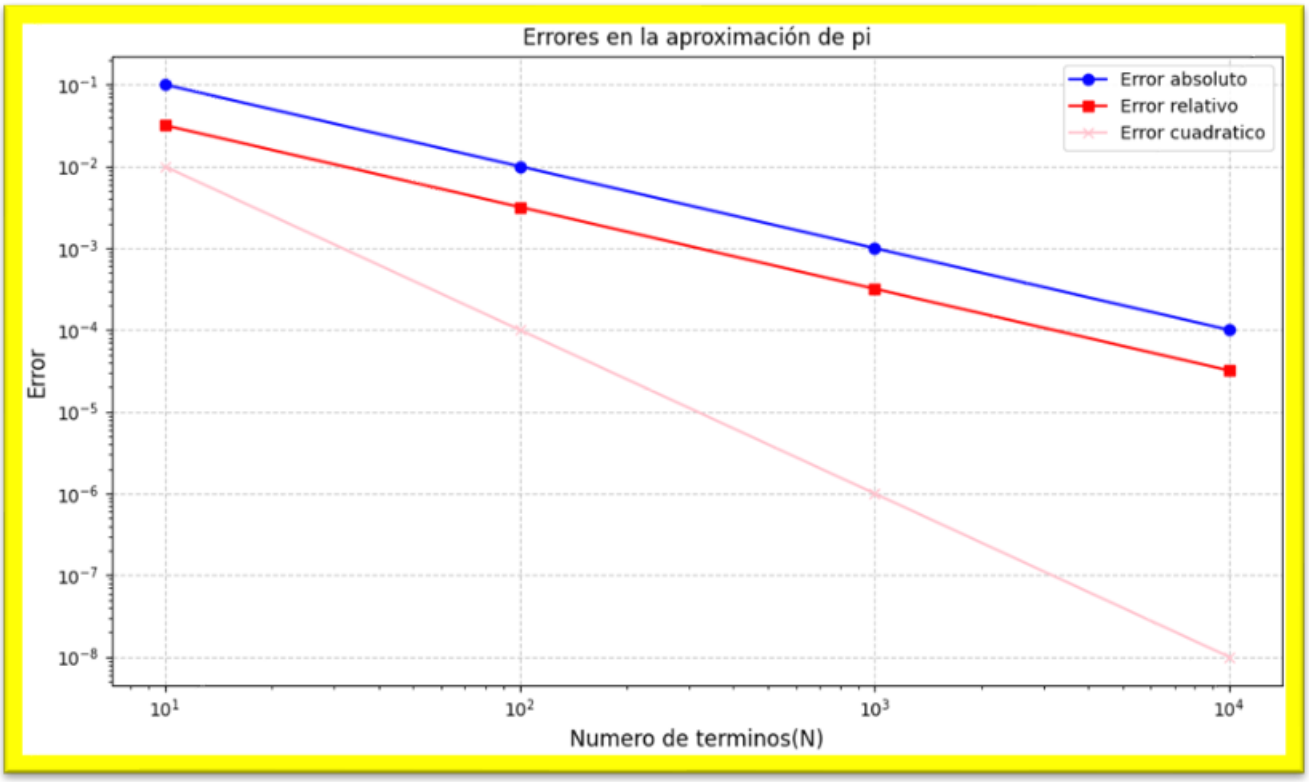
Se analizo valores para evaluar la tendencia de estos errores a medida que aumenta el número de términos en la serie.

Tablas y Gráficas Obtenidas

Tabla 2.1 Tabla de los errores (Absoluto, Relativo y Cuadrático)

Numero de términos (N)	Error Absoluto	Error Relativo	Error Cuadrático
10	0.09975303466038987	0.03175237710923643	0.009950667923956944
100	0.00999975003123943	0.00318301929431018	9.999500068727297e-05
1000	0.000999999749998981	0.0003183098066059948	9.999994999980246e-07
10000	9.99999997586265e-05	3.18309885415475e-05	9.9999999517253e-09

Grafica 2.2 Grafica de los errores en la aproximación de pi



Análisis de los Errores Encontrados

- Tendencia de convergencia: La gráfica muestra una disminución clara de los errores conforme aumenta el número de términos. Sin embargo, la velocidad de convergencia es lenta, ya que incluso con, el error absoluto sigue presente en el cuarto decimal.

- Error absoluto: Se observa que decrece exponencialmente con, lo que indica que agregar más términos mejora la precisión de la aproximación.
- Error relativo: Sigue la misma tendencia del error absoluto y confirma que el cálculo se vuelve más preciso con un mayor.
- Error cuadrático: Como se esperaba, sigue la misma tendencia de los otros errores, pero acentúa las desviaciones más grandes.

Conclusión

Este análisis confirma que la serie de Leibniz es una aproximación válida de , pero su convergencia es demasiado lenta para aplicaciones que requieren alta precisión en pocos cálculos. Para obtener resultados más exactos con menor esfuerzo computacional, se recomienda el uso de métodos alternativos como la serie de Machín o la serie de Gauss-Legendre.

