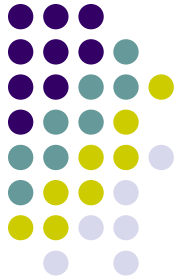
Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias

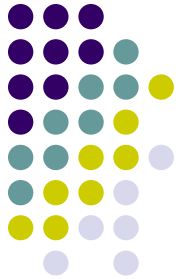
¿Qué es la física computacional?



La física computacional es una nueva manera de hacer investigación en física, próxima al experimento y a la teoría.

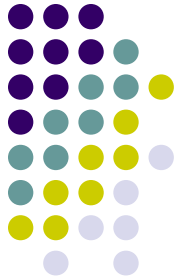
En el laboratorio se realizan mediciones en sistemas físicos reales, luego los teóricos explican esas mediciones mediante las teorías.

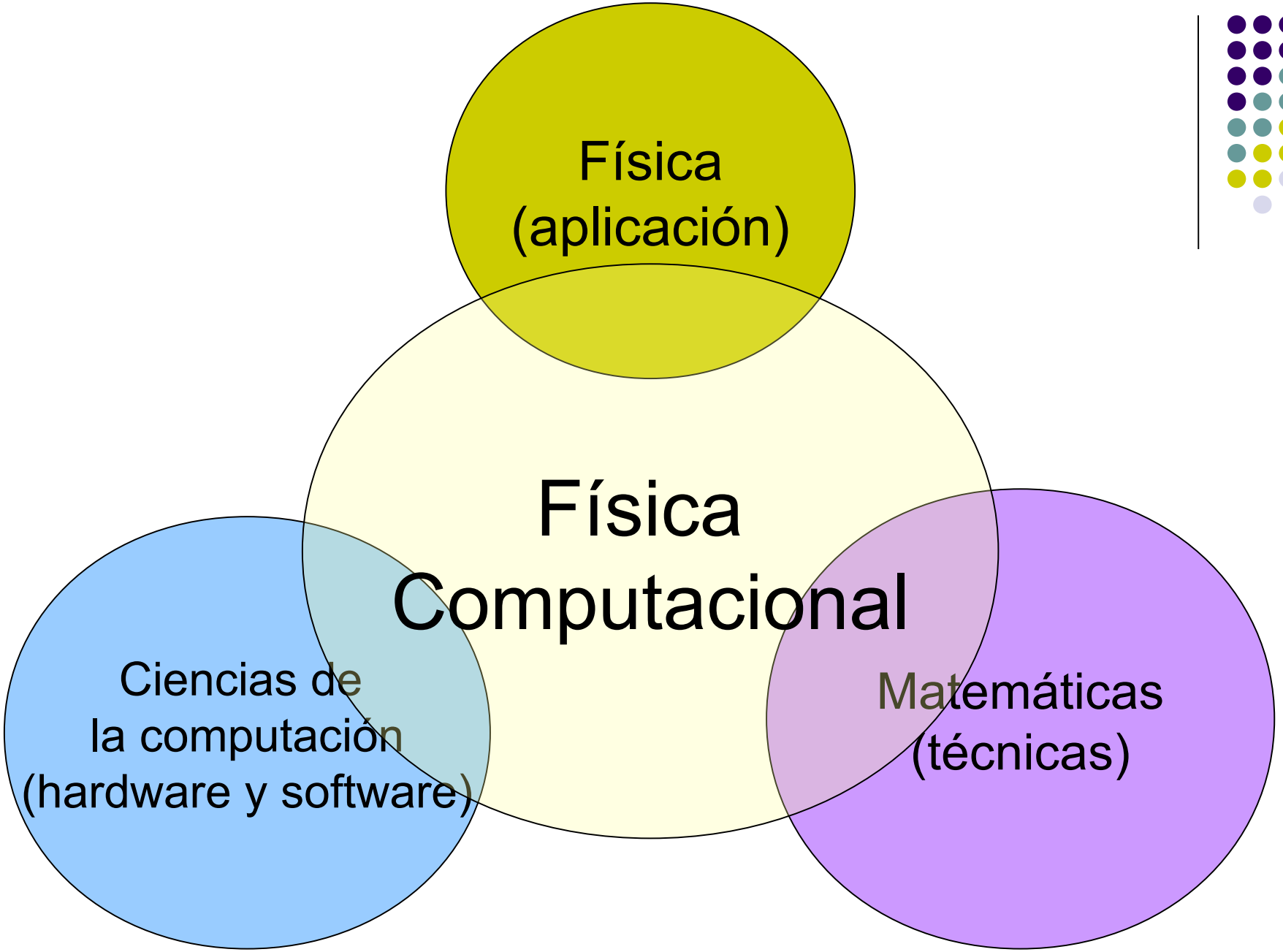
Áreas de investigación en la física



- ✔ Problemas que no tienen solución analítica.
- ✔ Validar aproximaciones y hacer efectivas las teorías propuestas.
- ✔ Comparar cuantitativamente teorías y mediciones experimentales.
- ✔ Visualizar conjuntos de datos complejos.
- ✔ Control y medición de experimentos.

- Predicción del clima.
- Superconductividad.
- Genoma Humano.
- Visión y lenguaje.
- Fusión nuclear.
- Oceanografía.
- Ciencia de los materiales.
- Diseño de semiconductores.
- Astrofísica relativista.
- Sistemas de combustión
- Estructura biológica.
- Diseño de fármacos.
- Turbulencia.
- Recuperación de petróleo y gas.
- Cromodinámica cuántica.




Física
(aplicación)
Física
Computacional
Ciencias de
la computación
(hardware y software)
Matemáticas
(técnicas)

Método numérico

Se puede representar como una cadena de algoritmos A_i (i = 1, 2, 3, … , *N*) en la entrada y salida.

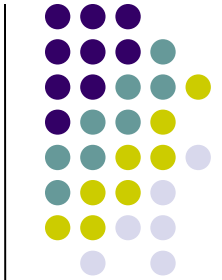
Entrada

Datos iniciales

$$A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \cdots \rightarrow A_N$$

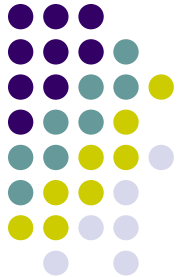
Salida

Solución numérica

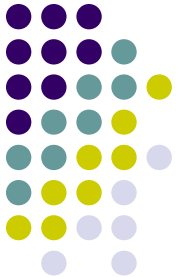


