Máster Universitario en Ingeniería de Caminos, Canales y Puertos

Introducción al Análisis Numérico

Departamento de Matemática Aplicada

Universidad Granada

Introducción

El *Cálculo o Análisis Numérico* es la rama de las Matemáticas que estudia los *métodos numéricos* de resolución de problemas; es decir, los métodos que permiten obtener una solución aproximada (en ocasiones exacta) del problema considerado, tras realizar un número finito de operaciones lógicas y algebraicas elementales.

El desarrollo del Análisis numérico como disciplina con entidad propia ha ido indisolublemente ligado a la vertiginosa evolución de los ordenadores.

Introducción

El *Cálculo o Análisis Numérico* es la rama de las Matemáticas que estudia los *métodos numéricos* de resolución de problemas; es decir, los métodos que permiten obtener una solución aproximada (en ocasiones exacta) del problema considerado, tras realizar un número finito de operaciones lógicas y algebraicas elementales.

El desarrollo del Análisis numérico como disciplina con entidad propia ha ido indisolublemente ligado a la vertiginosa evolución de los ordenadores.

Introducción

El *Cálculo o Análisis Numérico* es la rama de las Matemáticas que estudia los *métodos numéricos* de resolución de problemas; es decir, los métodos que permiten obtener una solución aproximada (en ocasiones exacta) del problema considerado, tras realizar un número finito de operaciones lógicas y algebraicas elementales.

El desarrollo del Análisis numérico como disciplina con entidad propia ha ido indisolublemente ligado a la vertiginosa evolución de los ordenadores.

Problemas numéricos

Dos grandes grupos:

- De naturaleza discreta: resolución de sistemas de ecuaciones lineales, cálculo de valores y vectores propios, resolución de ecuaciones y sistemas de ecuaciones no lineales, ...
- De naturaleza funcional: interpolación y aproximación de funciones, derivación e integración numéricas, problemas de valor inicial (PVIs) y de contorno (PCs) para ecuaciones diferenciales ordinarias (EDOs) y en Derivadas Parciales (EDPs).

Problemas numéricos

Dos grandes grupos:

- De naturaleza discreta: resolución de sistemas de ecuaciones lineales, cálculo de valores y vectores propios, resolución de ecuaciones y sistemas de ecuaciones no lineales, ...
- De naturaleza funcional: interpolación y aproximación de funciones, derivación e integración numéricas, problemas de valor inicial (PVIs) y de contorno (PCs) para ecuaciones diferenciales ordinarias (EDOs) y en Derivadas Parciales (EDPs).

Problemas numéricos

Dos grandes grupos:

- De naturaleza discreta: resolución de sistemas de ecuaciones lineales, cálculo de valores y vectores propios, resolución de ecuaciones y sistemas de ecuaciones no lineales, ...
- De naturaleza funcional: interpolación y aproximación de funciones, derivación e integración numéricas, problemas de valor inicial (PVIs) y de contorno (PCs) para ecuaciones diferenciales ordinarias (EDOs) y en Derivadas Parciales (EDPs).

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,

hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,

hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,

hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.
Paso 3. Para $i = 1, \ldots, n$,
hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.
PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,
hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,

hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,

hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Noción de Algoritmo

Es un procedimiento que describe, sin ambigüedad posible, una sucesión finita de pasos que hay que realizar en un orden preciso, desde la introducción de datos hasta la obtención de resultados. Incluye pues:

$$\text{ENTRADA} \quad \Rightarrow \text{PROCESO} \Rightarrow \text{SALIDA}.$$

EJEMPLO 1: un algoritmo para el cálculo de la suma de n números reales: $\sum_{i=1}^{n} x_i = x_1 + \cdots + x_n$, puede ser descrito bajo el siguiente pseudocódigo:

ENTRADA Paso 1. Entrar $n, x_1, \ldots, x_n$.

PROCESO Paso 2. Hacer $SUM = 0$.

Paso 3. Para $i = 1, \ldots, n$,

hacer $SUM = SUM + x_i$.

SALIDA Paso 4. Escribir SUM.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar $n, v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar n, $v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar n, $v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar n, $v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar n, $v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar n, $v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejemplos

EJEMPLO 2: cálculo del producto de las componentes no nulas de un vector de $\mathbb{R}^n$; es decir,

$$\prod_{v_i \neq 0} v_i.$$

Paso 1.- Entrar n, $v_1, \ldots, v_n$.

Paso 2.- Hacer $PROD = 1$.

Paso 3.- Para $i = 1 \ldots, n$,

Si $v_i \neq 0$ entonces hacer $PROD = PROD * v_i$.

Paso 4.- Escribir $PROD$.

PARAR.

Ejercicios

1.- Escribir un algoritmo que sume la serie finita $\sum_{i=1}^{n} x_i$ en orden inverso.

2.- Escribir un algoritmo que evalúe el polinomio

$$p(x) = \prod_{i=0}^{n} (x - x_i),$$

para un cierto número real x, conocidos el número de raíces $(n+1)$ y el valor de éstas: $x_0, \ldots, x_n$.

3.- Construir un algoritmo para el estudio de la existencia y el cálculo, en su caso, de las raíces de la ecuación $ax^2 + bx + c = 0$.

Representación de números en el ordenador

Los ordenadores operan en un sistema que no es el decimal (base 10), sino que lo hacen en binario (base 2).

