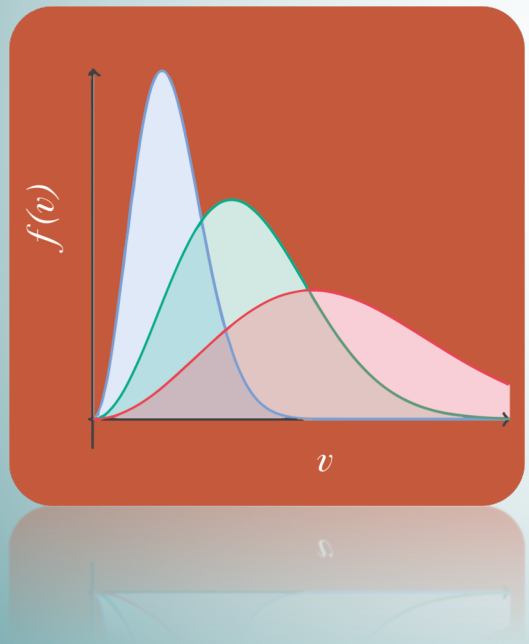
Métodos Matemáticos en la Ingeniería

Bloque II. Métodos Numéricos

Tema 5. Cuestiones Básicas sobre Aritmética Computacional



Vera Egorova

Sara Pérez Carabaza

Departamento de Matemática Aplicada
y Ciencias de la Computación

Contenido

¿Qué es un Método Numérico?

Método Numérico

Un método numérico es un procedimiento mediante el cual se obtiene, casi siempre de manera *aproximada*, la solución de ciertos problemas realizando cálculos puramente aritméticos y lógicos.

Consiste de una lista finita de instrucciones precisas que especifican una secuencia de operaciones algebraicas y lógicas (**algoritmo**), que producen una aproximación de la solución del problema (**solución numérica**).

Los métodos numéricos son esenciales para:

- Resolver problemas que no tienen solución analítica.
- Manejar problemas de gran tamaño o complejidad.
- Obtener soluciones rápidamente mediante el uso de ordenadores.
- Permitir la simulación y modelización de sistemas reales.

- **Aproximación**: Los métodos numéricos proporcionan soluciones aproximadas, no exactas.
- **Iteración**: Muchos métodos dependen de procesos iterativos para refinar la solución.
- **Error**: Siempre hay un error asociado con las soluciones, que se puede cuantificar y minimizar.
- **Computación**: Utilizan algoritmos que pueden ser implementados en computadoras para resolver problemas grandes y complejos.

La eficiencia en el cálculo depende de:

- La facilidad de implementación del algoritmo.
- Las características especiales y limitaciones de los instrumentos de cálculo.

No os fiáis del ordenador

1. Ejecutar sobre el ordenador: $0{,}3 - 0{,}2 - 0{,}1$
2. Ejecutar sobre el ordenador: $10^{25} + 345 - 10^{25} + 54 + 10^{50} - 2 - 10^{50}$

Importante

Cualquier procedimiento numérico debe considerar el control de errores para medir cómo afectan al resultado!

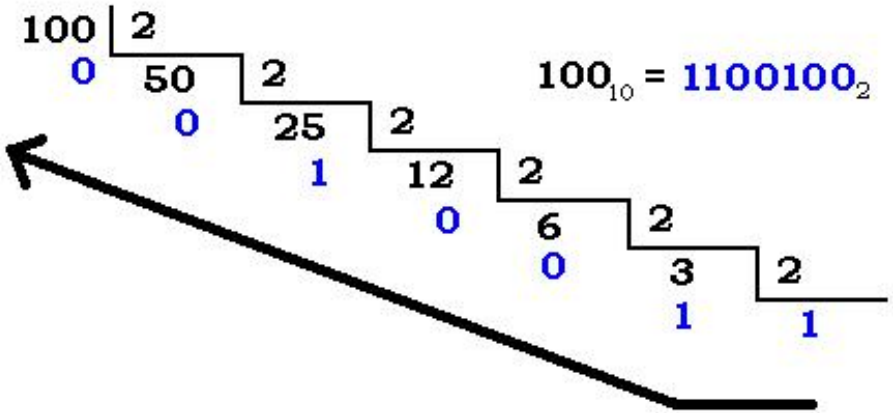
El ordenador, normalmente, recibe información en formato decimal, la cual es transformada a binario. Posteriormente, realiza las operaciones pertinentes, convierte el resultado de nuevo a decimal y finalmente informa al usuario de este resultado.

