Miscelánea Matemática **80** (2025) 21-33 SMM

DOI: https://doi.org/10.47234/mm.8003

¿Qué tan preciso es el cálculo de un promedio con números redondeados?

Juan Carlos Aguilar
Departamento Académico de Matemáticas
ITAM
aguilar@itam.mx

1. Introducción

Cuando hacemos algún cálculo aritmético en una calculadora o computadora (suma, resta, multiplicación, división, raíz cuadrada, porcentajes, etc), normalmente damos por sentado que el resultado que despliega la calculadora o computadora es correcto. En ocasiones, la respuesta obtenida en la calculadora no es confiable o tiene poca precisión; esto se debe a que la calculadora o computadora hace redondeos en los números y operaciones con números a cierto número de cifras. Tomemos un ejemplo: sabemos que $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$. Cuando en la expresión $\left(1+\frac{1}{n}\right)^n$, hacemos $n = 100$, $n = 1000$, en una calculadora que hace redondeo de números a 8 cifras decimales, obtenemos resultados que se van acercando al valor del número e. Pero cuando $n = 10^8$, el resultado que obtenemos al evaluar la expresión es 1, es decir, ninguna cifra correcta del número e. ¿ A qué se debe esto? Muy sencillo: al tomar $n = 10^8$ o mayor, el número $1+\frac{1}{10^8} = 1.00000001$ queda redondeado a 8 cifras decimales como 1.0000000 =1, por lo que la calculadora calculará $1^n = 1$.

Para el manejo de una gran cantidad de datos numéricos (por ejemplo 100 millones de números que representan las edades de personas en un país, o números seudoaleatorios para métodos de Monte Carlo), cálculos tan elementales e importantes como el promedio pueden perder algunas o todas las cifras de precisión si se hacen con redondeo a 8

Palabras clave: suma por parejas, suma acumulada, promedios, pérdida de precisión, redondeo de números.

cifras decimales (o en precisión sencilla disponible en algunos lenguajes de programación) si la suma del promedio es calculada con ciertos algoritmos. Nos enfocaremos al caso pérdida de precisión al calcular el promedio o sumas de una gran cantidad de números positivos.

2. Redondeo de números a p cifras decimales y notación

Daremos por hecho que las lectoras y lectores conocen el concepto de redondeo de números escritos en forma decimal (véase [4]). Escribiremos $fl_p(a)$ para denotar el redondeado de a a p cifras decimales.

Ejemplos:

a) $fl_4(34.215) = 34.22$, **b**) $fl_4(34.214) = 34.21$,
c) $fl_6(2/3) = 0.666667$, **d**) $fl_4(5.29999999) = 5.300 = 5.3$,
e) $fl_8(0.002300555555555) = 0.23005556 \times 10^{-2}$.

En el último ejemplo, la primera **cifra significativa** es 2, y los ceros a la izquierda del 2 no cuentan en el redondeo (pero sí cuenta para escribir el número en notación científica).

Cuando aproximamos un número $a \neq 0$ con otro número b, usamos el concepto de **error relativo de aproximación**, definido como $\dfrac{|a-b|}{|a|}$. Cuando $\dfrac{|a-b|}{|a|} \approx 10^{-q}$, se traduce en que a y b se «parecen» en aproximadamente q cifras decimales significativas. Por ejemplo, si $a = 0.0054371893444555$ y $b = 0.0054371452222222$, entonces $\dfrac{|a-b|}{|a|} \approx 8.11 \times 10^{-6} \approx 10^{-5}$, por lo que b aproxima a a en aproximadamente cinco cifras significativas (los ceros la izquierda no cuentan como cifras significativas).

3. Pérdida de precisión al calcular promedios

El cálculo de una suma de m números positivos $a_1 + a_2 + \cdots + a_m$ puede producir pérdida de precisión cuando los sumandos y sumas se redondean, y m es grande. La pérdida de precisón en este caso depende del algoritmo que se use para calcular la sumatoria. El siguiente ejemplo muestra que el cálculo de un promedio (o suma) de m números puede perder hasta r cifras significativas de precisión si $m \approx 10^r$ y si se calcula

acumulando las sumas:

Ejemplo. Calcular el promedio de $m = 10^8 = 100$ millones de números, $a_1, a_2, \ldots, a_m$, cada número es constante e igual a 1/3. Usar redondeo a 8 cifras decimales.

