Análisis Numérico I, Primavera 2026

Lección I. Sistema Numérico de Punto Flotante

Imelda Trejo

Última actualización: 1 de febrero de 2026

Índice

1. Introducción

Los **métodos numéricos** son algoritmos computacionales diseñados para construir aproximaciones numéricas de cantidades de interés, especialmente cuando las soluciones exactas son difíciles o imposibles de obtener de manera analítica. Ejemplos típicos incluyen la evaluación de funciones trascendentales, la resolución de sistemas de ecuaciones no lineales, la aproximación de integrales definidas y la simulación de sistemas de ecuaciones diferenciales.

Por ejemplo, funciones como $\sin(x)$, con $x \in \mathbb{R}$, no se calculan de forma exacta en una computadora. En su lugar, se emplean algoritmos que producen aproximaciones numéricas mediante procesos finitos, tales como series truncadas.

El **análisis numérico** proporciona el marco teórico para estudiar estos métodos. Su objetivo no es únicamente obtener aproximaciones, sino evaluar su comportamiento y confiabilidad. En particular, busca responder preguntas como:

- ¿El método converge a la solución correcta?
- ¿Con qué rapidez lo hace?
- ¿Cómo influyen los errores numéricos en el resultado final?

1.1. Fuentes de error en el cálculo numérico

Los errores de aproximación numérica surgen por varias limitaciones inevitables de la computadora, así como por errores de medición y en las operaciones aritméticas:

- **Memoria finita:** solo se dispone de un número fijo de bits (0 y 1) para representar cada número.
- **Longitud finita de palabra:** las operaciones se efectúan con registros de tamaño fijo (por ejemplo, 32 o 64 bits).
- **Redondeo y truncamiento:** muchos números reales no pueden representarse exactamente en el sistema de punto flotante.

Estas limitaciones hacen que la aritmética computacional difiera de la aritmética exacta de los números reales $\mathbb{R}$, introduciendo errores que deben ser analizados y controlados cuidadosamente.

2. Sistema de punto flotante

2.1. Representación general de un número real

Un número real en el lenguaje de máquina se expresa como

$$x = \text{signo} \times \text{mantisa} \times \text{base}^{\text{exponente}}.$$

La estructura típica de un número en punto flotante consta de tres campos:

Campo	Descripción
1 bit	Signo ($\pm$)
Varios bits	Exponente
Varios bits	Mantisa

Normalmente, la base es $\beta = 10$ (notación científica) o $\beta = 2$ (notación binaria).

2.2. Notación científica (base 10)

En notación científica normalizada, un número se expresa como

$$732.5051 = 0.7325051 \times 10^3, \qquad -0.005612 = -0.5612 \times 10^{-2}.$$

En general,

$$x = \pm r \times 10^n, \qquad \frac{1}{10} \leq r < 1,$$

donde r es la mantisa normalizada y $n \in \mathbb{Z}$ es el exponente.

2.3. Notación binaria (base 2)

De manera análoga, en base 2:

$$x = \pm q \times 2^m, \qquad \frac{1}{2} \leq q < 1,$$

donde q es la mantisa normalizada en binario y $m \in \mathbb{Z}$ es el exponente.

2.4. Definición formal del sistema de punto flotante

Definición 2.1. Un **sistema de punto flotante** se define como

$$F(\beta, t, L, U),$$

donde:

- $\beta \geq 2$ es la base (usualmente $\beta = 2$),
- $t \geq 1$ es el número de dígitos de la mantisa,
- L y U son los límites inferior y superior del exponente, con $L < 0 < U$.

Un número $x \in F(\beta, t, L, U)$, $x \neq 0$, tiene la forma normalizada

$$x = \pm(0.c_1 c_2 \cdots c_t)_\beta \times \beta^n,$$

donde $0 \leq c_i \leq \beta - 1$ para $i = 1, \ldots, t$, con $c_1 \neq 0$ (condición de normalización), y el exponente satisface $L \leq n \leq U$.

Observación 2.1. El cero se representa convencionalmente con todos los dígitos de la mantisa y el exponente iguales a cero, es decir, $x = 0$ no satisface la condición de normalización $c_1 \neq 0$.

2.5. Función de punto flotante

Definición 2.2. Dado un número real $x \in \mathbb{R}$, denotamos por

$$\mathrm{fl}(x)$$

su representación en punto flotante, es decir, el elemento de $F(\beta, t, L, U)$ que aproxima a x.