Sin embargo la comunicación con el ordenador se suele hacer siempre en base 10 (decimal); aunque en lenguaje ensamblador se permite trabajar en otro cuya base sea una potencia de 2 (por ejemplo 16).

Para pasar un número de la base decimal a la binaria debemos dividir entre 2 y anotar el resto, y así sucesivamente hasta que el cociente sea cero. Los restos escritos en orden inverso al que se han obtenido proporcionan la expresión del número en binario.

Representación de números en el ordenador

Los ordenadores operan en un sistema que no es el decimal (base 10), sino que lo hacen en binario (base 2).

Sin embargo la comunicación con el ordenador se suele hacer siempre en base 10 (decimal); aunque en lenguaje ensamblador se permite trabajar en otro cuya base sea una potencia de 2 (por ejemplo 16).

Para pasar un número de la base decimal a la binaria debemos dividir entre 2 y anotar el resto, y así sucesivamente hasta que el cociente sea cero. Los restos escritos en orden inverso al que se han obtenido proporcionan la expresión del número en binario.

Representación de números en el ordenador

Los ordenadores operan en un sistema que no es el decimal (base 10), sino que lo hacen en binario (base 2).

Sin embargo la comunicación con el ordenador se suele hacer siempre en base 10 (decimal); aunque en lenguaje ensamblador se permite trabajar en otro cuya base sea una potencia de 2 (por ejemplo 16).

Para pasar un número de la base decimal a la binaria debemos dividir entre 2 y anotar el resto, y así sucesivamente hasta que el cociente sea cero. Los restos escritos en orden inverso al que se han obtenido proporcionan la expresión del número en binario.

Ejemplo: paso de decimal a binario

Para escribir el número 53 del sistema decimal en el sistema binario se realizan las operaciones que se indican a continuación:

$$\begin{array}{lcll} & & & \text{Resto} \\ 53 & = & \underline{26}\times 2 & \rightarrow \quad 1 \\ 26 & = & \underline{13}\times 2 & \rightarrow \quad 0 \\ 13 & = & \underline{6}\times 2 & \rightarrow \quad 1 \\ 6 & = & \underline{3}\times 2 & \rightarrow \quad 0 \\ 3 & = & \underline{1}\times 2 & \rightarrow \quad 1 \\ 1 & = & \underline{\underline{0}}\times 2 & \rightarrow \quad 1 \end{array}$$

Los restos dan 110101 como expresión binaria de 53, lo que podemos notar como $53_{10} = 110101_2$.

ATENCIÓN: números que tienen una representación finita en el sistema decimal pueden tener una representación infinita en binario.

Ejemplo: paso de binario a decimal y viceversa

Por ejemplo el número

$$0.7_{10} = 0.101100110011100\ldots_2$$

tiene que ser aproximado redondeándolo por defecto o por exceso a un número cuya representación sea admisible por el ordenador (un *número máquina*: cadena finita de ceros y unos).

Sin embargo, el paso de la representación binario a decimal es muy fácil,

$$\begin{aligned} 10110.01_2 = & \\ & 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 \\ & + 0\times 2^{-1} + 1\times 2^{-2} \\ & = 22.25_{10} \end{aligned}$$

Ejemplo: paso de binario a decimal y viceversa

Por ejemplo el número

$$0.7_{10} = 0.101100110011100\ldots_2$$

tiene que ser aproximado redondeándolo por defecto o por exceso a un número cuya representación sea admisible por el ordenador (un *número máquina*: cadena finita de ceros y unos).

Sin embargo, el paso de la representación binario a decimal es muy fácil,

$$\begin{aligned} 10110.01_2 = & \\ & 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ & + 0 \times 2^{-1} + 1 \times 2^{-2} \\ & = 22.25_{10} \end{aligned}$$

Representación en coma flotante

La notación científica (o en coma flotante) para representar los números en el ordenador consta de *signo*, *mantisa* y *exponente*; es decir, un número diferente de cero queda representado de la forma

$$\pm m.b^e,$$

donde

- la mantisa m es un número comprendido entre 0 y 1 cuyo primer dígito es no nulo;
- b es la base (10 cuando estamos en el sistema decimal), y
- e es el exponente.

Representación de números en el ordenador

Para representar en binario el signo, la mantisa y el exponente de un número se usa una cadena de *bits*.

Un bit es un elemento físico que admite dos estados que podemos hacer corresponder pues al 0 y al 1.

Matemáticamente, la cadena de bits que se usa para representar a un número equivale a una sucesión finita de ceros y unos.

El primero de ellos se emplea para representar el signo del número, la mayor parte de los siguientes para la mantisa y los restantes para el exponente.

Una situación normal es disponer de 64 bits, de los que uno se dedica al signo, 52 a la mantisa y 11 al exponente.

Representación de números en el ordenador

Para representar en binario el signo, la mantisa y el exponente de un número se usa una cadena de *bits*.

Un bit es un elemento físico que admite dos estados que podemos hacer corresponder pues al 0 y al 1.

Matemáticamente, la cadena de bits que se usa para representar a un número equivale a una sucesión finita de ceros y unos.

El primero de ellos se emplea para representar el signo del número, la mayor parte de los siguientes para la mantisa y los restantes para el exponente.

Una situación normal es disponer de 64 bits, de los que uno se dedica al signo, 52 a la mantisa y 11 al exponente.