Para calcular el promedio, primero calculamos la suma $a_1 + a_2 + \cdots + a_m$. Un algoritmo sencillo y muy usado es el siguiente (al que llamaremos **suma acumulada**): en el primer paso, se define $s_1 = a_1$. Luego se calculan s_2, s_3,...,s_m como $s_{j+1} = s_j + a_{j+1}$, para $j = 1, 2, \ldots, m-1$. Entonces s_m es el valor de la suma buscada. Finalmente se calcula el promedio dividiendo s_m entre m. ¿En qué parte se pierde precisión?

Recordemos que en el algoritmo **suma acumulada** anterior, los sumando a_j y las sumas s_j se redondean a 8 cifras decimales, por lo que en realidad $s_1 = fl_8(a_1)$ y $s_{j+1} = fl_8(s_j + fl_8(a_{j+1}))$. En el paso $j = 60$ millones, la variable s_j ha calculado de manera aproximada $a_1 + a_2 + \cdots + a_{60000000}$. Notar que el valor exacto de dicha suma es 60000000/3=20000000. Aún si la variable s_j en dicho paso hubiera calculado de manera exacta dicho valor, en el siguiente paso se calcula $s_{j+1} = fl_8(20000000 + 0.33333333) = fl_8(20000000.33333333) = 20000000$. Por lo que en presencia de redondeo a 8 cifras, se tendría: $s_j = s_{60000000} = 20000000$ para $j = 60000001, \ldots, 100000000$ Es decir, la suma acumulada ya no acumula más sumando a partir del sumando 60 millones, por lo que los últimos 40 millones de sumandos no son tomados en cuenta. Entonces el promedio sería calculado como 20000000/100000000=1/5=0.2, en lugar de 0.33333333 (ninguna cifra correcta en el cálculo del promedio). La situación es aún peor debido a que la suma de los primeros 60000000 de términos no es calculada de manera exacta.

¿Cómo remediar la situación anterior? El problema del algoritmo anterior radica en que al calcular la suma de los primeros 60 millones de sumandos, el resto de los sumandos son demasiado pequeños como para seguir contribuyendo a la suma acumulada; como quedan 40 millones de términos por sumar, que en su conjunto suman una cantidad importante para el promedio, se pierde precisión. Para evitar la situación anterior, supongamos por un momento que la suma Σ_1 de los primeros 50 millones de sumandos es calculada de una manera «aceptable», y que la suma Σ_2 de los restantes 50 millones de sumandos también. Entonces la suma total $\Sigma_1 + \Sigma_2$ es calculada de una manera «aceptable». Pero para calcular Σ_1 con una precisión «aceptable», no podemos calcularla con el algoritmo **suma acumulada** ya que se pierde precisión. En su lugar, para el cálculo de Σ_1, los 50 millones de sumandos los separamos en 2 partes de 25 millones de sumandos cada una; se hace lo

mismo para el cálculo de Σ_2. El cálculo de una suma de 25 millones de términos con el algoritmo **suma acumulada** puede ser poco preciso en aritmética de redondeo a 8 cifras decimales, así que cada uno de esos grupos de 25 millones de sumandos se separa en 2 grupos de 12.5 millones de sumandos, y así sucesivamente, hasta llegar a grupos de 1 o 2 sumandos.