La función $\mathrm{fl}(x)$ se obtiene truncando o redondeando la expansión de x a t dígitos en la mantisa:

$$\mathrm{fl}(x) = \pm(0.c_1 c_2 \cdots c_t)_\beta \times \beta^n.$$

Existen dos mecanismos comunes para definir $\mathrm{fl}(x)$:

- **Truncamiento (chopping):** se descartan los dígitos posteriores al t-ésimo.
- **Redondeo (rounding):** se aproxima al número representable más cercano. En caso de empate (cuando x está exactamente a mitad de camino entre dos números representables), se suele elegir el que tiene el último dígito par (redondeo al par más cercano o *banker's rounding*).

Ejemplo 2.1. El número real

$$\frac{1}{3} = 0.333333\ldots$$

no puede representarse exactamente en base 10 con finitos dígitos. Usando cuatro cifras decimales:

- El truncamiento produce
$$\mathrm{fl}_{\mathrm{trunc}}\left(\frac{1}{3}\right) = 0.3333 \times 10^0.$$
- El redondeo produce
$$\mathrm{fl}_{\mathrm{round}}\left(\frac{1}{3}\right) = 0.3333 \times 10^0.$$

En este caso ambos métodos coinciden. Sin embargo, para $\frac{2}{3} = 0.666666\ldots$:

- Truncamiento: $\mathrm{fl}_{\mathrm{trunc}}(2/3) = 0.6666 \times 10^0$.
- Redondeo: $\mathrm{fl}_{\mathrm{round}}(2/3) = 0.6667 \times 10^0$.

En general, el redondeo proporciona menor error que el truncamiento.

Casos especiales. Números muy pequeños o muy grandes pueden quedar fuera del rango permitido por el exponente:

- **Subdesbordamiento (*underflow*):** ocurre cuando $0 < |x| < \beta^L$. En este caso, x se redondea típicamente a cero, aunque algunos sistemas implementan números subnormales para representar valores en este rango.
- **Desbordamiento (*overflow*):** ocurre cuando $|x| > \beta^U(1-\beta^{-t})$, es decir, cuando $|x|$ excede el máximo número representable. Esto produce un error y la computación se interrumpe o se asigna un valor especial ($\pm\infty$ o NaN, *Not a Number*).

2.6. Operaciones aritméticas en punto flotante

Dadas las operaciones aritméticas básicas $\circ \in \{+, -, \times, \div\}$ en $\mathbb{R}$, las operaciones en punto flotante se definen mediante:

$$\mathrm{fl}(x) \circ_{\mathrm{fl}} \mathrm{fl}(y) := \mathrm{fl}(\mathrm{fl}(x) \circ \mathrm{fl}(y)),$$

donde $x, y \in \mathbb{R}$ y $\mathrm{fl}(x), \mathrm{fl}(y) \in F(\beta, t, L, U)$.

Más brevemente, para $x, y \in F$:

$$x \circ_{\mathrm{fl}} y = \mathrm{fl}(x \circ y).$$

Es decir, la operación en punto flotante consiste en:

1. Realizar la operación exacta $x \circ y$ en $\mathbb{R}$.
2. Redondear el resultado mediante $\mathrm{fl}(\cdot)$ para obtener un elemento de $F(\beta, t, L, U)$.

Ejemplo 2.2. Consideremos $F(10, 4, -10, 10)$ con redondeo. Sean:

$$x = 0.1234 \times 10^1 = 1.234, \qquad y = 0.5678 \times 10^0 = 0.5678.$$

Entonces:

$$x + y = 1.234 + 0.5678 = 1.8018 = 0.18018 \times 10^1.$$

Como 0.18018 tiene 5 dígitos significativos, se redondea a 4:

$$x +_{\text{fl}} y = \text{fl}(x + y) = \text{fl}(1.8018) = 0.1802 \times 10^1 = 1.802.$$

2.6.1. $F(\beta, t, L, U)$ no es un campo

Del análisis anterior, se deduce inmediatamente que $F(\beta, t, L, U)$ con las operaciones $+_{\text{fl}}, \times_{\text{fl}}$ **no forma un campo**. Las razones son:

Observación 2.2 (Falta de cerradura)**.** Si $x, y \in F$, en general $x + y \notin F$ (el resultado exacto puede requerir más de t dígitos). Solo $\text{fl}(x + y) \in F$.

Por ejemplo, en $F(10, 2, -10, 10)$:

$$x = 0.99 \times 10^0, \quad y = 0.99 \times 10^0 \quad \Rightarrow \quad x + y = 1.98 = 0.198 \times 10^1 \notin F,$$

por que requiere 3 dígitos significativos, pero $\text{fl}(x + y) = 0.20 \times 10^1 \in F$.

Observación 2.3 (Falta de asociatividad)**.** En general, $\text{fl}(\text{fl}(x + y) + z) \neq \text{fl}(x + \text{fl}(y + z))$.