Sistema binario

Sistema binario es un sistema de numeración en el que los números se representan utilizando solamente dos cifras: cero y uno (0 y 1).

Para transformar un número del sistema decimal al sistema binario:

1. Se transforma la parte entera a binario:



2. Se sigue con la parte fraccionaria, multiplicando cada número por 2. Si el resultado obtenido es mayor o igual a 1 se anota como un uno (1) binario. Si es menor que 1 se anota como un 0 binario.
3. Después de realizar cada multiplicación, se colocan los números obtenidos en el orden de su obtención. Algunos números se transforman en dígitos periódicos, por ejemplo: el 0.1.

Ejercicio (1.1)

Sea el número decimal 10.5708, pasarse a binario hasta 8 decimales.

1. La parte entera: $10 = (1010)_2$
2. La parte fraccionaria:
   $0{,}5708 \times 2 = 1{,}1416 \rightarrow 1$
   $0{,}1416 \times 2 = 0{,}2832 \rightarrow 0$
   $0{,}2832 \times 2 = 0{,}5664 \rightarrow 0$
   $0{,}5664 \times 2 = 1{,}1328 \rightarrow 1$
   $0{,}1328 \times 2 = 0{,}2656 \rightarrow 0$
   $0{,}2656 \times 2 = 0{,}5312 \rightarrow 0$
   $0{,}5312 \times 2 = 1{,}0624 \rightarrow 1$
   $0{,}0624 \times 2 = 0{,}1248 \rightarrow 0$
3. $0{,}5708 = (0{,}10010010)_2$
4. $10{,}5708 = (1010{,}10010010)_2$

Los cálculos de ordenador utilizan 2 tipos de números:

- **Integer** (Números enteros)
  Representados por un número fijo de bits, estos números no tienen parte fraccionaria.
  Con 64 bits se pueden representar 2^{64} valores:
  - **Enteros sin signo:** Desde 0 hasta $2^{64} - 1$.
  - **Enteros con signo:** Desde -2^{63} hasta $2^{63} - 1$.

- **Números de punto flotante** (Números reales)
  Representados de la forma $(-1)^s \times m \times \beta^e$ donde
  - $s-$ Signo (0 para positivo, 1 para negativo);
  - $m-$ Mantisa, se representa como $0.a_1a_2a_3...$ en base β , donde los dígitos a_i cumplen $0 \leq a_i \leq \beta - 1$.
  - $\beta -$ Base del sistema de numeración (generalmente 2);
  - $e-$ Exponente, un entero que indica la potencia a la que se eleva la base.

- Un sistema de números de punto flotante es **discreto y finito**.
- Los números de punto flotante pueden representar un amplio rango de valores, desde números muy pequeños (cercanos a cero) hasta números muy grandes.
- La precisión de la representación depende de la longitud de la mantisa y del rango del exponente.

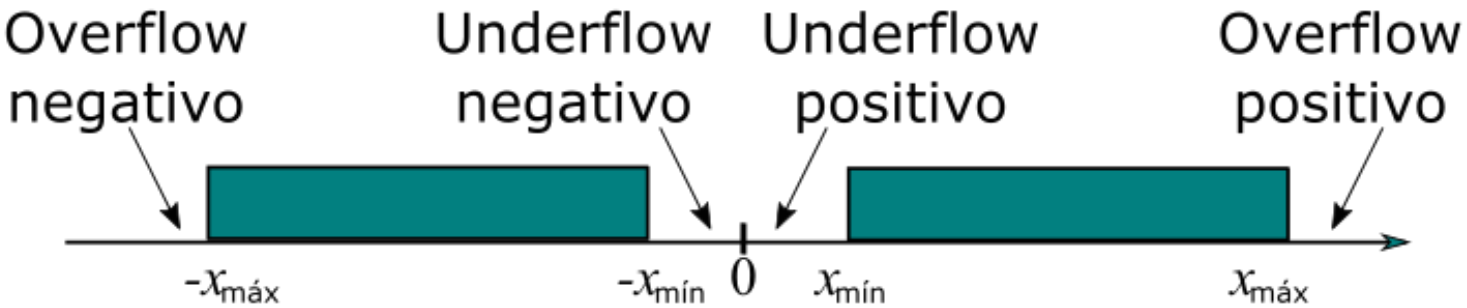
Ventajas	Desventajas
+ **Flexibilidad:** Permite representar tanto números muy grandes como muy pequeños.	- **Precisión limitada:** Debido a la longitud finita de la mantisa y del exponente, la precisión es limitada.
+ **Eficiencia:** Es una forma compacta y eficiente de almacenar números reales en la memoria del ordenador.	- **Errores de redondeo:** Las operaciones aritméticas con números de punto flotante pueden introducir pequeños errores de redondeo

Dado un sistema binario ($\beta = 2$) con una mantisa de 4 cifras. Representa los números 0.125, 0.5, 1, 1.125, 2.25.

