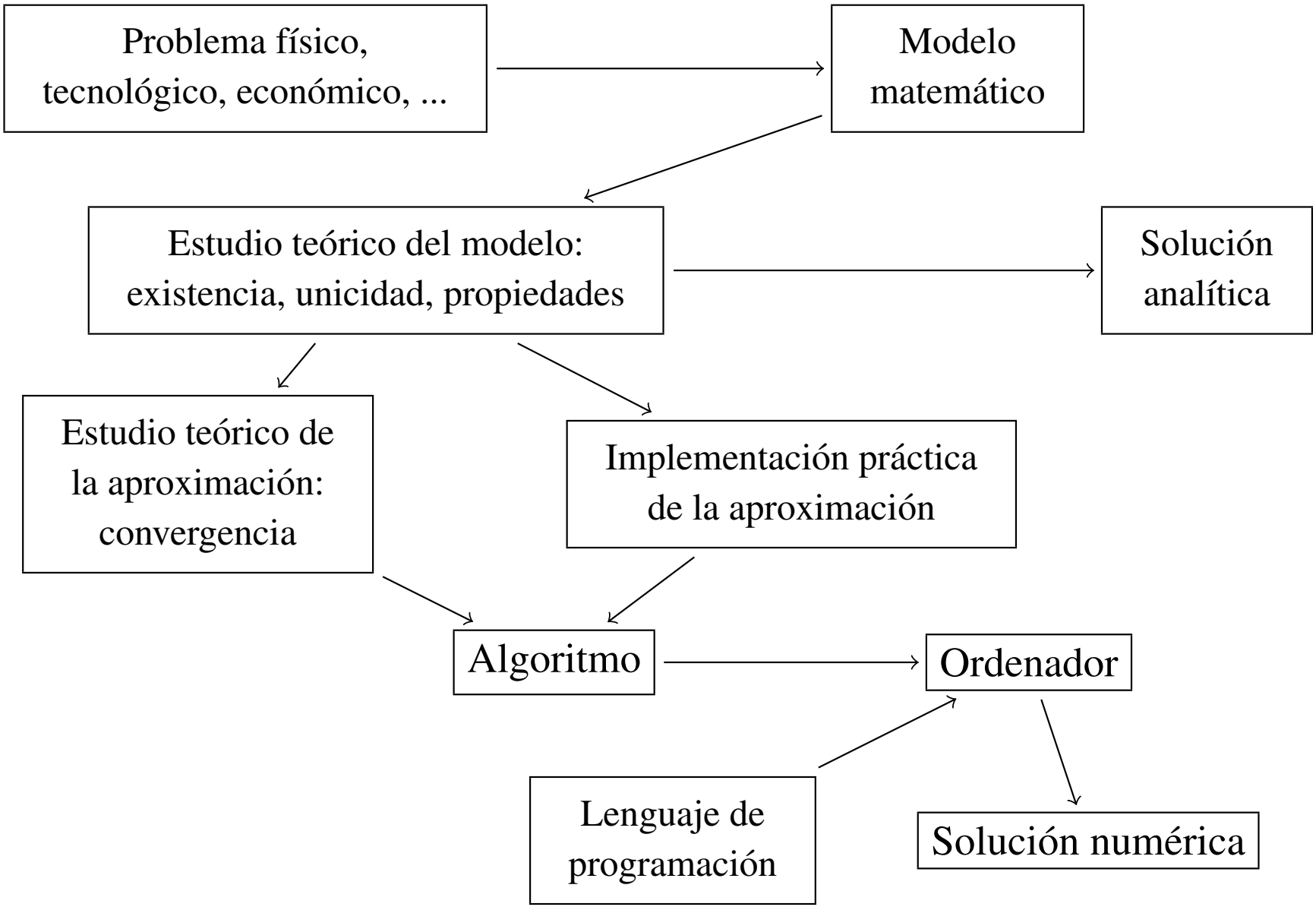
Análisis Numérico

1. Introducción. Errores

Febrero, 2007


Problema físico, tecnológico, económico, ...
Modelo matemático
Estudio teórico del modelo: existencia, unicidad, propiedades
Solución analítica
Estudio teórico de la aproximación: convergencia
Implementación práctica de la aproximación
Algoritmo
Ordenador
Lenguaje de programación
Solución numérica