Representación de números en el ordenador

Para representar en binario el signo, la mantisa y el exponente de un número se usa una cadena de *bits*.

Un bit es un elemento físico que admite dos estados que podemos hacer corresponder pues al 0 y al 1.

Matemáticamente, la cadena de bits que se usa para representar a un número equivale a una sucesión finita de ceros y unos.

El primero de ellos se emplea para representar el signo del número, la mayor parte de los siguientes para la mantisa y los restantes para el exponente.

Una situación normal es disponer de 64 bits, de los que uno se dedica al signo, 52 a la mantisa y 11 al exponente.

Representación de números en el ordenador

Para representar en binario el signo, la mantisa y el exponente de un número se usa una cadena de *bits*.

Un bit es un elemento físico que admite dos estados que podemos hacer corresponder pues al 0 y al 1.

Matemáticamente, la cadena de bits que se usa para representar a un número equivale a una sucesión finita de ceros y unos.

El primero de ellos se emplea para representar el signo del número, la mayor parte de los siguientes para la mantisa y los restantes para el exponente.

Una situación normal es disponer de 64 bits, de los que uno se dedica al signo, 52 a la mantisa y 11 al exponente.

Representación de números en el ordenador

Para representar en binario el signo, la mantisa y el exponente de un número se usa una cadena de *bits*.

Un bit es un elemento físico que admite dos estados que podemos hacer corresponder pues al 0 y al 1.

Matemáticamente, la cadena de bits que se usa para representar a un número equivale a una sucesión finita de ceros y unos.

El primero de ellos se emplea para representar el signo del número, la mayor parte de los siguientes para la mantisa y los restantes para el exponente.

Una situación normal es disponer de 64 bits, de los que uno se dedica al signo, 52 a la mantisa y 11 al exponente.

Representación de números en el ordenador

Los exponentes posibles irán pues desde el 0 hasta el $11111111111_2 = 2047_{10}$; de tal forma que, si se contempla un desplazamiento automático de 1024, tendríamos como posibles exponentes los valores enteros comprendidos entre -1024 y 1023, y el rango de números positivos con que podríamos trabajar va, aproximadamente, de

$$2^{-1024} \simeq 5.5626846462680035 * 10^{-309}$$

hasta

$$2^{1023} \simeq 8.9884656743115795 * 10^{307}$$

Si tras una operación se obtiene un valor superior a 2^{1023} se produce un desbordamiento (*overflow*). Un valor inferior a 2^{-1024} es tomado como 0 (*underflow*)

Representación de números en el ordenador

Los exponentes posibles irán pues desde el 0 hasta el $11111111111_2 = 2047_{10}$; de tal forma que, si se contempla un desplazamiento automático de 1024, tendríamos como posibles exponentes los valores enteros comprendidos entre -1024 y 1023, y el rango de números positivos con que podríamos trabajar va, aproximadamente, de

$$2^{-1024} \simeq 5.562684646268003 5 * 10^{-309}$$

hasta

$$2^{1023} \simeq 8.988465674311579 5 * 10^{307}$$

Si tras una operación se obtiene un valor superior a 2^{1023} se produce un desbordamiento (*overflow*). Un valor inferior a 2^{-1024} es tomado como 0 (*underflow*)

Representación de números en el ordenador

Los exponentes posibles irán pues desde el 0 hasta el $11111111111_2 = 2047_{10}$; de tal forma que, si se contempla un desplazamiento automático de 1024, tendríamos como posibles exponentes los valores enteros comprendidos entre -1024 y 1023, y el rango de números positivos con que podríamos trabajar va, aproximadamente, de

$$2^{-1024} \simeq 5.5626846462680035 * 10^{-309}$$

hasta

$$2^{1023} \simeq 8.9884656743115795 * 10^{307}$$

Si tras una operación se obtiene un valor superior a 2^{1023} se produce un desbordamiento (*overflow*). Un valor inferior a 2^{-1024} es tomado como 0 (*underflow*)

Representación de números en el ordenador

Los exponentes posibles irán pues desde el 0 hasta el $11111111111_2 = 2047_{10}$; de tal forma que, si se contempla un desplazamiento automático de 1024, tendríamos como posibles exponentes los valores enteros comprendidos entre -1024 y 1023, y el rango de números positivos con que podríamos trabajar va, aproximadamente, de

$$2^{-1024} \simeq 5.562684646268003 5 * 10^{-309}$$

hasta

$$2^{1023} \simeq 8.988465674311579 5 * 10^{307}$$

Si tras una operación se obtiene un valor superior a 2^{1023} se produce un desbordamiento (*overflow*). Un valor inferior a 2^{-1024} es tomado como 0 (*underflow*)

Precisión de los números en coma flotante

La precisión habitual es de $2^{-52} \simeq 2.2 * 10^{-16}$; es decir, de unas 16 cifras decimales. Sólo un conjunto finito de números puede ser representado en el ordenador de forma exacta. El siguiente a $0.1_2 = 2_{10}^{-1} = 0.5_{10}$ es

$$0.1\overbrace{0\cdots0}^{50}1_2 = 2^{-1} + 2^{-52} = 0.5000000000000022044_{10}$$

Cualquier otro número intermedio entre estos dos tiene que ser aproximado por uno de ellos (redondeo). Si se aproxima por el más cercano se dice que el redondeo es *simétrico*; si se cortan todos los dígitos de la representación que sobrepasan la dimensión de la mantisa el redondeo es por *corte*. Por ej., en binario, con 52 dígitos, el redondeo simétrico de 0.50000000000000022044 es $2^{-1} + 2^{-52}$, mientras que por corte sería redondeado a $2^{-1} = 0.5$.