Para ilustrar la idea anterior, supongamos que deseamos calcular la suma $s = a_1 + a_2 + a_3 + \cdots + a_8$ de 8 sumandos. Separamos la suma en 2 grupos $\Sigma_1 = a_1 + a_2 + a_3 + a_4$ y $\Sigma_2 = a_5 + a_6 + a_7 + a_8$. Ahora los sumandos de la suma Σ_1 se separan en dos grupos: $w_1 = a_1 + a_2$ y $w_2 = a_3 + a_4$; los sumandos de Σ_2 se separan en: $w_3 = a_5 + a_6$ y $w_4 = a_7 + a_8$. La suma buscada s de los 8 sumandos es calculada como sigue: se calculan w_1, w_2, w_3 y w_4 directamente como la suma de dos sumandos. Luego se calculan $\Sigma_1 = w_1 + w_2$, $\Sigma_2 = w_3 + w_4$. Finalmente, se calcula la suma total $s = \Sigma_1 + \Sigma_2$.

Esquemáticamente:

$$\underbrace{\underbrace{\underbrace{a_1}_{w_1 = a_1 + a_2} + \underbrace{a_2}_{}}_{\Sigma_1 = w_1 + w_2} + \underbrace{\underbrace{a_3}_{w_2 = a_3 + a_4} + \underbrace{a_4}_{}}_{} + \underbrace{\underbrace{a_5}_{w_3 = a_5 + a_6} + \underbrace{a_6}_{}}_{\Sigma_2 = w_3 + w_4} + \underbrace{\underbrace{a_7}_{w_4 = a_7 + a_8} + \underbrace{a_8}_{}}_{}}_{s = \Sigma_1 + \Sigma_2}$$

Si el número de sumandos no es potencia de 2, podría haber un sumando sin «pareja». Para ejemplificar, supongamos que deseamos calcular una suma con 11 términos: $s = a_1 + a_2 + a_3 + \cdots + a_{11}$. Esquemáticamente:

$$\underbrace{\underbrace{\underbrace{\underbrace{a_1}_{w_1 = a_1 + a_2} + \underbrace{a_2}_{}}_{\beta_1 = w_1 + w_2} + \underbrace{\underbrace{a_3}_{w_2 = a_3 + a_4} + \underbrace{a_4}_{}}_{} + \underbrace{\underbrace{a_5}_{w_3 = a_5 + a_6} + \underbrace{a_6}_{}}_{\beta_2 = w_3 + w_4} + \underbrace{\underbrace{a_7}_{w_4 = a_7 + a_8} + \underbrace{a_8}_{}}_{}}_{\Sigma_1 = \beta_1 + \beta_2} + \underbrace{\underbrace{\underbrace{a_9}_{w_5 = a_9 + a_{10}} + \underbrace{a_{10}}_{}}_{\beta_3 = w_5 + w_6} + \underbrace{\underbrace{a_{11}}_{w_6 = a_{11}}}_{}}_{\Sigma_2 = \beta_3}}_{s = \Sigma_1 + \Sigma_2}$$

Al algoritmo anterior le llamaremos **suma por parejas** (en inglés es conocido como *pairwise summation* o como *cascade sum*, véase [3]).

Para que la pérdida de precisión del algoritmo **suma acumulada** sea notable, el número de sumandos debe ser significativamente grande: si el número de sumandos es $m = 2 \times 10^6$, con sumandos constantes redondeados a 8 cifras decimales, $a_j = 0.33333333$, $j = 1, 2, \ldots, m$, la suma acumulada (exacta y redondeada) de los primeros 10^6 sumandos es $10^6(0.33333333) = 333333.33$, que al añadirle un sumando más, se obtiene $fl_8(333333.33 + 0.33333333) = fl_8(333333.66333333) = 333333.66$, lo que significa que la suma acumulada de los primeros 10^6 sumandos, únicamente «verá» las primeras 2 cifras significativas del

resto de los sumandos, por lo que la suma acumulada total perderá 6 cifras significativas de precisión de las 8 cifras. Si se usa redondeo a p cifras decimales para calcular una suma de $m \approx 10^r$ sumandos constantes e iguales a $1/3$, se pueden perder hasta r cifras significativas de precisión de las p cifras. En general, para el cálculo de promedios de números positivos no necesariamente constantes, $a_1, a_2, \ldots, a_m$, se puede perder poca o mucha precisión con **suma acumulada** dependiendo de algunas variables como: redondeo de los sumandos, el número y distribución de los sumandos. El ejemplo que hemos discutido indica qué tanta precisión se puede perder.