1. Definir el bit de signo: 0, si es positivo.
2. Representar el número en el sistema binario (obtener mantisa)
3. Normalizar: Ajustar el número binario de modo que la mantisa esté en la forma más simple posible (por lo general, se mueve el punto binario a la izquierda o a la derecha para que haya un solo 1 a la izquierda del punto binario).
4. Calcular el exponente: Contar cuántas posiciones se ha movido el punto binario para determinar el valor del exponente.

Por ejemplo, para el número $x = 0{,}125$:

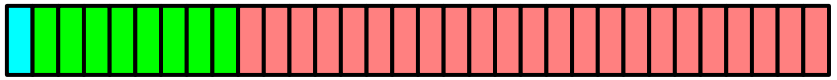
1. Bit de signo: 0 (es positivo).
2. Convertimos el número a binario: $0{,}125_{10} = 0{,}0010_2$.
3. Normalizamos: $0{,}0010 \rightarrow 0{,}1000$
4. $e = -2$, (se mueve el punto binario 2 posiciones a la izquierda).



En la recta de los números reales hay cinco regiones excluidas para los números de punto flotante:

1. Los números negativos $x < -x_{máx} \Rightarrow$ **desbordamiento (overflow) negativo**.
2. Los números negativos $x > -x_{mín} \Rightarrow$ **desbordamiento a cero (underflow) negativo**.
3. El cero.
4. Los números positivos $x < x_{mín} \Rightarrow$ **desbordamiento a cero (underflow) positivo**.
5. Los números positivos $x > x_{máx} \Rightarrow$ **desbordamiento (overflow) positivo**.

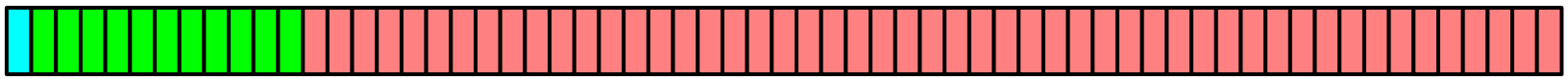
precisión simple



1 bit 8 bits 23 bits

signo exponente mantisa (fracción)

1 bit 11 bits 52 bits



precisión doble

	Precisión simple	Precisión doble
Memoria	$23+8+1=32$ bits	$52+11+1=64$ bits
$x_{\text{mín}}$	$2^{-126}\approx 10^{-38}$	$2^{-1022}\approx 10^{-308}$
$x_{\text{máx}}$	$2^{128}(1-2^{-24})\approx 10^{38}$	$2^{1024}(1-2^{-53})\approx 10^{308}$
Dígitos significativos	7	16

Ejercicio (1.2)

Escribir una función que transforma dado número al número de punto flotante (precisión simple).

- **Números denormalizados**: Son valores más pequeños que el mínimo valor normalizado, pero mayores que cero.
- **Ceros**
- **Infinitos**: Cuando un cálculo produce un desbordamiento (overflow)
- **NaN** (Not A Number) - Representa resultados indefinidos o no representables, como `0/0`, `Inf - Inf`, etc.

Valor	Exponente	Mantisa
Normalizados	$L \leq e \leq U$	$\neq 0$
Denormalizados	ceros $(e = L - 1)$	$\neq 0$
± 0	ceros $(e = L - 1)$	0
$\pm \infty$	unos $(e = U + 1)$	0
NaN	unos $(e = U + 1)$	$\neq 0$

Ejercicio (1.3)

El logaritmo natural corresponde a la función inversa de la función exponencial natural:

$$x = \ln(\exp(x)) = \exp(\ln(x)).$$

Muestra con un ejemplo que en aritmética computacional no es exactamente así.

Epsilon de la máquina

La precisión de la máquina (también conocida como "**epsilon de la máquina**.º eps) es el valor más pequeño que, al sumarse a 1, produce un resultado distinto de 1:

$$\varepsilon_M : \quad 1 + \varepsilon_M > 1$$

Ejercicio (1.4)

Usando la definición del epsilon de la maquina, escribe el código para obtener su valor.

```
epsilon = 1.0;
while (1+epsilon/2>1)
^^Iepsilon = epsilon/2;
end
epsilon
```

Compara el resultado con el valor predefinido de MATLAB:

```
eps
```

Supongamos que $\hat{p}$ es una aproximación a p.