La solución obtenida por un método numérico es ***aproximada***, es decir, hay cierta diferencia entre la solución exacta y la solución numérica.

Las principales causas de la diferencia son las siguientes:

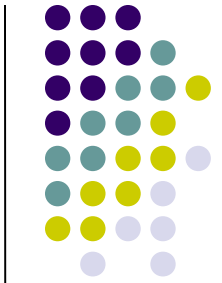


- Falta de correspondencia entre el problema (modelo) matemático y el fenómeno físico real.
- Errores en los datos iniciales (parámetros de entrada)
- Errores en el método numérico usado para resolver el problema.
- Errores de redondeo en las operaciones aritméticas.



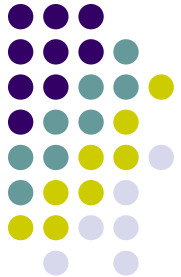
Conceptos principales relacionados con modelos y métodos numéricos

Aproximación



- Es la proximidad de un modelo numérico al modelo original (diferencial, integral, etc.) o el grado de aproximación, caracteriza el error que se introduce al hacer discreto el modelo continuo.

El grado de aproximación *n* se estima mediante un factor que tiene el error entre dos modelos.

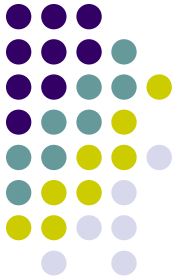


- Este factor tiene la forma h^n en el método de diferencias finitas, donde h es el tamaño de la malla.
- Es de la forma N^{-n} en un método de proyección (el método de colocación, el método de Rayleigh-Ritz, el método de Galërkin) donde N es el número de truncación de las series de Fourier.

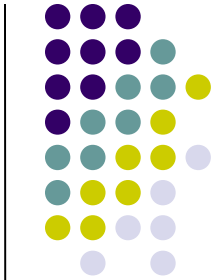
Estabilidad

- Caracteriza la manera de propagación de los errores iniciales dentro del algoritmo en el proceso del cálculo.
- Si el incremento de errores iniciales es considerable y sin ningún control, entonces el método numérico se llama inestable.
- Si los errores de cálculos dependen continuamente de los errores iniciales, entonces el método se llama estable.