¿Por qué **suma por parejas** pierde poca precisión? Si analizamos el algoritmo vemos que se evita hacer sumas acumuladas con muchos términos, que es donde se puede perder precisión. Si el número de sumandos m es muy grande, el algoritmo de **suma por parejas** hace sumas acumuladas de aproximadamente $\log_2(m)$ términos. Si m es enorme, por ejemplo $m = 10^{20} \approx 2^{66.44}$, entonces el algoritmo calculará sumas acumuladas de aproximadamente 67 términos. Como $67 < 100 = 10^2$, se pueden perder aproximadamente 2 cifras de precisión.

4. Ejemplos numéricos

En los siguientes ejemplos usaremos MATLAB 2023b, Python 3.12.4, y Octave 9.2.0 para hacer los cálculos de sumas y promedios. Estos lenguajes de programación incluyen dos instrucciones para calcular promedios o sumas: la instrucción **sum** y la instrucción **mean**. Si x es un arreglo de m números, con la instrucción **sum** se calcula la suma de las entradas de x; si se desea calcular un promedio, la suma calculada se divide entre m. La instrucción **mean** calcula el promedio de las entradas de x; si se desea calcular la suma de las entradas del arreglo x, el promedio se multiplica por m. En MATLAB y Octave se pueden usar ambas instrucciones con los tipos de variables numéricas: a) precisión sencilla (single) b) precisión doble (double).

La precisión sencilla es similar a hacer un redondeo a 8 cifras decimales; no es equivalente ya que en MATLAB, Octave y Python, los números se representan con cifras binarias en lugar de decimales (véase [2]). De la misma manera, la precisión doble es similar a hacer redondeo a 16 cifras decimales. Una ventaja de usar variables de precisión sencilla sobre doble, es que se usa menos memoria : una variable en precisión sencilla requiere 4 bytes en lugar de los 8 bytes de una variable de precisión doble, lo cual es una ventaja cuando se almacena una gran cantidad de datos en la memoria de la computadora. Una desventaja

es que, como veremos en los siguientes ejemplos, al calcular promedios o sumas de una gran cantidad de números (por ejemplo 10^8 números) con **suma acumulada**, se pueden perder hasta 8 cifras de precisión, y como consecuencia, el promedio calculado en precisión sencilla podría tener cero cifras significativas correctas. En cambio, en precisión doble, si el cálculo tiene 8 cifras incorrectas, aún quedan 8 cifras correctas ya que precisión doble usa aproximadamente redondeo a 16 cifras decimales. Por otra parte, si el promedio es calculado en precisión sencilla con **suma por parejas**, el cálculo perderá poca precisión aunque el número de sumandos sea enorme (10^{20}, por ejemplo), por lo que vale la pena usar precisión sencilla en combinación con algoritmos apropiados en el cálculo de promedios.

En Python las instrucciones **mean** y sum se usan únicamente en precisión doble (si el arreglo x se define en precisión sencilla, dichas instrucciones primero convierten el arreglo x a precisión doble). Por lo que no incluiremos ejemplos con Python en precisión sencilla.

4.1 Ejemplos con precisión sencilla

En estos ejemplos se usará precisión sencilla para aproximar promedios o sumas con muchos términos. En el mejor de los escenarios se aproximarán los resultados exactos con 8 cifras significativas correctas. Calcularemos los promedios de 6 maneras: a) **suma acumulada**, b) con la instrucción **mean** de MATLAB y de Octave, c) con la instrucción **sum** de MATLAB y Octave, d) **suma por parejas**.

Ejemplo 1. El cuadro 1 muestra los resultados del cálculo del promedio de $\dfrac{a_1 + a_2 + \cdots + a_m}{m}$, $m = 10^9$, con $a_j = 1/3$ para toda j. Cada a_j se calcula en precisión sencilla. Si se usa la instrucción **sum**, el resultado se divide entre m. En este caso el promedio exacto redondeado a 8 cifras es: 0.33333333.