- El **error absoluto** de la aproximación es $E_p = |p - \hat{p}|$
- El **error relativo** es $R_p = \dfrac{|p - \hat{p}|}{p}$ (El error relativo también se puede multiplicar por 100 %.)
- Diremos que un número $\hat{p}$ es una aproximación a p con d **cifras decimales significativas** si d es el mayor número natural tal que

$$\frac{|p - \hat{p}|}{|p|} < \frac{10^{-d}}{2},$$

de donde $d < \lg \dfrac{|p|}{2|p - \hat{p}|}$

Ejercicio

Determina el número de cifras significativas de las siguientes aproximaciones:

1. $x = 3{,}141592, \quad \hat{x} = 3{,}14$
2. $y = 1\,000\,000, \quad \hat{y} = 999\,996$
3. $z = 0{,}000012, \quad \hat{z} = 0{,}000009$

Factores que afectan la precisión numérica:

1. **Errores iniciales** son el resultado de medidas de precisión limitada.
2. **Errores de truncamiento**: que resultan al usar una aproximación en lugar de un procedimiento matemático exacto.
3. **Errores de redondeo** que se producen cuando se usan números que tienen un límite de cifras significativas para representar números exactos.
   - Representación de los números reales tales como π, e, $\sqrt{5}, \ldots$
   - Representación de algunos números reales en sistema binario (por ejemplo, 0,1)
4. **Estabilidad del algoritmo**: Un algoritmo numérico estable es aquel que no amplifica los errores de redondeo durante sus cálculos.
5. **Condicionamiento del problema**: Algunos problemas son inherentemente sensibles a pequeñas perturbaciones en los datos de entrada. Estos problemas se consideran mal condicionados, y los errores de entrada pueden amplificarse significativamente en el resultado.

La política de redondeo implícitamente contemplada en el estándar IEEE754 es el redondeo **al más cercano, desempatando al par**. Esto significa que si un número está exactamente a medio camino entre dos números representables, se elige aquel cuyo último bit de la mantisa es 0 (el "par"). El error relativo máximo cometido al redondeo de un número es la mitad del epsilon de la maquina:

Unidad de redondeo

$$\mu = \varepsilon_M/2$$

Para cualquier número x tal que $|x| < \mu$, la operación $\text{fl}(1 + x)$ se redondeará a 1. Esto se debe a que $1 + x$ está más cerca de 1 que del siguiente número representable, $1 + \varepsilon_M$.

Ejercicio (1.5)

Se realizan las siguientes operaciones:

1. $1 + \mu - 1$
2. $-1 + \mu + 1$
3. $(1 + 2\mu) + \mu - 1$
4. $(1 + 2\mu) + \mu - (1 + 2\mu)$

Ejecútalas y explica los resultados.

1. $1 + \mu - 1 \to 0$
   Paso 1: $\text{fl}(1 + \mu) = 1$.
   La mantisa de 1,0 es 1,000..,0_2, que termina en 0 (es par). La mantisa de $1 + \varepsilon_M$ es 1,000..,1_2, que es impar. Por lo tanto, se redondea hacia abajo.
   Paso 2: $\text{fl}(1 - 1) = 0$
2. $-1 + \mu + 1 \to \mu$
   En los compiladores modernos, el cálculo intermedio $(-1 + \mu)$ se realiza en registros internos de mayor precisión, donde el valor no se redondea prematuramente a -1. Al sumar 1 a continuación, la cancelación ocurre limpiamente dentro del registro, y el resultado final μ se obtiene antes de ser almacenado en la variable de 64 bits.
3. $(1 + 2\mu) + \mu - 1 \to 2\varepsilon_M$
   Paso 1: $\text{fl}(1 + 2\mu) = 1 + \varepsilon_M$
   Paso 2: $\text{fl}(1 + \varepsilon_M + \mu) = 1 + 2\varepsilon_M$ (la regla de redondeo al par)
   Paso 3: $\text{fl}(1 + 2\varepsilon_M - 1) = 2\varepsilon_M$
4. $(1 + 2\mu) + \mu - (1 + 2\mu) \to \varepsilon_M$
   Ya sabemos que $\text{fl}((1 + 2\mu) + \mu) = 1 + 2\varepsilon_M$ y $\text{fl}(1 + 2\mu) = 1 + \varepsilon_M$. Por tanto, $\text{fl}((1 + 2\varepsilon_M) - (1 + \varepsilon_M)) = \varepsilon_M$

Pérdida de cifras significativas

La pérdida de cifras significativas se produce cuando operaciones matemáticas reducen la precisión del resultado final al eliminar dígitos significativos.