Precisión de los números en coma flotante

La precisión habitual es de $2^{-52} \simeq 2.2 * 10^{-16}$; es decir, de unas 16 cifras decimales. Sólo un conjunto finito de números puede ser representado en el ordenador de forma exacta. El siguiente a $0.1_2 = 2^{-1}_{10} = 0.5_{10}$ es

$$0.1\overbrace{0\cdots0}^{50}1_2 = 2^{-1} + 2^{-52}$$
$$= 0.5000000000000022044_{10}$$

Cualquier otro número intermedio entre estos dos tiene que ser aproximado por uno de ellos (redondeo). Si se aproxima por el más cercano se dice que el redondeo es *simétrico*; si se cortan todos los dígitos de la representación que sobrepasan la dimensión de la mantisa el redondeo es por *corte*. Por ej., en binario, con 52 dígitos, el redondeo simétrico de 0.50000000000000022044 es $2^{-1} + 2^{-52}$, mientras que por corte sería redondeado a $2^{-1} = 0.5$.

Precisión de los números en coma flotante

La precisión habitual es de $2^{-52} \simeq 2.2 * 10^{-16}$; es decir, de unas 16 cifras decimales. Sólo un conjunto finito de números puede ser representado en el ordenador de forma exacta. El siguiente a $0.1_2 = 2^{-1}_{10} = 0.5_{10}$ es

$$0.1\overbrace{0\cdots0}^{50}1_2 = 2^{-1} + 2^{-52}$$
$$= 0.5000000000000022044_{10}$$

Cualquier otro número intermedio entre estos dos tiene que ser aproximado por uno de ellos (redondeo). Si se aproxima por el más cercano se dice que el redondeo es *simétrico*; si se cortan todos los dígitos de la representación que sobrepasan la dimensión de la mantisa el redondeo es por *corte*. Por ej., en binario, con 52 dígitos, el redondeo simétrico de 0.5000000000000022044 es $2^{-1} + 2^{-52}$, mientras que por corte sería redondeado a $2^{-1} = 0.5$.

Precisión de los números en coma flotante

La precisión habitual es de $2^{-52} \simeq 2.2 * 10^{-16}$; es decir, de unas 16 cifras decimales. Sólo un conjunto finito de números puede ser representado en el ordenador de forma exacta. El siguiente a $0.1_2 = 2^{-1}_{10} = 0.5_{10}$ es

$$0.1\overbrace{0\cdots0}^{50}1_2 = 2^{-1} + 2^{-52}$$
$$= 0.5000000000000022044_{10}$$

Cualquier otro número intermedio entre estos dos tiene que ser aproximado por uno de ellos (redondeo). Si se aproxima por el más cercano se dice que el redondeo es *simétrico*; si se cortan todos los dígitos de la representación que sobrepasan la dimensión de la mantisa el redondeo es por *corte*. Por ej., en binario, con 52 dígitos, el redondeo simétrico de 0.5000000000000022044 es $2^{-1} + 2^{-52}$, mientras que por corte sería redondeado a $2^{-1} = 0.5$.

Precisión de los números en coma flotante

El número máximo de cifras significativas que soporta el ordenador en la representación interna de un número real se denomina *precisión*. Si la precisión en decimal es $k \in \mathbb{N}$; entonces todo número real se puede representar de la forma

$$0.d_1 \ldots d_k * 10^n.$$

Supuesto que el redondeo sea simétrico, ese mismo número máquina representa a todos los comprendidos entre

$$0.d_1 \ldots d_k * 10^n - 5 * 10^{n-k-1}$$

y

$$0.d_1 \ldots d_k * 10^n + 5 * 10^{n-k-1}$$

Precisión de los números en coma flotante

El número máximo de cifras significativas que soporta el ordenador en la representación interna de un número real se denomina *precisión*. Si la precisión en decimal es $k \in \mathbb{N}$; entonces todo número real se puede representar de la forma

$$0.d_1 \ldots d_k * 10^n.$$

Supuesto que el redondeo sea simétrico, ese mismo número máquina representa a todos los comprendidos entre

$$0.d_1 \ldots d_k * 10^n - 5 * 10^{n-k-1}$$

y

$$0.d_1 \ldots d_k * 10^n + 5 * 10^{n-k-1}$$

Precisión de los números en coma flotante

El número máximo de cifras significativas que soporta el ordenador en la representación interna de un número real se denomina *precisión*. Si la precisión en decimal es $k \in \mathbb{N}$; entonces todo número real se puede representar de la forma

$$0.d_1 \ldots d_k * 10^n.$$

Supuesto que el redondeo sea simétrico, ese mismo número máquina representa a todos los comprendidos entre

$$0.d_1 \ldots d_k * 10^n - 5 * 10^{n-k-1}$$

y

$$0.d_1 \ldots d_k * 10^n + 5 * 10^{n-k-1}$$

Ejemplo

Por ej., si la precisión es $k = 9$ (en decimal) el número $0.123456789 * 10^2$ representa indistintamente a todos los números del intervalo

$$\left[0.1234567885 * 10^2, 0.1234567895 * 10^2\right[$$

La diferencia entre cualquiera de estos números y

$$0.123456789 * 10^2$$

es el error de redondeo correspondiente. Estos errores de redondeo se producen tanto en la entrada y salida de datos, como en todas las operaciones intermedias.