Ejemplo 1

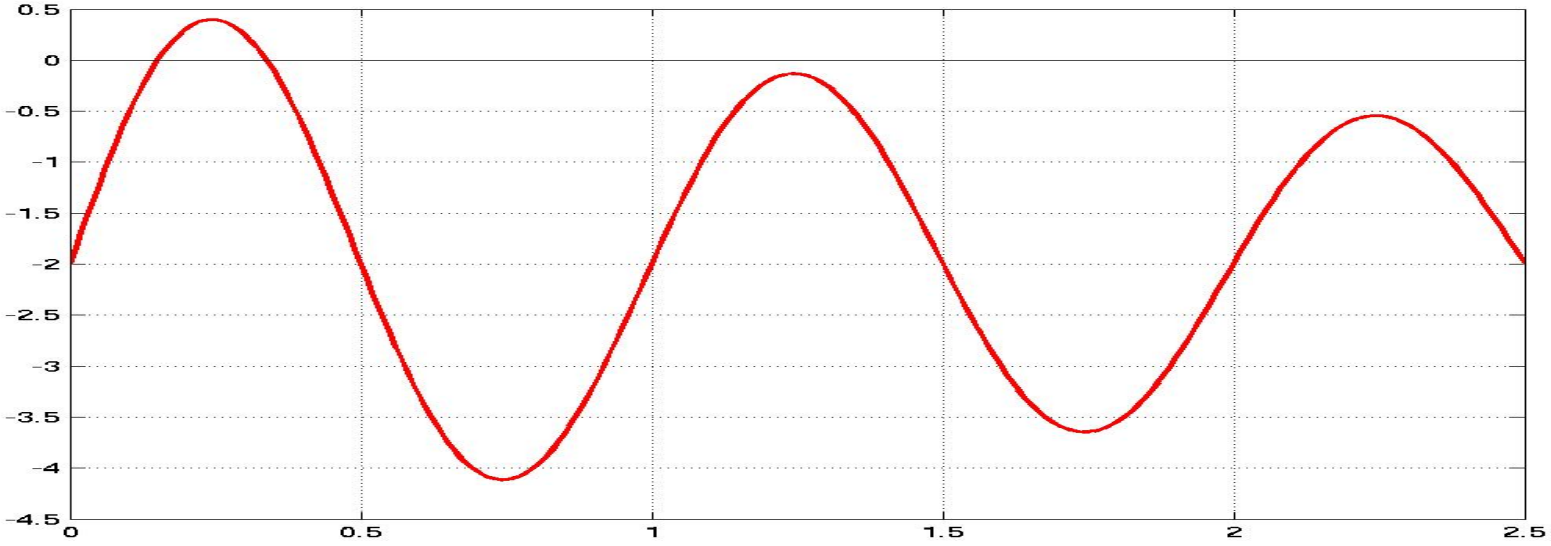
En un circuito eléctrico, la intensidad es

$$i(t) = 2.55\, e^{-0.25t} \operatorname{sen}(2\pi t)$$

Problema: queremos encontrar el instante t^* en que $i(t^*) = 2$

Modelo: encontrar el valor de t que verifica:

$$f(t) = 2.55\, e^{-0.25t} \operatorname{sen}(2\pi t) - 2 = 0$$



Ejemplo 1

Gráficamente, observamos que el modelo tiene dos soluciones. También puede deducirse utilizando el teorema de Bolzano.
Sin embargo, no disponemos de métodos analíticos para calcularlas.
Vamos a plantear un método numérico para **aproximar** la segunda de las soluciones:

$$\begin{cases} t_0 = 0.5 \\ t_{k+1} = t_k - \dfrac{f(t_k)}{f'(t_k)}, \quad k = 0, 1, 2, \ldots \end{cases}$$

Este algoritmo tiene unas condiciones de convergencia, que hay que estudiar.
Una vez implementado, obtenemos la siguiente sucesión:

$t_1 = 0.358552018 \quad t_2 = 0.338916275 \quad t_3 = 0.337305112 \quad t_4 = 0.33729374$

Dado que $f(t_4) = -3.8145 \times 10^{-6}$, daremos, como aproximación de la solución, $t^* \approx t_4 = 0.33729374$

Ejemplo 2

Los beneficios de cierta operación vienen dados por:

$$B(x,y) = (x^2 - y^2)\, e^{-(x^2+y^2)/2}$$

Problema: queremos las inversiones x e y que maximizan el beneficio

Modelo: encontrar el par (x,y) que verifica:

$$\begin{cases} B_{,x} = \left[2x - x(x^2 - y^2)\right] e^{-(x^2+y^2)/2} = 0 \\ B_{,y} = \left[-2y - y(x^2 - y^2)\right] e^{-(x^2+y^2)/2} = 0 \end{cases}$$

Algunos conceptos

Método constructivo: procedimiento que permite obtener la solución de un problema con una precisión determinada en un número finitos de pasos