Método de cálculo	Valor calculado	No. de cifras significativas correctas
suma acumulada	0.0083886078	0
Octave, mean	0.**3333333**4	7
Octave, sum	0.0083886078	0
MATLAB, mean	0.**333**24367	3
MATLAB, sum	0.**333**24367	3
suma por parejas	0.**3333333**4	7

Cuadro 1

Notamos que Octave con las instrucción **sum** produce el mismo resultado que **suma acumulada**, que en este ejemplo da cero dígitos de precisión. Por otra parte, la precisión más alta se obtiene con la instrucción **mean** de Octave y con **suma por parejas**.

Ejemplo 2. Deseamos calcular el valor numérico de $s = \sum_{n=1}^{\infty} \frac{1}{n^2}$ con 7 u 8 cifras decimales significativas correctas, usando una suma parcial con precisión sencilla.

Para comenzar, debemos estimar cuántos sumandos de la serie debemos tomar para obtener la precisión deseada sin tomar en cuenta el redondeo. Necesitamos encontrar una suma parcial $s_m = \sum_{n=1}^{m} \frac{1}{n^2}$ que aproxime a s con un error relativo tal que $\frac{|s - s_m|}{|s|} < 10^{-8}$, es decir, $\frac{\sum_{n=m+1}^{\infty} \frac{1}{n^2}}{s} < 10^{-8}$. Para ello, tomando la función $f(x) = 1/x^2$, $x > 0$, y tomando $x \in \{m+1, m+2, m+3, \dots\}$ se tiene que $1/n^2 = f(n)$. Como f es decreciente y positiva, se sigue que (véase la figura 1), $\int_{m+1}^{\infty} \frac{1}{x^2} dx < \sum_{n=m+1}^{\infty} \frac{1}{n^2}$.

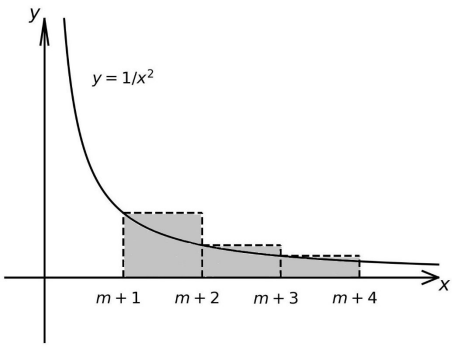

Figura 1. La suma de las áreas de los rectángulos es mayor que el área bajo la gráfica de $y = 1/x^2$, $x \in [m+1, \infty]$.

Similarmente (véase la figura 2), $\sum_{n=m+2}^{\infty} \frac{1}{n^2} < \int_{m+1}^{\infty} \frac{1}{x^2} dx$.

Dado que $\int_{m+1}^{\infty} \frac{1}{x^2} dx = \frac{1}{m+1}$, de las desigualdades anteriores se obtiene:

$$\frac{1}{m+1} < \sum_{n=m+1}^{\infty} \frac{1}{n^2} < \frac{1}{m+1} + \frac{1}{(m+1)^2}.$$

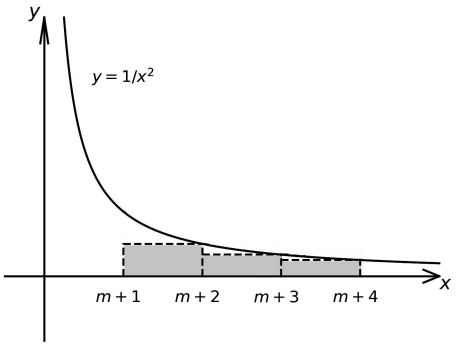

Figura 2. La suma de las áreas de los rectángulos es menor que el área bajo la gráfica de $1/x^2$, $x \in [m+1, \infty]$.