Ejemplos

Si la precisión fuese de 5 cifras y hubiese que multiplicar los números $a = 0.42183$ y $b = 0.38124$; entonces el valor $a \times b = 0.1608184692$ tiene que aproximarse por 0.16082 si redondea de forma simétrica, y por 0.16081 si se redondea por corte.

Reorganizando la memoria del ordenador se puede aumentar la precisión en los redondeos.

De hecho, muy pronto los ordenadores pudieron trabajar en doble precisión (64 bits); aunque otros programas como *Mathematica* van mucho más allá y son capaces de aumentar indefinidamente la precisión y trabajar en simbólico sin más límite que la memoria del ordenador.

Ejemplos

Si la precisión fuese de 5 cifras y hubiese que multiplicar los números $a = 0.42183$ y $b = 0.38124$; entonces el valor $a \times b = 0.1608184692$ tiene que aproximarse por 0.16082 si redondea de forma simétrica, y por 0.16081 si se redondea por corte.

Reorganizando la memoria del ordenador se puede aumentar la precisión en los redondeos.

De hecho, muy pronto los ordenadores pudieron trabajar en doble precisión (64 bits); aunque otros programas como *Mathematica* van mucho más allá y son capaces de aumentar indefinidamente la precisión y trabajar en simbólico sin más límite que la memoria del ordenador.

Ejemplos

Si la precisión fuese de 5 cifras y hubiese que multiplicar los números $a = 0.42183$ y $b = 0.38124$; entonces el valor $a \times b = 0.1608184692$ tiene que aproximarse por 0.16082 si redondea de forma simétrica, y por 0.16081 si se redondea por corte.

Reorganizando la memoria del ordenador se puede aumentar la precisión en los redondeos.

De hecho, muy pronto los ordenadores pudieron trabajar en doble precisión (64 bits); aunque otros programas como *Mathematica* van mucho más allá y son capaces de aumentar indefinidamente la precisión y trabajar en simbólico sin más límite que la memoria del ordenador.

Error absoluto y relativo

Si x^* es una aproximación del número real x, se llama *error absoluto* a $|a_x| \equiv |x - x^*|$ y *error relativo* a $|r_x| \equiv \dfrac{|x - x^*|}{|x|}$,o bien $|\widetilde{r}_x| \equiv \dfrac{|x - x^*|}{|x^*|}$ siempre que sean $x \neq 0 \neq x^*$.

El error relativo tiene que ser muy próximo a cero para que la aproximación sea buena; por ejemplo, si el peso exacto debía ser 500 gramos, se habría cometido un error muy importante en la pesada (el error relativo sería ((300)/(500))=0.6). Sin embargo, el error relativo sería 0.00003 y el error absoluto despreciable si el peso exacto fuese 10.000 Kg.

Por ejemplo, $\pi = 3.14159265\ldots$ se representa, con una precisión $k = 5$, mediante el número $0.34516 \times 10^1 = 3.1416$. Por tanto, el error absoluto es $|3.14159265\ldots - 3.1416| = 0.00000735\ldots < 0.8 \times 10^{-5}$.

Error absoluto y relativo

Si x^* es una aproximación del número real x, se llama *error absoluto* a $|a_x| \equiv |x - x^*|$ y *error relativo* a $|r_x| \equiv \dfrac{|x - x^*|}{|x|}$,o bien $\left|\widetilde{r}_x\right| \equiv \dfrac{|x - x^*|}{|x^*|}$ siempre que sean $x \neq 0 \neq x^*$.

El error relativo tiene que ser muy próximo a cero para que la aproximación sea buena; por ejemplo, si el peso exacto debía ser 500 gramos, se habría cometido un error muy importante en la pesada (el error relativo sería ((300)/(500))=0.6). Sin embargo, el error relativo sería 0.00003 y el error absoluto despreciable si el peso exacto fuese 10.000 Kg.

Por ejemplo, $\pi = 3.14159265\ldots$ se representa, con una precisión $k = 5$, mediante el número $0.34516 \times 10^1 = 3.1416$. Por tanto, el error absoluto es $|3.14159265\ldots - 3.1416| = 0.00000735\ldots < 0.8 \times 10^{-5}$.

Error absoluto y relativo

Si x^* es una aproximación del número real x, se llama *error absoluto* a $|a_x| \equiv |x - x^*|$ y *error relativo* a $|r_x| \equiv \dfrac{|x - x^*|}{|x|}$,o bien $\left|\widetilde{r}_x\right| \equiv \dfrac{|x - x^*|}{|x^*|}$ siempre que sean $x \neq 0 \neq x^*$.

El error relativo tiene que ser muy próximo a cero para que la aproximación sea buena; por ejemplo, si el peso exacto debía ser 500 gramos, se habría cometido un error muy importante en la pesada (el error relativo sería ((300)/(500))=0.6). Sin embargo, el error relativo sería 0.00003 y el error absoluto despreciable si el peso exacto fuese 10.000 Kg.

Por ejemplo, $\pi = 3.14159265\ldots$ se representa, con una precisión $k = 5$, mediante el número $0.34516 \times 10^1 = 3.1416$. Por tanto, el error absoluto es $|3.14159265\ldots - 3.1416| = 0.00000735\ldots < 0.8 \times 10^{-5}$.