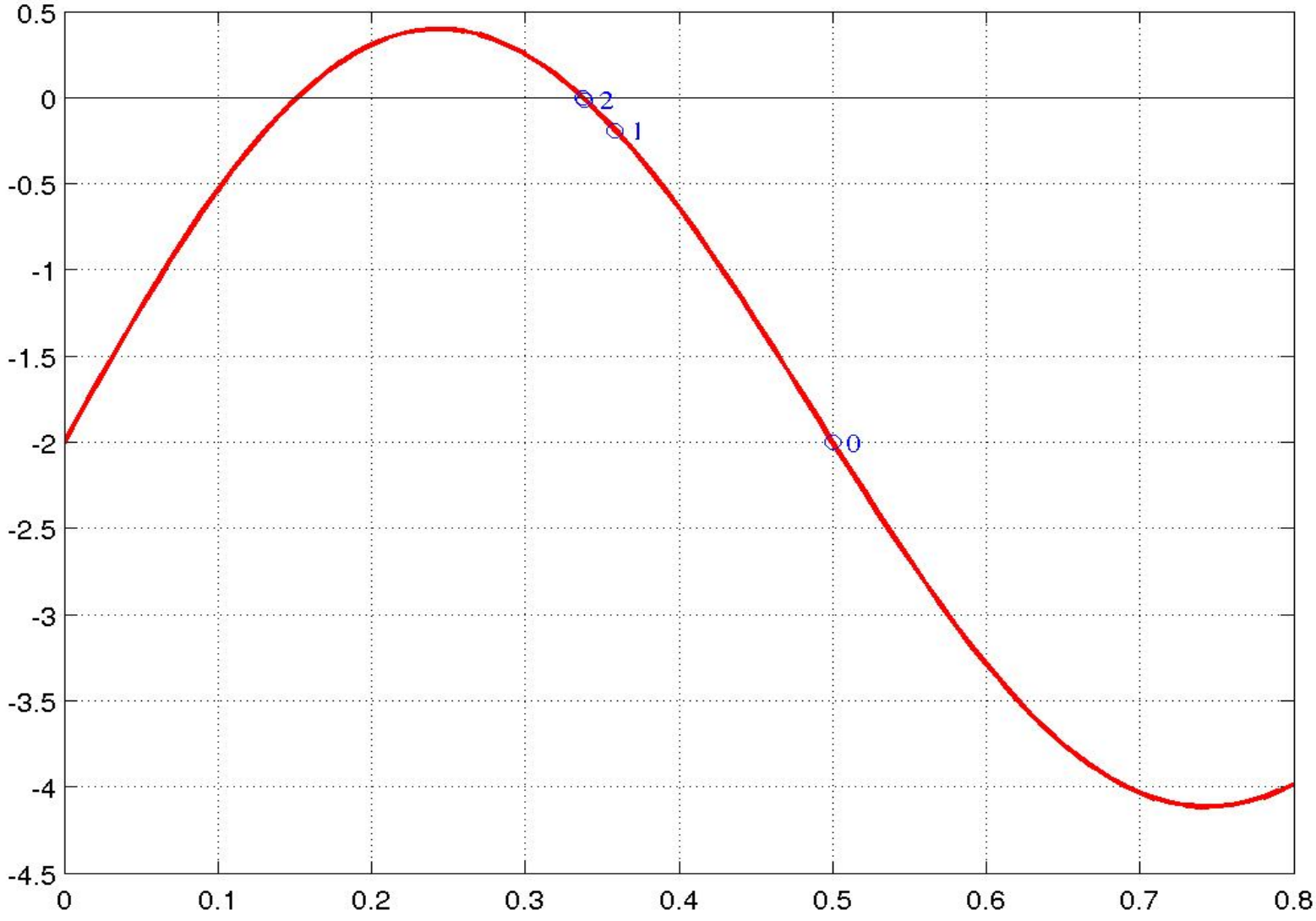
Análisis Numérico: teoría de los **métodos constructivos** en el Análisis Matemático

Ejemplo: Sea una función real de variable real, f. Para obtener un número real, α, tal que $f(\alpha) = 0$, se plantea el algoritmo de Newton–Raphson:

Dado x_0, se construye la sucesión:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad \text{si } f'(x_k) \neq 0, \text{ para } k = 0, 1, \ldots$$

$$f(t) = 2.55\,e^{-0.25t}\,\text{sen}\,(2\pi t) - 2 = 0$$



- ¿Bajo qué condiciones es convergente la sucesión $(x_k)_k$?
  En caso afirmativo, ¿converge a α?

- ¿En qué paso del proceso iterativo tendremos una aproximación razonable de α?

- Si nos detenemos en el paso k, ¿cuál es el error $|x_k - \alpha|$ cometido?