Convergencia

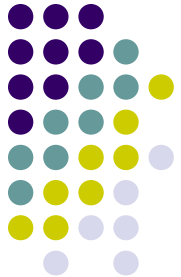


- Significa que la solución numérica converge hacia la solución exacta cuando el tamaño de la malla *h* tiene a cero, o el número de truncación *N* tiende al infinito.



Errores en los métodos numéricos

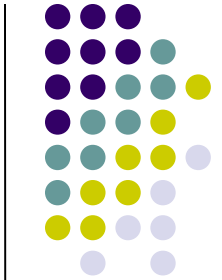
Errores en los métodos numéricos



- **Truncamiento**: se debe a las aproximaciones utilizadas en la fórmula matemática del modelo.

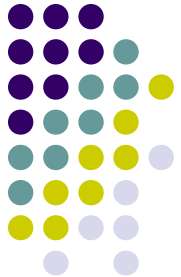
- **Redondeo** : se asocia al hecho de que la representación de un número en la computadora se hace con un conjunto limitado de dígitos.

Series de Taylor



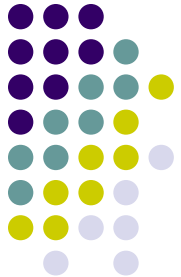
- Las soluciones numéricas son en su mayoría, aproximaciones de las soluciones exactas.

- Gran parte de los métodos numéricos se basan en la aproximación de funciones por medio de polinomios.



El desarrollo de Taylor es una serie infinita de potencias, representa de manera exacta a una función dentro de un cierto radio alrededor de un punto dado.

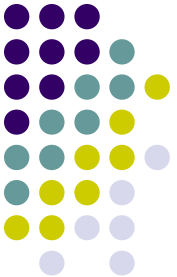
Al comparar el desarrollo polinomial de la solución numérica con la serie de Taylor de la solución exacta, es posible evaluar el error, conocido como *error de truncamiento.*



Si se ignoran todos los términos de la serie de Taylor, excepto algunos cuantos, se puede obtener un polinomio que se aproxime a la función verdadera.

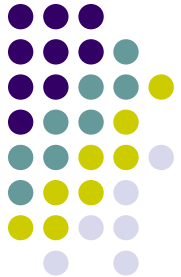
A este polinomio se le llama *serie de Taylor truncada* y se usa como punto de partida para obtener métodos numéricos.

Definición de Serie de Taylor



Una función $f(x)$ es analítica en $x=a$ si $f(x)$ se puede representar por medio de una serie de potencias en términos de $h = x-a$ dentro de un radio de convergencia $D > |x-a| > 0$

Una condición necesaria para que una función sea analítica es que todas sus derivadas sean continuas tanto en $x=a$ como en alguna vecindad alrededor de ese punto.



Un punto en donde una función *f(x)* no es analítica recibe el nombre de *punto singular*.

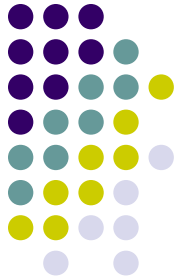
Si *f(x)* es diferenciable en todas las partes de la vecindad de x_0, excepto en x_0 , entonces x_0 es un punto singular.

Los polinomios son analíticos en todas partes.

Si *f* es analítica alrededor de *x* = *a*, se puede representar *f(x)* de manera exacta en la vecindad de x = a por medio de una serie de Taylor, dada por:

$$f(x)=f(a)+hf'(a)+\frac{h^2}{2}f''(a)+\frac{h^3}{6}f'''(a)+\cdots$$

$$\cdots+\frac{h^m}{m!}f^{(m)}(a)+\cdots$$



- La serie de Taylor es única, esto quiere decir que no existe otra serie de potencias en

  $h = x-a$ para representar $f(x)$

- El desarrollo de Taylor de una función alrededor de $x = 0$ recibe el nombre de *serie de Maclaurin*.