Error absoluto y relativo

Si x^* es una aproximación del número real x, se llama *error absoluto* a $|a_x| \equiv |x - x^*|$ y *error relativo* a $|r_x| \equiv \dfrac{|x - x^*|}{|x|}$,o bien $\left|\widetilde{r}_x\right| \equiv \dfrac{|x - x^*|}{|x^*|}$ siempre que sean $x \neq 0 \neq x^*$.

El error relativo tiene que ser muy próximo a cero para que la aproximación sea buena; por ejemplo, si el peso exacto debía ser 500 gramos, se habría cometido un error muy importante en la pesada (el error relativo sería ((300)/(500))=0.6). Sin embargo, el error relativo sería 0.00003 y el error absoluto despreciable si el peso exacto fuese 10.000 Kg.

Por ejemplo, $\pi = 3.14159265\ldots$ se representa, con una precisión $k = 5$, mediante el número $0.34516 \times 10^1 = 3.1416$. Por tanto, el error absoluto es $|3.14159265\ldots - 3.1416|$ $= 0.00000735\ldots < 0.8 \times 10^{-5}$.

Error absoluto y relativo

En la práctica no se suelen conocer los errores absoluto y relativo, sino cotas de los mismos, δ_x y ε_x, de tal forma que se verifican las desigualdades

$$|a_x| \leq \delta_x \quad , \qquad |r_x| \simeq \left|\widetilde{r}_x\right| \leq \varepsilon_x .$$

También se suele notar

$$x = x^* \pm \delta_x \quad , \quad x = x^* (1 \pm \varepsilon_x)$$

o a la inversa

$$x^* = x \pm \delta_x \quad , \quad x^* = x (1 \pm \varepsilon_x) .$$

Error absoluto y relativo

En la práctica no se suelen conocer los errores absoluto y relativo, sino cotas de los mismos, δ_x y ε_x, de tal forma que se verifican las desigualdades

$$|a_x| \leq \delta_x \quad , \qquad |r_x| \simeq \left|\widetilde{r}_x\right| \leq \varepsilon_x .$$

También se suele notar

$$x = x^* \pm \delta_x \quad , \quad x = x^* \left(1 \pm \varepsilon_x\right)$$

o a la inversa

$$x^* = x \pm \delta_x \quad , \quad x^* = x \left(1 \pm \varepsilon_x\right) .$$

Error absoluto y relativo

En la práctica no se suelen conocer los errores absoluto y relativo, sino cotas de los mismos, δ_x y ε_x, de tal forma que se verifican las desigualdades

$$|a_x| \leq \delta_x \quad , \qquad |r_x| \simeq \left|\widetilde{r}_x\right| \leq \varepsilon_x .$$

También se suele notar

$$x = x^* \pm \delta_x \quad , \quad x = x^* \left(1 \pm \varepsilon_x\right)$$

o a la inversa

$$x^* = x \pm \delta_x \quad , \quad x^* = x \left(1 \pm \varepsilon_x\right) .$$

Propagación del error relativo en las operaciones elementales

Supuesto que se conozcan cotas para los errores absoluto y relativo de unas cantidades x, y, ... es interesante conocer una cota del resultado obtenido al efectuar una operación elemental $(+, -, *, /)$ con las mismas.

Es inmediato comprobar que

$$\delta_{x+y} = \delta_x + \delta_y$$
$$\delta_{x-y} = \delta_x + \delta_y$$

ya que, si $x = x^* + a_x$ e $y = y^* + a_y$, entonces

$$|(x \pm y) - (x^* \pm y^*)| \leq \delta_x + \delta_y.$$

Propagación del error relativo en las operaciones elementales

Supuesto que se conozcan cotas para los errores absoluto y relativo de unas cantidades x, y, ... es interesante conocer una cota del resultado obtenido al efectuar una operación elemental $(+, -, *, /)$ con las mismas.

Es inmediato comprobar que

$$\delta_{x+y} = \delta_x + \delta_y$$
$$\delta_{x-y} = \delta_x + \delta_y$$

ya que, si $x = x^* + a_x$ e $y = y^* + a_y$, entonces

$$|(x \pm y) - (x^* \pm y^*)| \leq \delta_x + \delta_y.$$

Propagación del error relativo en las operaciones elementales

De $x^* = x\,(1 - r_x)$ e $y^* = y\,(1 - r_y)$ se deduce que $(x \pm y) - (x^* \pm y^*) = x\,r_x \pm y\,r_y$; por tanto, supuesto que $x \pm y \neq 0$, la división conduce a

$$r_{x\pm y} = \frac{x}{x \pm y}\,r_x \pm \frac{y}{x \pm y}\,r_y,$$

de tal forma que, si tanto x como $\pm y$ son de igual signo, se tiene

$$\left|r_{x\pm y}\right| \leq |r_x| + \left|r_y\right|,$$

o equivalentemente, $\varepsilon_{x\pm y} \leq \varepsilon_x + \varepsilon_y$. Pero si x e $\pm y$ tienen signos opuestos, el error relativo se puede *disparar*; es el denominado efecto de cancelación.