Este fenómeno puede ocurrir, por ejemplo, durante la resta de números similares.

Ejemplo

Dados dos números:
$a_1 = 0{,}3333333333333298$ y $a_2 = 0{,}3333333333333187$.
Resta Exacta: resultado $= 1{,}11 \times 10^{-14}$
Resta en MATLAB: resultado $= 1{,}11022302462515 7e-14$

Estrategias para mitigar la pérdida de cifras significativas

- Reformular el problema para evitar operaciones que conduzcan a cancelaciones sustractivas.
- Utilizar representaciones de mayor precisión para reducir los errores de redondeo.

Ejercicios

1.6. Reformula las siguientes expresiones de manera que su evaluación numérica sea estable:

a) $\dfrac{1}{1+2x} - \dfrac{1-x}{1+x}$ para $|x| \ll 1$

b) $\sqrt{x+\dfrac{1}{x}} - \sqrt{x-\dfrac{1}{x}}$ para $|x| \gg 1$

1.7. Dada la función

$$f(x) = \frac{1}{1-x} - \frac{1}{1+x}, \quad |x| \neq 1.$$

¿Para qué valores de x es difícil calcular precisamente la expresión en aritmética de punto flotante? Reformula la expresión de manera que su evaluación numérica sea estable.

1.8. Para calcular el punto medio m de un intervalo $[a, b]$, ¿cuál de las siguientes expresiones es preferible en aritmética de punto flotante? ¿Cuándo? ¿Por qué?

a) $m = \dfrac{a+b}{2{,}0}$

b) $m = a + \dfrac{b-a}{2{,}0}$

Ecuación cuadrática: $ax^2 + bx + c = 0$

- Raices:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

- ¿Qué pasa si $b^2 \gg ac$?
  - $\sqrt{b^2 - 4ac} \approx |b| \quad \rightarrow \quad$ Cancelación en una de las raíces.
- ¿Cómo evitar la cancelación?
- Multiplicar numerador y denominador de la raíz "problemática"por $-b - \sqrt{b^2 - 4ac}$:

$$x_{1,2} = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}} \tag{2}$$

- Si $b > 0$, utilizar (2) para x_1 y (1) – para x_2
- Si $b < 0$, utilizar (1) para x_1 y (2) – para x_2

Ejercicio 1.9

Construir un algoritmo estable que calcule las raíces de una ecuación cuadrática en todas las situaciones posibles, incluyendo los casos problemáticos cuando $\sqrt{b^2 - 4ac} \approx |b|$.

Se dice que una función $f(h)$ es de orden $g(h)$ cuando $h \to 0$, lo que se denota por $f(h) = O(g(h))$ (O mayúscula de Landau), si existen constantes C y c tales que

$$|f(h)| \leq C|g(h)|, \quad |h| \leq c.$$

Orden de aproximación

Supongamos que una función $p(h)$ aproxima a otra $f(h)$ y que existen una constante M y un número natural n tales que

$$\frac{|f(h) - p(h)|}{|h^n|} \leq M \text{ para } h \text{ suficientemente pequeño.}$$

Entonces, se dice que $p(h)$ aproxima a $f(h)$ con orden de aproximación $O(h^n)$:

$$f(h) = p(h) + O(h^n)$$

Denotaremos la versión punto flotante de la máquina de un número x como

$$fl(x) = x(1 + \epsilon),$$

donde ε es un número pequeño dependiente de x.

El error relativo del número de punto flotante:

$$\frac{|x - fl(x)|}{|x|} = \frac{|x - (x + x\epsilon)|}{|x|} = |\epsilon|$$

Se tiene representaciones de punto flotante $fl(x)$ y $fl(y)$ para los números reales x e y.