En aplicaciones prácticas, se debe de truncar la serie de Taylor después cierto orden, ya que es imposible incluir un número infinito de términos. Si la serie se trunca después del término *N*, se expresa por:

$$f(x)=f(a)+hf'(a)+\frac{h^2}{2}f''(a)+\frac{h^3}{6}f'''(a)+\cdots$$

$$\cdots+\frac{h^N}{N!}f^{(N)}(a)+O(h^{N+1})$$

Donde $h = x - a$ y $O(h^{N+1})$ representa el error por el truncamiento de los términos de orden *N+1*

El error global se puede representar como:

$$O(h^{N+1}) = f^{(N+1)}(a+\xi h)\frac{h^{N+1}}{(N+1)!} \qquad 0 \leqslant \xi \leqslant 1$$

Dado que ξ no puede calcularse con exactitud, se aproxima el término del error, haciendo

$\xi = 0$

$$O(h^{N+1}) \approx f^{(N+1)}(a)\frac{h^{N+1}}{(N+1)!}$$

Si *N* = *1*, la serie de Taylor truncada es:

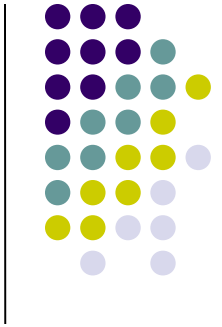
$$f(x) \approx f(a) + f'(a)h \qquad h = x - a$$

Incluyendo el efecto del error, tenemos que:

$$f(x) \approx f(a) + f'(a)h + O(h^2)$$

donde:

$$O(h^2) \approx f''(a + \xi h)\frac{h^2}{2} \qquad 0 < \xi < 1$$



Representación de los números en las computadoras

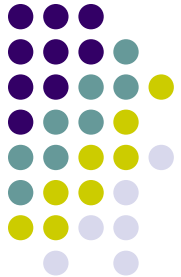
Base decimal

El valor decimal de un número de base r es:

$$(abcdefg.hijk)_r$$

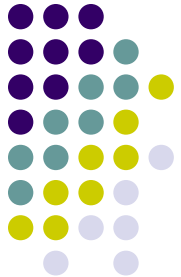
Que se calcula como:

$$ar^6+br^5+cr^4+dr^3+er^2+fr+g$$
$$+hr^{-1}+ir^{-2}+jr^{-3}+kr^{-4}$$



La menor y mayor magnitud de un número real que se pueden representar en una computadora, varían de acuerdo con el diseño tanto de hardware como de software.

Los número reales en una computadora no son continuos. Si nos fijamos en número cercanos a cero, el número positivo más pequeño en una IBM es 2.9×10^{-39}



Por tanto, no se pueden representar números entre 0 y 2.9×10^{-39}

A este intervalo se le conoce como *épsilon de la máquina*.

Épsilon de la máquina

Hay dos maneras de definir el épsilon de la maquina: un épsilon absoluto y un épsilon relativo.

Éste último es el más usado. Como el conjunto de números en la computadora es finito, la siguiente definición tiene sentido:

$$\epsilon_{maq} = e = min\left[t > 0 : 1 + t > 1\right]$$

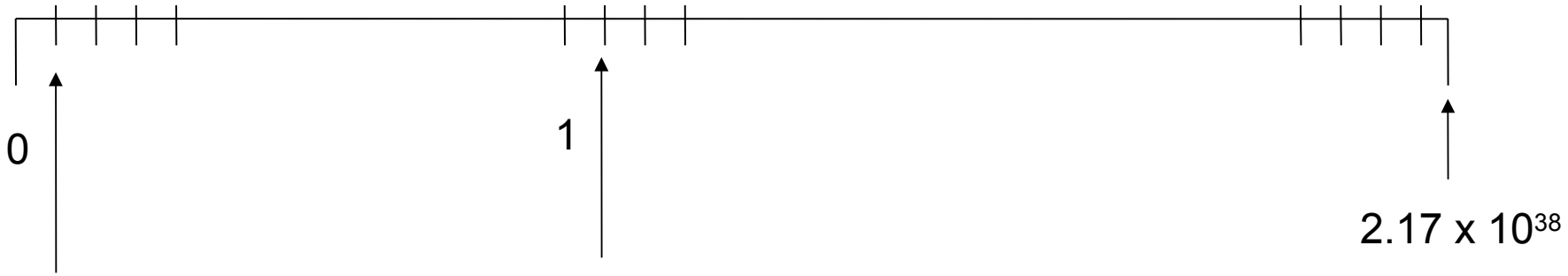
Épsilon de la máquina

El épsilon absoluto se define comparando con cero:

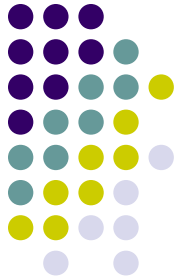
$$\epsilon_{absoluto} = e = min\left[t > 0 : t \neq 0\right]$$

En realidad el épsilon depende de la máquina pero también del sistema operativo, del compilador y del tipo de números utilizados.