Propagación del error relativo en las operaciones elementales

De $x^* = x\,(1 - r_x)$ e $y^* = y\,(1 - r_y)$ se deduce que $(x \pm y) - (x^* \pm y^*) = x\,r_x \pm y\,r_y$; por tanto, supuesto que $x \pm y \neq 0$, la división conduce a

$$r_{x\pm y} = \frac{x}{x \pm y}\,r_x \pm \frac{y}{x \pm y}\,r_y,$$

de tal forma que, si tanto x como $\pm y$ son de igual signo, se tiene

$$\left|r_{x\pm y}\right| \leq |r_x| + \left|r_y\right|,$$

o equivalentemente, $\varepsilon_{x\pm y} \leq \varepsilon_x + \varepsilon_y$. Pero si x e $\pm y$ tienen signos opuestos, el error relativo se puede *disparar*; es el denominado efecto de cancelación.

Propagación del error relativo en las operaciones elementales

De $x^* = x\,(1 - r_x)$ e $y^* = y\,(1 - r_y)$ se deduce que $(x \pm y) - (x^* \pm y^*) = x\,r_x \pm y\,r_y$; por tanto, supuesto que $x \pm y \neq 0$, la división conduce a

$$r_{x\pm y} = \frac{x}{x \pm y}\,r_x \pm \frac{y}{x \pm y}\,r_y,$$

de tal forma que, si tanto x como $\pm y$ son de igual signo, se tiene

$$\left|r_{x\pm y}\right| \leq |r_x| + \left|r_y\right|,$$

o equivalentemente, $\varepsilon_{x\pm y} \leq \varepsilon_x + \varepsilon_y$. Pero si x e $\pm y$ tienen signos opuestos, el error relativo se puede *disparar*; es el denominado efecto de cancelación.

Propagación del error relativo en las operaciones elementales

En cuanto al producto y al cociente, supondremos que el producto de r_x por r_y y sus cuadrados son cantidades despreciables. Para la multiplicación se tiene que

$$\begin{aligned} & x\left(1-r_x\right) * y\left(1-r_y\right) \\ & = x * y\left(1-r_x-r_y+r_x r_y\right) \\ & \simeq x * y\left(1-r_x-r_y\right) \end{aligned}$$

por tanto, el error relativo en la multiplicación está acotado por

$$\varepsilon_{xy} \simeq \varepsilon_x + \varepsilon_y$$

De forma análoga, para el cociente se llega a la misma cota:

$$\varepsilon_{x/y} \simeq \varepsilon_x + \varepsilon_y$$

Propagación del error relativo en las operaciones elementales

En cuanto al producto y al cociente, supondremos que el producto de r_x por r_y y sus cuadrados son cantidades despreciables. Para la multiplicación se tiene que

$$\begin{aligned} & x\left(1-r_x\right) * y\left(1-r_y\right) \\ & = x * y\left(1-r_x-r_y+r_x r_y\right) \\ & \simeq x * y\left(1-r_x-r_y\right) \end{aligned}$$

por tanto, el error relativo en la multiplicación está acotado por

$$\varepsilon_{xy} \simeq \varepsilon_x + \varepsilon_y$$

De forma análoga, para el cociente se llega a la misma cota:

$$\varepsilon_{x/y} \simeq \varepsilon_x + \varepsilon_y$$

Propagación del error relativo en las operaciones elementales

En cuanto al producto y al cociente, supondremos que el producto de r_x por r_y y sus cuadrados son cantidades despreciables. Para la multiplicación se tiene que

$$\begin{aligned} & x\left(1-r_x\right) * y\left(1-r_y\right) \\ & = x * y\left(1-r_x-r_y+r_x r_y\right) \\ & \simeq x * y\left(1-r_x-r_y\right) \end{aligned}$$

por tanto, el error relativo en la multiplicación está acotado por

$$\varepsilon_{xy} \simeq \varepsilon_x + \varepsilon_y$$

De forma análoga, para el cociente se llega a la misma cota:

$\varepsilon_{x/y} \simeq \varepsilon_x + \varepsilon_y$

Error por cancelación

Por ejemplo, como $f(a+h) = f(a) + hf'(a) + \frac{1}{2}h^2 f''(\xi)$ entonces al aproximar la derivada de f en a por

$$f'(a) \simeq \frac{f(a+h) - f(a)}{h},$$

supuesto que no se cometan errores al evaluar la función f en $a+h$ y en a, ni en la resta, ni en la división, el error cometido es

$$f'(a) - \frac{f(a+h) - f(a)}{h} = -\frac{h}{2} f''(\xi)$$

y corresponde al truncamiento efectuado al despreciar el último término del desarrollo de Taylor.

Pero cuando h es pequeño las cantidades $f(a+h)$ y $f(a)$ son muy similares pudiendo llegar a darse el *error por cancelación*.

Error por cancelación

Por ejemplo, como $f(a+h) = f(a) + hf'(a) + \frac{1}{2}h^2 f''(\xi)$ entonces al aproximar la derivada de f en a por

$$f'(a) \simeq \frac{f(a+h) - f(a)}{h},$$

supuesto que no se cometan errores al evaluar la función f en $a+h$ y en a, ni en la resta, ni en la división, el error cometido es

$$f'(a) - \frac{f(a+h) - f(a)}{h} = -\frac{h}{2} f''(\xi)$$

y corresponde al truncamiento efectuado al despreciar el último término del desarrollo de Taylor.