Por ejemplo, en $F(10, 3, -10, 10)$ con redondeo, sean:

$$x = 0.123 \times 10^4 = 1230, \quad y = 0.456 \times 10^0 = 0.456, \quad z = -0.123 \times 10^4 = -1230.$$

Entonces:

$$\begin{aligned}
\text{fl}(\text{fl}(x + y) + z) &= \text{fl}(\text{fl}(1230.456) + (-1230)) \\
&= \text{fl}(0.123 \times 10^4 + (-0.123 \times 10^4)) \\
&= \text{fl}(0) = 0, \\
\text{fl}(x + \text{fl}(y + z)) &= \text{fl}(1230 + \text{fl}(-1229.544)) \\
&= \text{fl}(1230 + (-0.123 \times 10^4)) \\
&= \text{fl}(0.456) = 0.456 \times 10^0.
\end{aligned}$$

Por lo tanto, $(x +_{\text{fl}} y) +_{\text{fl}} z \neq x +_{\text{fl}} (y +_{\text{fl}} z)$.

Observación 2.4 (Falta de distributividad)**.** En general,

$$\text{fl}(x \cdot \text{fl}(y + z)) \neq \text{fl}(\text{fl}(x \cdot y) + \text{fl}(x \cdot z)).$$

Es decir, $x \times_{\text{fl}} (y +_{\text{fl}} z) \neq (x \times_{\text{fl}} y) +_{\text{fl}} (x \times_{\text{fl}} z)$.

Observación 2.5 (Cancelación catastrófica)**.** La sustracción de números casi iguales puede resultar en pérdida significativa de dígitos significativos. Si $x \approx y$, entonces $\text{fl}(x - y)$ puede tener muy pocos dígitos correctos, amplificando errores relativos.

Por ejemplo, en $F(10, 4, -10, 10)$:

$$x = 0.1234 \times 10^1, \quad y = 0.1233 \times 10^1.$$

Entonces:

$$x - y = 0.0001 \times 10^1 = 0.1000 \times 10^{-2},$$

pero el resultado tiene solo 1 dígito significativo correcto (el 1), mientras que x y y tenían 4 dígitos cada uno.

Conclusión: El conjunto $F(\beta, t, L, U)$ es simplemente un subconjunto finito de $\mathbb{R}$, y la aritmética de punto flotante no satisface los axiomas de campo. Esta pérdida de propiedades algebraicas es una fuente fundamental de errores en el cómputo numérico y motiva el estudio cuidadoso de la estabilidad numérica de los algoritmos.

3. Precisión del sistema de punto flotante

3.1. Relación entre bits y cifras decimales

La cantidad de cifras decimales significativas que puede representar un número en punto flotante depende del número de bits asignados a la mantisa. En particular, si la mantisa tiene t bits, entonces el número máximo de valores distinguibles en la mantisa es exactamente 2^t. Esto corresponde aproximadamente a

$$\log_{10}(2^t) = t \log_{10}(2) \approx 0.301\, t$$

cifras decimales significativas.

Por esta razón:

- La **precisión simple** (IEEE 754, 32 bits: 1 bit de signo, 8 bits de exponente, 23 bits de mantisa más 1 bit implícito) permite alrededor de $0.301 \times 24 \approx 7$ cifras decimales significativas.
- La **precisión doble** (IEEE 754, 64 bits: 1 bit de signo, 11 bits de exponente, 52 bits de mantisa más 1 bit implícito) permite alrededor de $0.301 \times 53 \approx 16$ cifras decimales significativas.

Observación 3.1. En el estándar IEEE 754, el bit más significativo de la mantisa normalizada siempre es 1 (excepto para números subnormales y cero), por lo que no se almacena explícitamente. Esto proporciona un bit adicional de precisión efectiva.

3.2. Consideraciones prácticas sobre la precisión

En aplicaciones científicas tradicionales se emplea comúnmente aritmética de 64 bits (precisión doble), que permite controlar adecuadamente los errores numéricos en la mayoría de los problemas. Sin embargo, en aplicaciones modernas que involucran grandes volúmenes de datos y cálculos masivos —como simulaciones a gran escala, procesamiento de imágenes, visión computacional o aprendizaje automático (*machine learning*)— se utilizan con frecuencia representaciones de menor precisión (por ejemplo, 32 o incluso 16 bits, como `float16` o `bfloat16`) para reducir costos computacionales y de memoria.

Aceptar menor precisión implica tolerar errores numéricos más grandes, pero en muchos casos estos errores no afectan de manera significativa el resultado final, especialmente cuando se promedian sobre grandes conjuntos de datos o cuando el problema es inherentemente ruidoso. Esto resalta una idea central del análisis numérico:

La precisión necesaria depende del problema que se desea resolver.

4. Error absoluto y error relativo

Sea x un valor exacto (o verdadero) y sea x^* una aproximación de x.