Distribución de números reales en una IBM PC

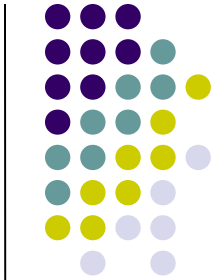


Cálculo del épsilon de la máquina



```
PROGRAM ceromaquina
    t = 1;
    DO WHILE (1 + t .NE. 1)
        eps = t
        t = t / 2
    END DO
    PRINT *, 'El cero de la máquina es : ' eps
END PROGRAM ceromaquina
```

Cálculo del épsilon de la máquina



```
PROGRAM ceromaquina2
t = 1
uno = 1
t1 = uno + t
DO WHILE (uno .NE. t1)
   eps = t
   t=t / 2
   t1 = uno + t
END DO
PRINT *, “El cero de la máquina es : “ , eps
END PROGRAM ceromaquina2
```

Error de truncamiento

Se originan al emplear al número finito de términos para calcular un valor que requiere un número infinito de términos.

Por ejemplo, una expresión que permite determinar de forma exacta el valor del número de Euler (base de los logaritmos naturales) a través de una serie de MacLaurin es:

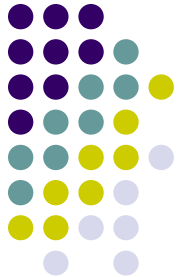
$$e^x = \sum \frac{x_i}{i!}$$

Sin embargo, una aproximación a dicho valor, puede obtenerse a través de su expresión finita:

$$e^{x} \approx \sum_{i=0}^{k} \frac{x_i}{i!} \qquad k < \infty$$

Es claro que esta expresión finita es manejable *computacionalmente* hablando, al contrario que la fórmula expresada en su forma infinita.

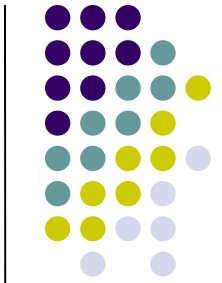
Error de redondeo



Se origina por el hecho de que una computadora sólo puede representar un número finito de términos. Para expresar una cantidad con un desarrollo decimal infinito, se tiene que prescindir de la mayoría de ellos.

Por ejemplo, el número π = 3.14159265...., tiene un desarrollo decimal infinito no periódico. Por lo tanto, para fines de cálculo, sólo se toman algunos de sus dígitos.

Estrategias

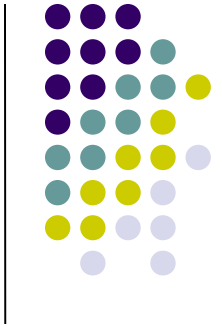


• Redondeo. Prescinde de cierto número de cifras significativas y realiza un ajuste, sobre la última cifra no descartada :

$\pi = 3.1416$

• Corte o poda: Prescinde de cierto número de cifras significativas sin realizar un ajuste sobre la última cifra no descartada:

$\pi = 3.1415$



Una vez que se ha establecido la clasificación del error, definimos los conceptos de error absoluto verdadero, error absoluto relativo, error absoluto aproximado y error relativo aproximado, todos ellos como una suma o consecuencia de los errores de redondeo y truncamiento.

Error absoluto verdadero

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

El error absoluto verdadero se define con la siguiente expresión:

$$E_v = |p - \hat{p}|$$

Esta definición de error, lo cuantifica en términos brutos. No obstante, una medida que puede describir con mayor detalle o proporción el error, es aquella que lo expresa en términos porcentuales. Para ello se emplea el error verdadero relativo.

Error relativo verdadero

Supóngase que $\hat{p}$ es una aproximación a p. El error relativo verdadero se calcula con la siguiente expresión:

$$e_v = \frac{|p - \hat{p}|}{p} \qquad p \neq 0$$

El resultado suele expresarse en términos porcentuales.