Código: Tarea 1.1 Calculo de Errores GitHub

Ejercicio 3: Errores en operaciones numéricas

Explicación de los Resultados

Este código realiza el cálculo de diversos errores en operaciones numéricas y los representa mediante gráficos. Utiliza dos conjuntos de datos $(x, y, valor\_real)$ y, para cada caso, determina:

- Diferencia: Resultado de restar $x - y$.
- Error absoluto: Distancia entre el valor real y la diferencia, es decir, $|valor\_real - diferencia|$.
- Error relativo: Se obtiene dividiendo el error absoluto entre el valor real: $(error\ absoluto)\ /\ |valor\_real|$.

- Error porcentual: Se expresa en porcentaje multiplicando el error relativo por 100: $(error\ relativo)$ × 100.
- Error cuadrático: Se calcula elevando al cuadrado el error absoluto: $(error\ absoluto)^2$.

Los valores obtenidos se almacenan en listas y posteriormente se visualizan con gráficos de distintos tipos:

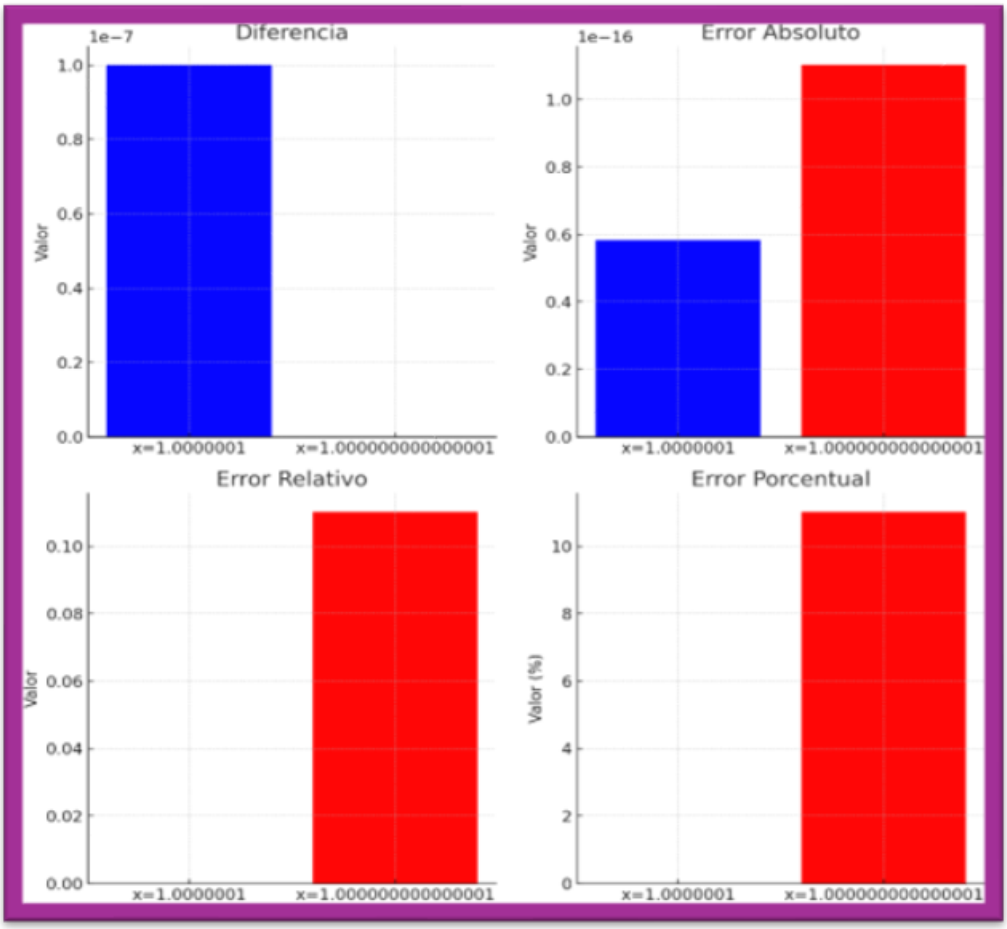
- Gráfico de barras: Representa el error absoluto y el error porcentual.
- Gráfico de líneas: Muestra la variación del error relativo y el error cuadrático.

Tablas y Graficas Obtenidas

Tabla 3.1 Tabla de errores para los diferentes valores de x

x	Diferencia	Error Absoluto	Error Relativo	Error Porcentual	Error Cuadrático
1.0000001	1.0000000005838672e-07	5.838672220806777e-17	5.838672220806777e-10	5.838672220806777e-08%	5.838672220806777e-17
1.00000000000001	1.1102302462515650e-15	1.1022302462515646e-16	0.11022302462515646	11.022302462515645%	1.1022302462515646e-16

Grafica 3.2 Graficas de los diferentes errores para los distintos valores de x



Análisis de los Errores Encontrados

En este código se encontró los errores como:

- La coherencia en los valores utilizados para calcular la diferencia:
  la operación $x - y$ puede no reflejar con precisión el error en relación con ***valor_real***, dado que la diferencia esperada no siempre coincide con este último. Para mejorar la evaluación del error, sería más adecuado considerar su relación con x o y en lugar de depender únicamente de ***valor_real***.

- El manejo optimizado de valores extremadamente pequeños
  Cuando ***valor_real*** es demasiado pequeño (por ejemplo, 0.000000000000001), el cálculo del error relativo puede resultar en valores

desproporcionadamente altos, generando posibles errores numéricos en divisiones. Una posible solución sería implementar una verificación que prevenga divisiones por números extremadamente pequeños.

Código: Tarea 1.1 Calculo de Errores GitHub