Operaciones aritméticas

$$\begin{aligned} x \oplus y &= fl(fl(x) + fl(y)) \\ x \ominus y &= fl(fl(x) - fl(y)) \\ x \otimes y &= fl(fl(x) \cdot fl(y)) \\ x \oslash y &= fl(fl(x)/fl(y)) \end{aligned}$$

Consideramos la suma:

$$x \oplus y = fl(fl(x) + fl(y)$$

Según la definición de punto flotante tenemos:

$$x \oplus y = (x(1+\epsilon_x) + y(1+\epsilon_y))(1+\epsilon_{x+y})$$

Elegimos $\epsilon = \text{máx}(|\epsilon_x|, |\epsilon_y|, |\epsilon_{x+y}|) < \mu$:

$$x \oplus y = (x+y)(1+\epsilon_{x+y})^2 = (x+y)(1+2\epsilon+\epsilon^2)$$

Error relativo de la suma:

$$\frac{|(x+y)-(x\oplus y)|}{|x+y|} \leq \frac{|x|+|y|}{|x+y|}(2\epsilon+\epsilon^2) \simeq \frac{|x|+|y|}{|x+y|} \cdot 2\epsilon \leq \frac{|x|+|y|}{|x+y|}\varepsilon_M$$

- **Error de la suma**

$$\frac{|(x+y)-(x\oplus y)|}{|x+y|} \leq \frac{|x|+|y|}{|x+y|}\cdot \varepsilon_M$$

- **Error del producto**

$$\frac{|xy - fl(fl(x)\cdot fl(y))|}{|xy|} \leq 3\epsilon + 3\epsilon^2 + \epsilon^3 \simeq 3\epsilon \leq 2\varepsilon_M$$

- **Error de la división**

$$\frac{|x/y - fl(fl(x)/fl(y))|}{|x/y|} \leq \frac{3\epsilon+\epsilon^2}{1-\epsilon} \simeq 3\epsilon \leq 2\varepsilon_M$$

- **Error (absoluto) en una función**

$$\Delta f(\hat{x}) = |f(x) - f(\hat{x})| \cong |f'(\hat{x})|\Delta\hat{x}$$

Dado un valor de $\hat{x} = 2{,}5$ con un error $\Delta\hat{x} = 0{,}01$, estimar el error resultante en la función $f(x) = x^3$.

Valor verdadero:

$f(2{,}5) = 15{,}625$

$\Delta f(\hat{x}) = |f'(\hat{x})|\Delta\hat{x} = 3 \cdot 2{,}5^2 \cdot 0{,}01 = 0{,}1875$

Ya que el valor verdadero se encuentra entre 15,4375 y 15,8125.

De hecho,

$f(2{,}49) = 15{,}4382$, $f(2{,}51) = 15{,}8132$

Estabilidad

- Un proceso numérico es **inestable** cuando pequeños errores en los datos de entrada, o errores de redondeo en alguna de las etapas el proceso, producen errores grandes en los datos de salida
- Un proceso numérico es **estable** cuando no es inestable.
- Un mismo algoritmo puede ser estable para algunos datos iniciales e inestable para otros. Entonces se dice que el algoritmo es **condicionalmente estable**.

- El producto $\otimes$ de dos números de máquina es un cálculo **estable**, sólo se pueden producir errores de desbordamiento (overflow)
- La división $\oslash$ de dos números de máquina es un cálculo **estable**, sólo se pueden producir errores de desbordamiento (overflow)
- La suma $\oplus$ de dos números de máquina es estable cuando los dos números tienen el mismo signo, y puede ser inestable cuando los dos números tienen signo distinto.

- **Suma de números con diferentes exponentes**
- **Cálculos grandes**: aunque el error de redondeo individual sea pequeño, el efecto acumulativo durante el proceso de muchos cálculos puede ser relevante.
- **Suma de un número grande y uno pequeño**
  - Sumar la serie en orden ascendente.
- **Cancelación por resta**: cuando se restan dos números de punto flotante casi iguales
  - Se puede evitar empleando una transformación.
- **Evaluación de e^x usando series infinitas**
  - Presentar $e^{-x} = \frac{1}{e^x}$.

Ejercicio 1.10

Se puede calcular e^x utilizando el desarrollo en serie de la función exponencial:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

a) Suma los 100 primeros términos de la serie para calcular e^{-30}.

b) Compara con $exp(-30)$. Explicar los resultados.

c) Formula el problema para obtener la estimación más precisa.

Bibliografía

1. **Capítulo 3** de

   Chapra S.C.; Canale R. (2005) *Métodos Numéricos para Ingenieros*. Ed. McGRaw-Hill

2. **Capítulo 1** de

   Mathews J.H., Fink K.D. (2000) *Métodos Numéricos con MATLAB*. Ed. Prentice Hall.

3. D. Goldberg, What Every Computer Scientist Should Know About Floating-Point Arithmetic

   `https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html`