Error relativo aproximado

El error relativo aproximado, mide el error de un método numérico, determinando el error de la iteración actual respecto el error surgido en la iteración anterior:

$$e_a = \frac{\left| \hat{x}_i - \hat{x}_{i-1} \right|}{\hat{x}_i}$$

Donde x_i = aproximación actual a x.

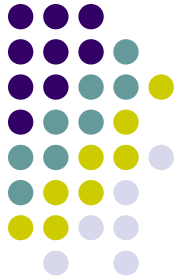
x_{i-1} = aproximación anterior a x.

En métodos numéricos suele establecerse una tolerancia porcentual como criterio de paro, tal que el error relativo aproximado de un método, no exceda dicha tolerancia.

$$e_a < t$$

donde t, es tolerancia fijada de antemano. A menor tolerancia se tiene mayor precisión en la aproximación al valor verdadero, sin embargo esto implica un aumento en el número de iteraciones requeridas para detener el método.

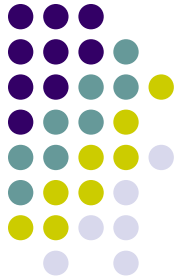
Observaciones sobre la tolerancia t de un método numérico



Puede demostrarse que si el siguiente criterio se cumple, se tiene la seguridad que un resultado es correcto en al menos *n* dígitos significativos:

$$t = \left\lfloor \frac{10^{2-n}}{2} \right\rfloor \text{‰}$$

Consideraciones al momento de realizar programas



En los dos siguientes ejemplos veremos cómo los cálculos nos generan problemas:

1. cuando se suma (o se resta) un número muy pequeño de uno muy grande.
2. cuando un número se resta de otro que es muy cercano.

Sumando un número muy pequeño a uno grande

```
PROGRAM ejem1
    suma=1
    DO i=1,10000
        suma = suma+0.00001
    END DO
    PRINT *, suma
END PROGRAM ejem1
```

¿De dónde viene este error?

Valor exacto de la suma = 1.1

Valor devuelto por el programa = 1.1001358

Error relativo = 1.2343E-02 %

Números reales

El formato para un número real en una computadora difiere según el diseño de hardware y software.

Los números reales se almacenan en el formato de punto flotante normalizado en binario. En precisión simple, se usan 4 bytes, es decir, 32 bits.

Si se introduce un número decimal como dato, primero se convierte al binario más cercano en el formato normalizado:

$$(\pm 0.abbbbbbbbb \ldots bbbb)_2 \times 2^z$$

donde *a* siempre es 1, cada *b* es un dígito binario (0, 1) y *z* es un exponente que también se expresa en binario.

Distribución de dígitos

Existen 24 dígitos para la mantisa, incluyendo la *a* y las *b.* Los 32 bits se distribuyen de la siguiente manera:

a. El primer bit se usa para el signo de la mantisa.

b. Los siguientes 8 bits para el exponente *z*.

c. Los últimos 23 para la mantisa.

11111111 11111111 11111111 11111111

En el formato de punto flotante normalizado, el primer dígito de la mantisa siempre es 1, por lo que no se almacena físicamente, por tanto, una mantisa de 24 bits, se almacena en 23.

Exponentes en binario

Si los 8 dígitos asignados al exponente se usan sólo para enteros positivos, el exponente puede representar desde 0 hasta 28 − 1 = 225, aunque puede incluir a números negativos.

Para el manejo de exponentes positivos o negativos, el exponente en decimal es sesgado (o sumado) con 128 y después convertido en binario (completo a dos)

Por ejemplo, si el exponente es −3, entonces −3 + 128 = 125 que se convierte a binario y se almacena en los 8 bits.

Los exponentes que se pueden almacenar en 8 bits, van desde 0 − 128 = −128 hasta 255 − 128 = 127

Errores por redondeo

Consideremos el cálculo de 1 + 0.00001. Las representaciones binarias de 1 y 0,00001 son, respectivamente:

$$(1)_{10} = (0.1000\,0000\,0000\,0000\,0000\,0000)_2 \times 2^1$$

$$(0.00001)_{10} = (0.1010\,0111\,1100\,0101\,1010\,1100)_2 \times 2^{-16}$$

La suma de estos dos números es:

$$(1)_{10}+(0.00001)_{10}=$$
$$=(0.1000\,0000\,0000\,0000\,0101\,0011\,1100\,1011\,0101\,100)_2\times 2^1$$

Sin embargo, éste número se redondea ya que la mantisa usa 24 bits, por lo que el número se guarda como:

$$(0.1000\,0000\,0000\,0000\,0101\,0100)_2\times 2^1$$

que es equivalente a: $(1.0000000136)_{10}$

Conclusión

Por lo que siempre que se sume 0.00001 a 1, el resultado agrega 0.0000000136 como error.