Además, $s = \sum_{n=1}^{\infty} \frac{1}{n^2} > 1$, y $s = 1 + \sum_{n=2}^{\infty} \frac{1}{n^2} < 1 + 1/2 + 1/4 = 7/4$, por lo que $1 < s < 7/4$. Por lo tanto,

$$\frac{4}{7(m+1)} < \frac{\sum_{n=m+1}^{\infty} \frac{1}{n^2}}{s} < \frac{1}{m+1} + \frac{1}{(m+1)^2}. \tag{1}$$

Usando la desigualdad (1), tomemos el mínimo entero positivo m tal que $\frac{1}{m+1} + \frac{1}{(m+1)^2} \leq 10^{-8}$. Para ello, sea $x = 1/(m+1)$. Las raíz positiva de la ecuación $x + x^2 = 10^{-8}$ es $x = \frac{-1 + \sqrt{1 + 4(10^{-8})}}{2} = \frac{2(10^{-8})}{1 + \sqrt{1 + 4(10^{-8})}} = 9.999999900000001..\times 10^{-9}$. Tomar $m = \lceil \frac{1}{x} - 1 \rceil = 10^8$. Cuando $m = 10^8$, se cumple $5.7 \times 10^{-9} < \frac{|s - s_m|}{|s|} < 10^{-8}$, por lo que la suma parcial $\sum_{n=1}^{10^8} \frac{1}{n^2}$ estima el valor de $\sum_{n=1}^{\infty} \frac{1}{n^2}$ con 8 (sin llegar a 9) cifras decimales significativas correctas. El cuadro 2 muestra los resultados de calcular dicha suma parcial en precisión sencilla. Si se usa la instrucción **mean**, el resultado se multiplica por $m = 10^8$. En este caso, la suma exacta es $\pi^2/6$ =1.64493406684822643...

En este caso los resultados más precisos se obtienen con **suma por parejas** y con la instrucción **mean** de Octave. El resultado menos preciso es con **suma acumulada** y con la instrucción **sum** de Octave.

Ejemplo 3. Se generan m=100 millones de números aleatorios (o seudoaleatorios) a_1, a_2,...,a_m con distribución uniforme en $[0, 2/3]$. En teoría, el promedio de dichos números debe ser cercano a 1/3, que es el

Método de cálculo	Valor calculado	No. de cifras significativas correctas
suma acumulada	**1.644**7253	4
Octave, mean	**1.644934**2	7
Octave, sum	**1.644**7253	4
MATLAB, mean	**1.644939**1	6
MATLAB, sum	**1.644939**1	6
suma por parejas	**1.644934**0	8

Cuadro 2

punto medio del intervalo $[0, 2/3]$. Con $m = 10^8$ números, se espera que el promedio aproxime a $1/3$ con aproximadamente 4 cifras decimales (véase [1]). La sucesión de números generados no es la misma en los distintos lenguajes de programación. El cuadro 3 muestra los promedios calculados al usar precisión sencilla.

Método de cálculo	Promedio calculado	Está en el rango esperado
suma acumulada	0.16777216	No
Octave, sum	0.16777216	No
Octave, mean	**0.3333**2801	Sí
MATLAB, mean	**0.3333**0747	Sí
MATLAB, sum	**0.3333**0747	Sí
suma por parejas	**0.3333**0745	Sí

Cuadro 3

4.2 Ejemplos en precisión doble

En los siguientes ejemplos usaremos precisión doble para aproximar promedios o sumas con muchos términos. En el mejor de los casos podremos aproximar los resultados exactos con 16 cifras significativas correctas. Usaremos los 6 métodos de la sección anterior junto con las instrucciones **numpy.mean** y **numpy.sum** de la biblioteca numpy de Python, además de la instrucción no optimizada **sum** de Python (estas instrucciones no están disponibles con precisión sencilla en Python. Debido a esto no se incluyó Python en la sección anterior).

Ejemplo 1. Calcularemos el promedio de $\dfrac{a_1 + a_2 + \cdots + a_m}{m}$, $m = 10^9$, con $a_j = 1/3$ para toda j. Usaremos MATLAB, Octave, y Python en precisión doble. Los resultados se muestran en el cuadro 4.