Clasificación de los problemas en Análisis Numérico

Problemas de dimensión finita: interviene un conjunto finito de números en el planteamiento del problema

- **Ejemplo 1:** Resolución de sistemas lineales ($Ax = b$)

  Datos: Matriz de orden n, vector de n componentes $\longrightarrow n^2 + n$ números

  Solución: Vector de n componentes $\longrightarrow n$ números

- **Ejemplo 2**: Raíces de ecuaciones algebraicas ($p(x) = 0$)

  Datos: Coeficientes del polinomio de grado $n \longrightarrow n + 1$ números

  Solución: n raíces de la ecuación $\longrightarrow n$ números

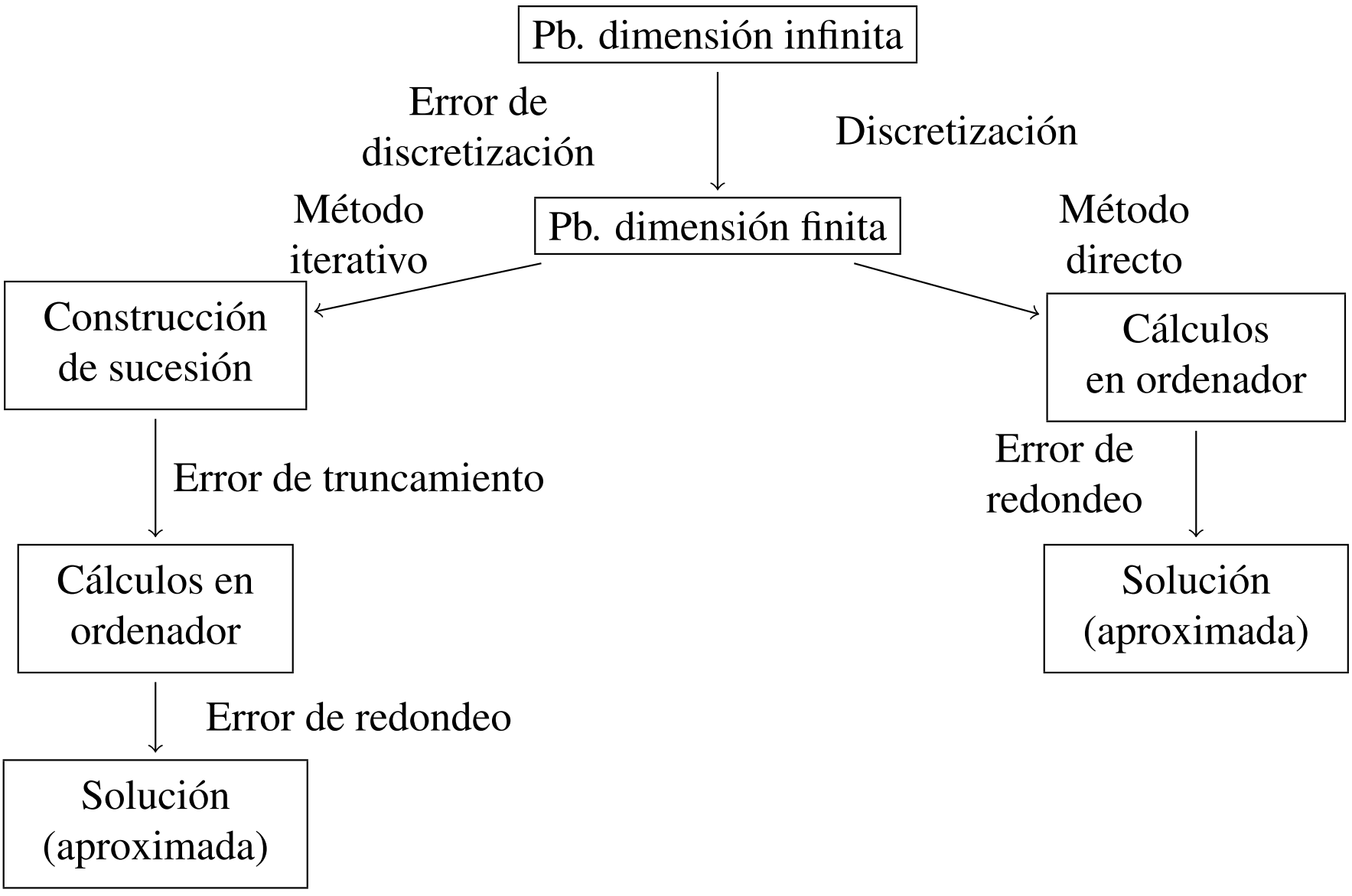
Problemas de dimensión infinita: interviene un conjunto infinito de números en el planteamiento del problema; por ejemplo, las funciones definidas en conjuntos infinitos de números dan lugar a problemas de dimensión infinita

- **Ejemplo 3**: Ecuaciones diferenciales ordinarias

  Datos: los números x_0 e y_0, que definen la condición inicial $y(x_0) = y_0$, y la función f, que define la expresión: $y' = f(x, y)$