Al repetir 10 000 veces la suma de 0.00001 a 1, se genera un error de diezmil veces.

Otra manera de sumar

```
PROGRAM ejem1
suma=1
DO i=1,10000
    suma = suma+0.00001
END DO
PRINT *, suma
END PROGRAM ejem1
```

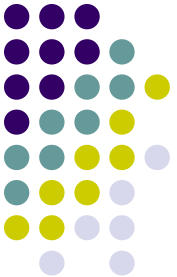
Valor exacto = 1.1

Valor numérico = 1.1000047

Error relativo = 4.2265E-04

```
PROGRAM ejem2
suma=1
DO i=1,100
    total = 0
    DO k=1,100
        total = total+0.00001
    END DO
    suma = suma+total
END DO
PRINT *, suma
END PROGRAM ejem2
```

Errores por redondeo al restar

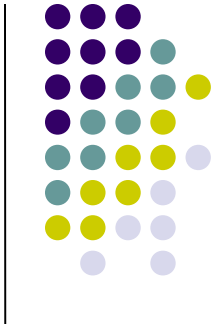


Sabemos del cálculo que

$$\lim_{\theta \to 0} \frac{f(x+\theta)-f(x)}{\theta} = f'(x)$$

Si hacemos *f(x)* = *sen(x)*, revisamos para

$$d = \frac{\sin(1+\theta)-\sin(1)}{\theta}$$



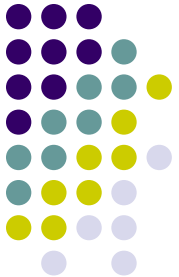
Cuando tiende θ a 0, la precisión se vuelve muy pobre debido a los errores de redondeo. Por medio de un desarrollo de Taylor, podemos re-escribir la ecuación de tal forma que se mejore la exactitud para θ.

El desarrollo de Taylor de sen(1+θ) es

$$\sin(1+\theta)=\sin(1)+\theta\cos(1)-0.5\,\theta^2\sin(1)\cdots$$

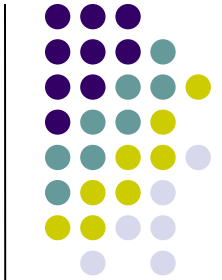
Si aproximamos sen(1+θ) con los tres primeros términos, obtenemos

$$d=\cos(1)-0.5\,\theta\,sen(1)$$



```
PROGRAM redondeoresta
theta=0.1
WRITE (*,*) 'theta          d'
DO i=1,7
    d = cos(1.) - (0.5*theta*sin(1.))
    theta = theta / 10
    WRITE(*,100) theta, d
    100 FORMAT (E11.2,4X, F7.6)
END DO
END PROGRAM redondeoresta
```

Estabilidad de un algoritmo



A fin de mostrar la estabilidad de un algoritmo numérico, calculemos la integral

$$y_n = \int_0^1 \frac{x^n}{(x+5)}\,dx$$

Para n = 0, 1, 2, 3, ... La solución exacta es siempre positiva y su valor disminuye cuando n aumenta.

Usando la siguiente fórmula recurrente:

$$y_n + 5\,y_{n-1} = \frac{1}{n}$$

Que se obtiene de:

$$y_n + 5\,y_{n-1} = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5}\,dx = \int_0^1 \frac{x^{n-1}(x+5)}{x+5}\,dx =$$

$$= \int_0^1 x^{n-1}\,dx = \frac{1}{n}$$

Tenemos que:

$$y_0 = \int_0^1 \frac{1}{x+5}\,dx = \left[\ln(x+5)\right]_0^1 = \ln(6) - \ln(5) \simeq 0.182$$

$$y_1 = 1 - 5y_0 \simeq \quad 0.090$$

$$y_2 = \frac{1}{2} - 5y_1 \simeq \quad 0.050$$

$$y_3 = \frac{1}{3} - 5y_2 \simeq \quad 0.083$$

$$y_4 = \frac{1}{4} - 5y_3 \simeq \; -0.165$$