Computación Científica (cont.)

Yarko Niño C. y Paulo Herrera R.

Departamento de Ingeniería Civil, Universidad de Chile

Semestre Primavera 2011

Clase anterior

- Problemas bien y mal condicionados.
- Algoritmos estables o inestables.
- Precisión requiere utilizar un algoritmo estable para resolver problemas bien condicionados.
- Representación de números reales en computadores como números de coma flotante.
- Sistema de números de coma flotante son discretos y finitos.
- Métodos de redondeo y precisión.

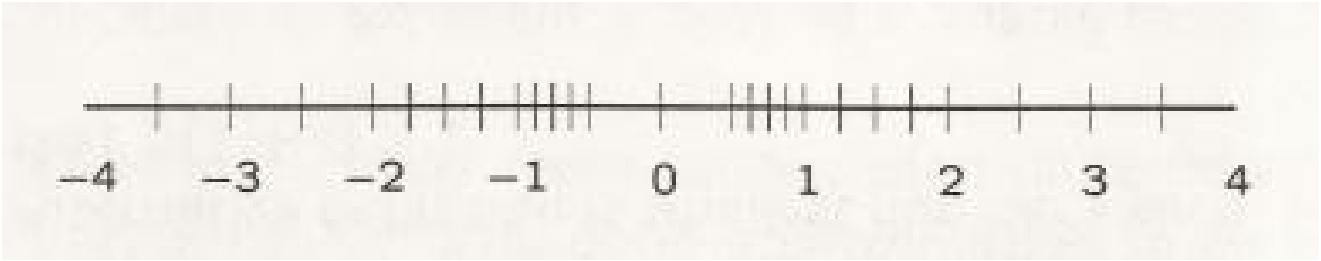
$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_{p-1}}{\beta^{p-1}} \right) \beta^E$$

Precision de la máquina

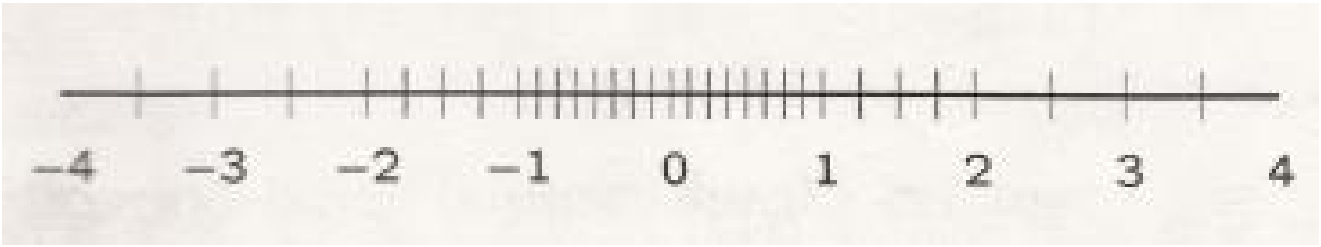
En todos los sistemas prácticos de números de coma flotante,

$$0 < UFL < \epsilon_{mach} < OFL$$

Normalización causa espacio alrededor del zero,



Esa es la motivación para implementar **sistemas subnormalizados**, donde el primer dígito puede ser zero sólo si el exponente $E = -L$. En ese caso,



Números especiales

En algunas situaciones el resultado de una operación es indefinido o no representable en el sistema de número flotante, por lo que los resultados de esas operaciones deben ser tratados de forma especial.

- `Inf`, representa infinito y es el resultado de operaciones como 1/0.
- `NaN` (**N**ot **a** **n**umber), por ejemplo el resultado de `0/0`, `0*Inf`, `Inf/Inf`.

El estándard IEEE especifica el resultado de operaciones que tienen `Inf` o `NaN` como argumentos.

NOTA: Siempre es bueno chequear por la presencia de `NaN` en los resultados de simulaciones.

Aritmética de números de coma flotante

- **Suma o resta**: Traslado de mantissa puede ocasionar pérdida de dígitos en argumento más pequeño.
- **Multiplicación**: Producto de dos mantissas con p dígitos puede tener hasta $2p$ dígitos que no pueden ser almacenados.
- **División**: Cuociente de dos mantissas con p dígitos puede tener más de p dígitos que no pueden ser almacenados. Ejemplo, `1/10` en base 2.

CONCLUSION: Resultado de operaciones aritméticas con números de coma flotante **pueden ser (muy) distintos** a los resultados de las mismas operaciones con números reales.