  Solución: la función y

Problemas, métodos y errores



Los **métodos directos** permiten calcular la solución del problema en un número finito de pasos, conocido *a priori*

- en la práctica, se cometen errores (de redondeo) debido al empleo de sistemas de cálculo que usan aritmética finita

Los **métodos iterativos** construyen una sucesión diseñada para converger a la solución exacta del problema

- además de los errores de redondeo, está presente el error de truncamiento, debido al hecho de truncar la sucesión (infinita); en general, no puede calcularse exactamente (salvo que conozcamos la solución exacta), pero sí puede acotarse superiormente

Tipos de errores

Error de discretización: mide la diferencia entre la solución (continua) del problema original y la solución (discreta) del problema de dimensión finita que lo aproxima

- La solución de un PVI es una función y, mientras que obtenemos aproximaciones de $y(x_0), y(x_1), \ldots, y(x_n)$

Error de truncamiento: surge en los métodos iterativos, al tomar como aproximación de la solución un término de la sucesión que tiende a la solución

- En el ejemplo 1, hemos tomado $t^* \approx t_4$; el error de truncamiento es $|t^* - t_4|$

Error de redondeo: es consecuencia del empleo de sistemas de cálculo que utilizan aritmética finita

- La solución de una ecuación es π, pero obtenemos 3.141592654; el error de redondeo es $|\pi - 3.141592654|$

Base decimal y base binaria

- $x = (101.001101)_2 = 2^2 + 0 + 2^0 + 0 + 0 + 2^{-3} + 2^{-4} + 0 + 2^{-6} = 5.203125$

- Ejemplo de un número binario periódico:

$$x = (0.000\widehat{11000})_2$$
$$2^3 x = 8x = (0.\widehat{11000})_2$$
$$2^5 \times 2^3 x = 2^8 x = 256x = (11000.\widehat{11000})_2$$

y, restando las dos últimas igualdades,

$$(256 - 8)x = (11000)_2 = 2^4 + 2^3 = 24$$

de donde $x = \dfrac{24}{248} = \dfrac{3}{31}$.

- Sea $x = (0.7)_{10}$

$$
\begin{array}{lll}
2x = 1.4 & d_1 = 1 & r_1 = 0.4 \\
2r_1 = 0.8 & d_2 = 0 & r_2 = 0.8 \\
2r_2 = 1.6 & d_3 = 1 & r_3 = 0.6 \\
2r_3 = 1.2 & d_4 = 1 & r_4 = 0.2 \\
2r_4 = 0.4 & d_5 = 0 & r_5 = 0.4 \\
2r_5 = 0.8 & d_6 = 0 & r_6 = 0.8 \\
2r_6 = 1.6 & d_7 = 1 & r_7 = 0.6 \\
\ldots & \ldots & \ldots
\end{array}
$$

de donde $x = (0.1011001\ldots)_2$

Representación de números

Notación científica normalizada para un número $x \neq 0$ en sistema decimal:

$$x = \sigma \bar{x} 10^e$$

- $\sigma = \pm 1$: signo
- $\bar{x} \in [0.1\,,\,1)$: mantisa, significante o fracción
- $e \in \mathbb{Z}$: exponente

Ejemplo: $x = 123.45 = (+1) \times 0.12345 \times 10^3$

Representación en coma flotante de un número $x \neq 0$ en sistema decimal:

$$x = \sigma \bar{x} 10^e \qquad \text{con } \bar{x} \in [1, 10)$$

y **limitando** el número de cifras en la mantisa y el exponente

Ejemplo: $\pi = (+1) \times 3.1416 \times 10^0$

El número de dígitos en la mantisa (la parte entera más la parte decimal) se denomina **precisión** de la representación

Representación de números

Un número $x \neq 0$ puede escribirse, en sistema binario, como:

$$x = \sigma \bar{x} 2^e \qquad \text{con } (1)_2 \leq \bar{x} < (10)_2$$

Ejemplo: $x = (10101.11001)_2$

$$\begin{cases} \sigma = +1 \\ \bar{x} = (1.010111001)_2 \\ e = (100)_2 = (4)_{10} \end{cases}$$

Representación en coma flotante de números en sistema binario: la anterior, limitando el número de cifras en la mantisa y el exponente

El **estándar IEEE 754**

precisión simple: $x = \sigma(1.a_1a_2\ldots a_{23})\,2^e, \quad e \in [-126, 127]$

precisión doble: $x = \sigma(1.a_1a_2\ldots a_{52})\,2^e, \quad e \in [-1022, 1023]$

Ejemplo

$$x = (123.456789)_{10}$$
$$= (1111011.011101001111000000011111101110000010110000101\ldots)_2$$

- en notación científica normalizada

  $\bar{x} = 0.123456789 \qquad e = 3$

- en coma flotante (decimal)