Método de cálculo	Valor calculado	No. de cifras significativas correctas
suma acumulada	0.**3333333**26651181	8
Octave, sum	0.**3333333**26651181	8
Octave, mean	0.**3333333**26651181	8
MATLAB, mean	0.**333333333335**002	12
MATLAB, sum	0.**333333333335**002	12
Python, numpy.mean	0.**333333333334**148	12
Python, numpy.sum	0.**333333333334**148	12
Python, sum	0.**3333333**26651181	8
suma por parejas	0.**3333333333333333**	16

Cuadro 4

Del cuadro se observa que Python con la instrucción no optimizada **sum**, y Octave con las instrucciones **sum** y **mean**, dan el mismo resultado que **suma acumulada**, que en este ejemplo produce 8 dígitos de precisión. Dicha precisión es la más baja de los resultados mostrados.

Ejemplo 2. Calcular el valor numérico de $s = \sum_{n=1}^{\infty} \frac{1}{n^3}$ con 15 o 16 cifras decimales significativas correctas, usando una suma parcial con precisión doble.

En este caso, de forma análoga al ejemplo 2 de la sección anterior, si $s_m = \sum_{n=1}^{m} \frac{1}{n^3}$, y tomando $f(x) = 1/x^3$, $x > 0$, se obtiene la desigualdad:

$$\frac{1}{2(m+1)^2} < \sum_{n=m+1}^{\infty} \frac{1}{n^3} < \frac{1}{2(m+1)^2} + \frac{1}{(m+1)^3},$$

de donde se concluye que $s = 1 + \sum_{n=2}^{\infty} \frac{1}{n^3} < 1 + \frac{1}{2(2)^2} + \frac{1}{2^3} = \frac{5}{4}$. Por lo tanto, $1 < s < 5/4$, con lo cual se obtiene $\frac{2}{5(m+1)^2} < \frac{|s - s_m|}{|s|} < \frac{1}{2(m+1)^2} + \frac{1}{(m+1)^3}$. Para estimar un entero positivo mínimo m tal que $\frac{1}{2(m+1)^2} + \frac{1}{(m+1)^3} \approx 10^{-16}$, usemos el hecho de que para m grande, $1/(m+1)^3$ es muy pequeño comparado con $1/(2(m+1)^2)$, así que tomamos $m{+}1 \approx \lceil 1/\sqrt{2 \times 10^{-16}} \rceil < 71{\times}10^6$. Se puede verificar que tomando $m{=}71$ millones, se garantiza que la suma parcial s_m aproxima a s con 16 cifras significativas.

En este caso no hay una expresión analítica para el valor de la serie $s = \sum_{n=1}^{\infty} \frac{1}{n^3}$. Dicha serie es el valor $\zeta(3)$ de la función zeta de Riemann, la cual se puede evaluar con software especializado. Por ejemplo, con WolframAlpha se obtiene: $\zeta(3) = 1.20205690315959428539973 81\ldots$ En el cuadro 5 se muestran los resultados del cálculo de la suma parcial con 71 millones de términos.

Método de cálculo	Valor calculado	No. de cifras significativas correctas
suma acumulada	**1.202056903150321**	12
Octave, mean	**1.202056903150321**	12
Octave, sum	**1.202056903150321**	12
MATLAB, mean	**1.20205690315955**9	14
MATLAB, sum	**1.20205690315955**9	14
Python, numpy.mean	**1.20205690315957**2	14
Python, numpy.sum	**1.20205690315957**2	14
Python, sum	**1.202056903150321**	12
suma por parejas	**1.20205690315959**4	16

Cuadro 5

Ejemplo 3. Mismo ejemplo que el ejemplo 3 de la sección anterior, pero en precisión doble. De nuevo, el promedio de dichos números debe ser cercano a $1/3$ en aproximadamente 4 cifras decimales, ya que son 10^8 números pseudoaleatorios con distribución uniforme en $[0, 2/3]$. El cuadro 6 muestra los resultados del cálculo del promedio en precisión doble.