Ejemplos: Aritmética de números de coma flotante

Tomando $\beta = 10$ y $p = 6$ y los números: $x = 1.92403 \cdot 10^2$ y $y = 6.35782 \cdot 10^{-1}$.

Calcule:

- $x + y =$

$$x + y = 192,403 + 0,635782 = 193,03872$$
$$= 1,93039 \cdot 10^2$$

- $x * y =$

$$x * y = 12,2326364146 \cdot 10^1 = 1,22326 \cdot 10^1$$

En ambos casos se pierden dígitos cuando se redondean los resultados se redondean. Para valores más pequeños de y el efecto del redondeo sería peor.

Overflow y *underflow*

En algunos casos el resultado de una operación puede no ser representable porque el exponente $E > U$ (*overflow*) o $E < L$ (*undeflow*).

En general, *underflow* no es problema porque resultado puede ser aproximado por 0.

Overflow es más difícil porque no existe buena aproximación.

En general, *overflow* puede causar problemas (ej. `NaN`), mientras que *underflow* es silencioso.

Ejemplo: *overflow* y *underflow*

Considere la serie,

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

Esta serie es **divergente** si es evaluada con aritmética con precisión infinita, pero tiene un **valor finito** si es evaluada en un computador. Causas:

- Suma es más grande que el máximo número que se puede representar OFL (*overflow*).
- $1/n$ es más pequeõ que UFL y no agrega nada a la suma.
- $1/n$ es demasiado pequeño en comparación al resultado parcial de la suma,

$$\frac{1}{n} \leq \epsilon_{mach} \sum_{k=1}^{n-1} \frac{1}{k}$$

Reglas de aritmética de números de coma flotante

En la mayoría de los casos resultados computados con aritmética de sistemas de coma flotante son cercanos al redondeo de resultados computados con números reales, es decir,

$$x \quad \texttt{flop} \quad y \quad = \quad \texttt{fl}(x \,\texttt{op}\, y)$$

Sin embargo algunas propiedades de las operaciones con números reales **no son válidas** con números de coma flotante. Por ejemplo, suma y multiplicación son conmutativas **pero no asociativas**.

Ejemplo, si ϵ es un poco menor que ϵ_{mach}, entonces

$$(1 + \epsilon) + \epsilon = 1$$

pero

$$1 + (\epsilon + \epsilon) > 1$$

Cancelación

Resta entre dos númerios de similar magnitud resulta en la **cancelación** de las primeras cifras significativas $\Rightarrow$ resultado tienes menos cifras significativas que los operandos.
Por ejemplo,

$$1,92403 \cdot 10^2 - 1,92275 \cdot 10^2 = 1,28000 \cdot 10^{-1}$$

Eliminación de cifras significativas representa una pérdida de información. Por ejemplo, si ϵ es un poco más pequeño que ϵ_{mach}, entonces

$$(1 - \epsilon) - (1 - \epsilon) =$$

Cancelación

Resta entre dos númerios de similar magnitud resulta en la **cancelación** de las primeras cifras significativas $\Rightarrow$ resultado tienes menos cifras significativas que los operandos.
Por ejemplo,

$$1,92403 \cdot 10^2 - 1,92275 \cdot 10^2 = 1,28000 \cdot 10^{-1}$$

Eliminación de cifras significativas representa una pérdida de información. Por ejemplo, si ϵ es un poco más pequeño que ϵ_{mach}, entonces

$$(1 - \epsilon) - (1 - \epsilon) = 1 - 1 = 0 \neq 2\epsilon$$

Resultado incorrecta debido a pérdida de información durante operaciones intermedias.
RECETA: No calcular pequeñas diferencias como el resultado de la resta entre dos números "grandes".

Rutinas y paquetes computacionales

Debido a las "trampas" de la aritmética de coma flotante, en general, es una mala idea tratar de implementar rutinas matemáticas para uso personal.

Además, existen muchos otros "trucos" para escribir rutinas rápidas que están **más allá del conocimiento de los "meros mortales"**.

Internet está lleno de bibliotecas matemáticas (gratuitas y comerciales) que contienen una gran cantidad de rutinas genéricas, por ej: algebra linear (BLAS, LAPACK, ATLAS, UMFPACK), funciones estadísticas (Intel MKL, NAG), Transformada de Fourier (FFTW), etc. **Netlib** es un buen lugar para buscar rutinas gratuitas.

Mejor aún, hay varios paquetes computacionales (gratuitos y comerciales) que permiten hacer uso de esas bibliotecas en un ambiente gráfico y con un lenguage de programación simple, por ejemplo: MATLAB, Octave, R, Python + Numpy + SciPy, **SciLab**, etc.