  $\bar{x} = 1.23456789 \qquad e = 2$

- en coma flotante (binario)

  $\bar{x} = (1.11101101110100111100000001111110111000001011000010 1\ldots)_2$
  $e = (6)_{10} = (110)_2$

Ejemplo

$$x = (0.0987654321)_{10}$$
$$= (0.000110010100100010110000111111001101100001\ldots)_2$$

- en notación científica normalizada

  $\bar{x} = 0.987654321 \qquad e = -1$

- en coma flotante (decimal)

  $\bar{x} = 9.87654321 \qquad e = -2$

- en coma flotante (binario)

  $\bar{x} = (1.10010100100010110000111111001101100001\ldots)_2$
  $e = (4)_{10} = (100)_2$

Almacenamiento de números en el ordenador (IEEE 754)

Precisión simple: se utilizan 4 bytes (32 bits)

b_1	$b_2 \dots b_9$	$b_{10} \dots b_{32}$

- $b_1 = \begin{cases} 0, & \text{si } \sigma = +1 \\ 1, & \text{si } \sigma = -1 \end{cases}$
- $b_2 \dots b_9$ se usan para almacenar $E = e + 127$
- $b_{10} \dots b_{32}$ se usan para almacenar la parte decimal (m) de la mantisa ; la parte entera es siempre 1 salvo si $E = 0$, en cuyo caso se toma la parte entera nula

Almacenamiento de números en el ordenador (IEEE 754)

En precisión simple:

	$E=(255)_{10}$	$0<E<(255)_{10}$	$E=0$
$m\neq 0$	`NaN`	(*)	(**)
$m=0$	$\pm\infty$	(*)	0

(*) $x=(-1)^{b_1}\times 2^{E-127}\times\left(1+b_{10}\times 2^{-1}+b_{11}\times 2^{-2}+\ldots+b_{32}\times 2^{-23}\right)$

(**) Números no normalizados:

$$x=(-1)^{b_1}\times 2^{-126}\times\left(0+b_{10}\times 2^{-1}+b_{11}\times 2^{-2}+\ldots+b_{32}\times 2^{-23}\right)$$

- los números no normalizados toman valores entre 0 y el *epsilon* de la máquina

Algunos ejemplos (en precisión simple):

0	00000000	00000000000000000000000	$= 0$
0	11111111	00000000000000000000000	$= +\infty$
1	11111111	00000000000000000000000	$= -\infty$
1	11111111	00000010000000001000000	= `NaN`
1	10000001	10100000000000000000000	$= (-1)\times 2^{129-127}\times(1.101)_2 =$ $= -4\times 1.625 = -6.5$
0	00000001	10100000000000000000000	$= (+1)\times 2^{1-127}\times(1.101)_2 =$ $= 1.625\times 2^{-126} \approx$ $\approx 1.9102\times 10^{-38}$
0	00000001	00000000000000000000001	$= (+1)\times 2^{1-127}\times(1+2^{-23})$ $\approx 1.1755\times 10^{-38}$
0	00000000	00000000000000000000001 *(no normalizado)*	$= (+1)\times 2^{-126}\times(0+2^{-23})$ $= 2^{-149} \approx 1.4013\times 10^{-45}$

`http://www.h-schmidt.net/FloatApplet/IEEE754.html`

Almacenamiento de números en el ordenador (IEEE 754)

Precisión doble: se utilizan 8 bytes (64 bits)

b_1	$b_2 \ldots b_{12}$	$b_{13} \ldots b_{64}$

- $b_1 = \begin{cases} 0, & \text{si } \sigma = +1 \\ 1, & \text{si } \sigma = -1 \end{cases}$
- $b_2 \ldots b_{12}$ se usan para almacenar $E = e + 1023$
- $b_{13} \ldots b_{64}$ se usan para almacenar la parte decimal (m) de la mantisa

	$E = (2047)_{10}$	$0 < E < (2047)_{10}$	$E = 0$
$m \neq 0$	`NaN`	(*)	números no normalizados
$m = 0$	$\pm\infty$	(*)	0

$$(*) \quad x = (-1)^{b_1} \times 2^{E-1023} \times \left(1 + b_{13} \times 2^{-1} + b_{14} \times 2^{-2} + \ldots + b_{64} \times 2^{-52}\right)$$

Exactitud de la representación en coma flotante

Epsilon de la máquina: es la diferencia entre 1 y el siguiente número mayor que 1 que puede ser almacenado

- en simple precisión, $\varepsilon = 2^{-23} \approx 1.19 \times 10^{-7}$
- en doble precisión, $\varepsilon = 2^{-52} \approx 2.22 \times 10^{-16}$