5. Conclusiones

El cálculo aritmético con números redondeados hace que se pierda precisión en ciertos casos. En el cálculo de promedios o sumas con muchos términos positivos, se puede producir pérdida de precisión si el algoritmo que se usa es el de *suma acumulada*. Para el caso en que se usa precisión sencilla, que es aproximadamente redondeo a 8 cifras decimales, la pérdida de precisión puede ser total cuando el número de sumandos es cercano a 10^8. Si se usa el algoritmo de *suma por parejas*, la pérdida de precisión es muy baja aunque se procese una cantidad enorme de sumandos. En precisión sencilla, el algoritmo de *suma por parejas* produce

Método de cálculo	Valor calculado	Está en el rango esperado
suma acumulada	0.**3333**232569110475	Sí
Octave, mean	0.**3333**311524303167	Sí
Octave, sum	0.**3333**311524303167	Sí
MATLAB, mean	0.**3333**240168830786	Sí
MATLAB, sum	0.**3333**240168830786	Sí
Python, numpy.mean	0.**33332**491352753746	Sí
Python, numpy.sum	0.**33332**491352753746	Sí
Python, sum	0.**33332**491352734794	Sí
suma por parejas	0.**3333**477967616926	Sí

Cuadro 6

mayor precisión que los algoritmos programados con las instrucciones *mean* y *sum* en Octave y MATLAB. En la versión analizada de Octave, la instrucción *sum* produce la mayor pérdida de precisión ya que sus resultados son idénticos al los del algoritmo de *suma acumulada*. Por otra parte, los lenguajes aquí analizados son actualizados constantemente, por ejemplo, en Octave se actualizó la instrucción *mean* en precisión sencilla (pero no en precisión doble), con la cual Octave ahora produce alta precisión en el cálculo de promedios o sumas de una gran cantidad de números positivos en precisión sencilla.

Para el caso de precisión doble, que es aproximadamente redondeo a 16 cifras decimales, en el cálculo de promedios y sumas de números positivos usando 10^8 sumandos, se pueden perder hasta 8 cifras de precisión con los siguientes métodos: *suma acumulada*, las instrucciones *mean* y *sum* de Octave, y la instrucción no optimizada *sum* de Python. En este caso, aún quedan al menos $16 - 8 = 8$ cifras de precisión en los cálculos. Para que en precisión doble el algoritmo de *suma acumulada* pueda perder 16 cifras de precisión, se requiere calcular promedios de 10^{16} sumandos positivos. En este caso, el algoritmo de *suma por parejas* perderá a lo más 2 dígito de precisión. Los cálculos con las instrucciones *numpy.sum* y *numpy.mean* de Python producen resultados con la misma precisión que las instrucciones *mean* y *sum* de MATLAB, aunque pierden un poco de precisión con respecto a *suma por parejas*, para el caso de sumas de números positivos.

Si el lector o lectora desea programar el algoritmo de *suma por parejas* en MATLAB, Octave, o Python, se debe tener cuidado en la forma de programarlo. Para obtener una mayor rapidez de ejecución, se debe usar una versión vectorizada. Con ChatGPT se puede obtener un código de este tipo (usar el nombre *pairwise summation* en lugar de *suma por parejas* y pedir una versión vectorizada).

En resumen, el cálculo de sumas de una gran cantidad de datos numéricos requiere tener cuidado con el tipo de variables en las que se almacenan los números y del algoritmo con el que se hacen los cálculos. Problemas de este tipo aparecen en el cálculo de promedios, normas, sumas parciales de una serie, un producto punto, etcétera.

Agradecimientos

Agradezco el tiempo y la paciencia de quienes revisaron este trabajo; sus comentarios fueron de gran ayuda para mejorar este documento.

Bibliografía

[1] R. E. Caflisch, «Monte Carlo and quasi-Monte Carlo methods», *Acta Numerica*, 1998, 1–49.
[2] I. Gladwell, J. G. Nagy y W. E. Ferguson Jr., *Introduction to scientific computing*, Pearson Education, 2011.
[3] R. W. Hockney y C. R. Jesshope, *Parallel computers 2*, CRC Press, 2019.
[4] J. Stoer y R. Bulirsch, *Introduction to numerical analysis*, Springer-Verlag, New York, 1993.