Definición 4.1 (Error absoluto)**.** El **error absoluto** de la aproximación x^* respecto a x se define como

$$E_{\text{abs}} = |x - x^*|.$$

Definición 4.2 (Error relativo)**.** El **error relativo** de la aproximación x^* respecto a x se define como

$$E_{\text{rel}} = \frac{|x - x^*|}{|x|}, \quad \text{para } x \neq 0.$$

Observación 4.1. El error relativo es una medida adimensional que es independiente de la magnitud de x, lo cual permite comparar la calidad de aproximaciones de valores en diferentes escalas. Por ejemplo, un error absoluto de 10^{-6} es significativo si $x \approx 10^{-5}$ pero negligible si $x \approx 10^3$.

Observación 4.2 (Diferentes fuentes de aproximación)**.** El valor aproximado x^* puede provenir de diversas fuentes:

1. **Representación en punto flotante:** $x^* = \text{fl}(x)$. Por ejemplo, $x = \pi$ y $x^* = 3.14159265$.
2. **Medición experimental:** x es el valor real de una cantidad física y x^* es el valor medido con un instrumento.
3. **Métodos iterativos:** x es la solución exacta de una ecuación y x^* es la aproximación obtenida después de n iteraciones.
4. **Truncamiento de series:** $x = \sum_{k=0}^{\infty} a_k$ y $x^* = \sum_{k=0}^{n} a_k$, donde $n \in \mathbb{N}$ es finito.

En todos estos casos, el análisis de error mediante E_{abs} y E_{rel} permite cuantificar la calidad de la aproximación.

5. Épsilon de máquina

Definición 5.1 (Épsilon de máquina)**.** El **épsilon de máquina** (ε_{maq} o *machine epsilon*) es la cota superior del error relativo de redondeo para todos los números representables en $F(\beta, t, L, U)$.

Para un sistema con redondeo:

$$\varepsilon_{\text{maq}} = \frac{1}{2}\beta^{1-t}.$$

Para un sistema con truncamiento:

$$\varepsilon_{\text{maq}} = \beta^{1-t}.$$

Esto garantiza que para todo $x \in \mathbb{R}$, $x \neq 0$, que puede ser representado sin *overflow* ni *underflow*, se cumple:

$$\frac{|x - \text{fl}(x)|}{|x|} \leq \varepsilon_{\text{maq}}.$$

Equivalentemente, podemos escribir

$$\text{fl}(x) = x(1 + \delta), \quad \text{donde } |\delta| \leq \varepsilon_{\text{maq}}.$$

Ejemplo 5.1 (Valores para IEEE 754)**.** Para el estándar IEEE 754 con redondeo:

- Precisión simple ($t = 24$ bits efectivos):

$$\varepsilon_{\text{maq}} = \frac{1}{2} \cdot 2^{1-24} = 2^{-24} \approx 5.96 \times 10^{-8}.$$

- Precisión doble ($t = 53$ bits efectivos):

$$\varepsilon_{\text{maq}} = \frac{1}{2} \cdot 2^{1-53} = 2^{-53} \approx 1.11 \times 10^{-16}.$$

Observación 5.1. El épsilon de máquina también puede definirse alternativamente como el número positivo más pequeño ε tal que $\text{fl}(1 + \varepsilon) > 1$. Ambas definiciones están relacionadas pero pueden diferir por un factor de 2 dependiendo del contexto.

6. Observaciones finales

El sistema de punto flotante es la base de toda la aritmética computacional moderna. Comprender sus limitaciones y propiedades es esencial para:

- Diseñar algoritmos numéricamente estables.
- Analizar la propagación de errores en cálculos complejos.
- Interpretar correctamente los resultados numéricos.

En las próximas lecciones se estudiarán temas como el análisis de errores de redondeo en operaciones aritméticas básicas, la estabilidad numérica de algoritmos, y métodos para controlar y minimizar los errores numéricos en problemas prácticos.

Lecturas recomendadas

Bibliografía básica. Para el estudio introductorio del análisis numérico, la aritmética de punto flotante y el análisis del error, se recomiendan los textos clásicos de Kincaid y Cheney [2] y de Burden y Faires (Capítulo 1, pag. 20-22) `https://www.academia.edu/30466412/An%C3%A1lisis_Num%C3%A9rico_J_Douglas_Faires_y_Richard_L_Burden_M%C3%A9todos_Num%C3%A9ricos_` [1].

Para una rápida introducción al mundo moderno AI Precisión reducida y aprendizaje automático [3]

Referencias

[1] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Brooks/Cole, 7 edition, 2001.

[2] David Kincaid and Ward Cheney. *Numerical Analysis of Scientific Computing*. Brooks/Cole Publishing Company, 1991.

[3] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. `https://arxiv.org/abs/1710.03740`, 2017. Published as a conference paper at ICLR 2018.