El mayor entero, M, tal que pueden representarse (y almacenarse) de forma exacta todos los enteros positivos menores o iguales a M

- en simple precisión, $M = 2^{24} = 16777216$
- en doble precisión, $M = 2^{53} \approx 9.0 \times 10^{15}$

Por ejemplo, supongamos un sistema con una precisión de 4 dígitos; podemos representar:

$$(1.111)_2 \times 2^3 = (1111)_2 = 2^4 - 1 = 15$$
$$(1.000)_2 \times 2^4 = 2^4 = 16$$

pero el siguiente entero que podemos representar de forma exacta es:

$$(1.001)_2 \times 2^4 = \left(1 + \frac{1}{8}\right) \times 2^4 = 18$$

Exactitud de la representación en coma flotante

Error de **Overflow**: se produce cuando se intenta usar números demasiado grandes para el formato de coma flotante correspondiente

Underflow: se produce cuando se intenta usar números demasiado pequeños; normalmente se toman como *cero* y continúan las operaciones

Aproximación por redondeo y redondeo a cero

Sea un número x que representamos en notación científica normalizada:

$$x = \pm 0.d_1 d_2 d_3 \ldots \times 10^n = \pm \left(\sum_{k=1}^{+\infty} d_k \times 10^{-k} \right) \times 10^n$$

con $1 \leq d_1 \leq 9$ y $0 \leq d_k \leq 9$ $(k = 2, 3, \ldots)$

La aproximación de x por otro número con p decimales en la mantisa puede hacerse de una de las dos formas siguientes:

- truncamiento, o **redondeo a cero**:

$$\widehat{x}_0 = \pm 0.d_1 d_2 d_3 \ldots d_p \times 10^n$$

- **redondeo**:

$$\widehat{x} = \begin{cases} \pm 0.d_1 d_2 d_3 \ldots d_p \times 10^n \,, & \text{si } 0 \leq d_{p+1} \leq 4 \\ \pm (0.d_1 d_2 d_3 \ldots d_p + 10^{-p}) \times 10^n \,, & \text{si } 5 \leq d_{p+1} \leq 9 \end{cases}$$

Aproximación por redondeo y redondeo a cero

Ejemplo: $x = 0.99995 \times 10^0$; con $p = 4$ decimales en la mantisa obtenemos:

redondeo a cero: $\widehat{x}_0 = 0.9999 \times 10^0$

redondeo: $\widehat{x} = 0.1 \times 10^1$

Ejemplo: $x = 0.4332609 \times 10^0$; con $p = 3$ decimales en mantisa obtenemos:

$\widehat{x} = \widehat{x}_0 = 0.433 \times 10^0$

Aproximación por redondeo y redondeo a cero

En sistema binario,

- **redondeo a cero**: se almacenan los p primeros dígitos de la mantisa
- **redondeo**: se almacenan los p primeros dígitos de la mantisa, sumando 1 al último de ellos si el dígito $p+1$ es no nulo

Ejemplo: $x = (1.10011)_2 = (1.59375)_{10}$; para una precisión de 5 cifras tenemos:

con redondeo a cero: $\widehat{x}_0 = (1.1001)_2 = (1.5625)_{10}$

con redondeo: $\widehat{x} = (1.1010)_2 = (1.625)_{10}$

Los errores por redondeo pueden cancelarse, y su efecto se amplifica menos

Error de redondeo y estabilidad numérica

- Los errores de redondeo surgen al operar con precisión finita
- Son pequeños en cada cálculo concreto
- Al realizar grandes cantidades de cálculos, los **errores de redondeo** pueden acumularse y propagarse, de modo que la diferencia entre la solución exacta y el resultado final de los cálculos sea grande. Este fenómeno se conoce como **inestabilidad numérica**

  `http://www.ima.umn.edu/~arnold/disasters/disasters.html`

Ejemplo: los términos de la sucesión dada por:

$$x_k = \left(\frac{1-\sqrt{5}}{2}\right)^k, \quad k = 0, 1, 2, \ldots$$

pueden obtenerse de dos formas:

(a) $x_0 = 1, x_1 = (1-\sqrt{5})/2, \quad x_k = x_{k-1} + x_{k-2} \quad (k \geq 2)$

(b) $x_0 = 1, x_k = \left((1-\sqrt{5})/2\right) x_{k-1} \quad (k \geq 1)$

Tras 100 iteraciones con un ordenador, obtenemos:

(a) $\widehat{x}_{100} = -1.18921493 \times 10^4$

(b) $\widehat{x}_{100} = -2.04278949 \times 10^{-21}$

Error absoluto y error relativo

Dada una aproximación $\widehat{x}$ del número x, se definen:

- **Error absoluto** entre x y $\widehat{x}$:

$$e_a = |x - \widehat{x}|$$

- **Error relativo** entre $x \neq 0$ y $\widehat{x}$:

$$e_r = \frac{|x - \widehat{x}|}{|x|} = \frac{e_a}{|x|}$$

Ejemplo:

x	$\widehat{x}$	e_a	e_r
0.3000×10^{1}	0.3100×10^{1}	0.1	$0.333\widehat{3} \times 10^{-1}$
0.3000×10^{-3}	0.3100×10^{-3}	0.1×10^{-4}	$0.333\widehat{3} \times 10^{-1}$
0.3000×10^{4}	0.3100×10^{4}	0.1×10^{3}	$0.333\widehat{3} \times 10^{-1}$

Error absoluto y error relativo

Se dice que $\widehat{x}$ aproxima a x con t **cifras o dígitos significativos**, si t es el mayor entero no negativo tal que:

$$\frac{|\widehat{x}-x|}{|x|} \leq 5\times 10^{-t}$$

Ejemplos:

- $\widehat{x} = 124.45$ aproxima a $x = 123.45$ con **2** cifras significativas:

$$\frac{|\widehat{x}-x|}{|x|} = \frac{1.}{123.45} \approx 8.1\times 10^{-3} \leq 5\times 10^{-2}$$

- $\widehat{x} = 0.0012445$ aproxima a $x = 0.0012345$ con **2** cifras significativas:

$$\frac{|\widehat{x}-x|}{|x|} = \frac{0.00001}{0.0012345} \approx 8.1\times 10^{-3} \leq 5\times 10^{-2}$$

- $\widehat{x} = 999.8$ aproxima a $x = 1000$ con **4** cifras significativas:

$$\frac{|\widehat{x}-x|}{|x|} = \frac{0.2}{1000} = 2\times 10^{-4} \leq 5\times 10^{-4}$$

Condicionamiento

Se dice que un **problema** matemático está **bien condicionado** cuando pequeñas variaciones de los datos producen pequeñas variaciones en la solución. Cuando esto no ocurre, nos referimos a **problemas mal condicionados**

Ejemplos de problemas mal condicionados:

- En la resolución de ciertos sistemas de ecuaciones lineales, pequeñas variaciones de sus coeficientes pueden producir grandes variaciones en la solución
- En algunos polinomios, pequeñas variaciones de sus coeficientes hacen variar mucho el valor de sus raíces (condicionamiento en el cálculo de raíces de polinomios); por ejemplo:

$$p(x) = \quad x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720$$
$$q(x) = 1.1x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720$$

El condicionamiento de un problema es **independiente del método numérico** empleado para resolverlo

Condicionamiento en la evaluación de una función

Pretendemos ver el comportamiento del valor de una función real de variable real cuando modificamos muy poco el valor de la variable

Sea x un número real y $h > 0$; consideramos los números x y $x+h$

- Error relativo entre x y $x+h$:

$$e_r(x) = \frac{|(x+h)-x|}{|x|} = \frac{h}{|x|}$$

- Error relativo entre $f(x)$ y $f(x+h)$:

$$e_r(f(x)) = \frac{|f(x+h)-f(x)|}{|f(x)|} = \frac{h\,|f'(\xi)|}{|f(x)|} \approx \frac{|xf'(x)|}{|f(x)|}\,\frac{h}{|x|}$$

donde hemos utilizado el desarrollo en serie de Taylor, de modo que:

$$e_r(f(x)) = \frac{|xf'(x)|}{|f(x)|} e_r(x)$$

Condicionamiento en la evaluación de una función

Ejemplo de función bien condicionada: $f(x) = \sqrt{x}$

$$e_r(\sqrt{x}) = \frac{1}{2} e_r(x)$$

En el punto $a = 1.0$ y tomando $\varepsilon = 10^{-5}$,

$$f(a) = 1.0 \qquad f(a+\varepsilon) = 1.0000049999875$$

$$\frac{|f(a+\varepsilon) - f(a)|}{|f(a)|} = 4.9999875000031722 \times 10^{-6}$$

El error relativo en la solución es la mitad del error relativo en los datos

Condicionamiento en la evaluación de una función

Ejemplo de función mal condicionada: $f(x) = \arcsin(x)$
(para x próximo a 1)

$$e_r(\operatorname{arcsen} x) = \left| \frac{x}{\sqrt{1-x^2}\operatorname{arcsen}(x)} \right| e_r(x)$$

En el punto $a = 0.99$ y tomando $\varepsilon = 10^{-5}$,

$$f(a) = 1.42925685\ldots \qquad f(a-\varepsilon) = 1.42918598\ldots$$

$$\frac{|f(a-\varepsilon) - f(a)|}{|f(a)|} = 4.95855642 \times 10^{-5}$$

El error se ha multiplicado por cinco !