Pero cuando h es pequeño las cantidades $f(a+h)$ y $f(a)$ son muy similares pudiendo llegar a darse el *error por cancelación*.

Error por cancelación

Para algunos problemas podemos organizar los cálculos de forma que se evite el error por cancelación. Por ejemplo, para todo $x \geq 0$, la expresión $x\left(\sqrt{x+1}-\sqrt{x}\right)$ es equivalente a $\dfrac{x}{\sqrt{x+1}+\sqrt{x}}$ cuyo valor es próximo a $\frac{1}{2}\sqrt{x}$; sin embargo, para x suficientemente grande la primera expresión da cero (operando en coma flotante).

Ejerc. 4.- Calcular el valor de $f(x)=\sqrt{x^2+1}-1$ y de $g(x)=\dfrac{x^2}{\sqrt{x^2+1}+1}$, con un ordenador o una calculadora, para una sucesión de valores $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ Aunque $f=g$, los resultados pueden resultar diferentes. ¿Cuáles de los resultados son de fiar y cuáles no?

Error por cancelación

Para algunos problemas podemos organizar los cálculos de forma que se evite el error por cancelación. Por ejemplo, para todo $x \geq 0$, la expresión $x\left(\sqrt{x+1}-\sqrt{x}\right)$ es equivalente a $\dfrac{x}{\sqrt{x+1}+\sqrt{x}}$ cuyo valor es próximo a $\frac{1}{2}\sqrt{x}$; sin embargo, para x suficientemente grande la primera expresión da cero (operando en coma flotante).

Ejerc. 4.- Calcular el valor de $f(x)=\sqrt{x^2+1}-1$ y de $g(x)=\dfrac{x^2}{\sqrt{x^2+1}+1}$, con un ordenador o una calculadora, para una sucesión de valores $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ Aunque $f=g$, los resultados pueden resultar diferentes. ¿Cuáles de los resultados son de fiar y cuáles no?

Error por cancelación

Para algunos problemas podemos organizar los cálculos de forma que se evite el error por cancelación. Por ejemplo, para todo $x \geq 0$, la expresión $x\left(\sqrt{x+1}-\sqrt{x}\right)$ es equivalente a $\dfrac{x}{\sqrt{x+1}+\sqrt{x}}$ cuyo valor es próximo a $\frac{1}{2}\sqrt{x}$; sin embargo, para x suficientemente grande la primera expresión da cero (operando en coma flotante).

Ejerc. 4.- Calcular el valor de $f(x)=\sqrt{x^2+1}-1$ y de $g(x)=\dfrac{x^2}{\sqrt{x^2+1}+1}$, con un ordenador o una calculadora, para una sucesión de valores $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ Aunque $f=g$, los resultados pueden resultar diferentes. ¿Cuáles de los resultados son de fiar y cuáles no?

Otros errores

- errores inherentes al planteamiento del problema (al imponer, por ejemplo, hipótesis que simplifiquen un modelo complejo)
- errores inherentes a la obtención de datos iniciales (por ejemplo, errores de medida de datos físicos).

ERRORES DE DISCRETIZACIÓN O DE TRUNCATURA: originados al aproximar un proceso infinito por un proceso finito. Un método numérico se dice *convergente* si es capaz de proporcionar una solución aproximada del problema considerado para cualquier grado prefijado de tolerancia; es decir, si x_n es la solución obtenida en n pasos y s es la solución exacta, dado $\varepsilon > 0$, existe $n_0 \in \mathbb{N}$ tal que

$$|x_{n_0} - s| < \varepsilon.$$

Error por truncamiento

Cuando en un problema se dispone de una sucesión de valores, $\{x_n\}_{n\geq 1}$, convergente a la solución del mismo y se usa un método numérico que emplee un valor x_n como aproximación de la solución, el error cometido se llama *error de truncamiento* o *truncatura*.

Por ejemplo, si consideramos la aproximación

$$e^x \simeq 1 + x + \frac{1}{2}x^2,$$

cometemos el error de truncatura $\frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots$

Análogamente, a veces se tienen fórmulas procedentes de despreciar el último término de un desarrollo finito de Taylor, y el error cometido es también por truncamiento.

Error por truncamiento

Cuando en un problema se dispone de una sucesión de valores, $\{x_n\}_{n\geq 1}$, convergente a la solución del mismo y se usa un método numérico que emplee un valor x_n como aproximación de la solución, el error cometido se llama *error de truncamiento* o *truncatura*.

Por ejemplo, si consideramos la aproximación

$$e^x \simeq 1 + x + \frac{1}{2}x^2,$$

cometemos el error de truncatura $\frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots$

Análogamente, a veces se tienen fórmulas procedentes de despreciar el último término de un desarrollo finito de Taylor, y el error cometido es también por truncamiento.

Error por truncamiento

Cuando en un problema se dispone de una sucesión de valores, $\{x_n\}_{n\geq 1}$, convergente a la solución del mismo y se usa un método numérico que emplee un valor x_n como aproximación de la solución, el error cometido se llama *error de truncamiento* o *truncatura*.

Por ejemplo, si consideramos la aproximación

$$e^x \simeq 1 + x + \frac{1}{2}x^2,$$

cometemos el error de truncatura $\frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots$

Análogamente, a veces se tienen fórmulas procedentes de despreciar el último término de un desarrollo finito de Taylor, y el error cometido es también por truncamiento.

Otro ejemplo de desarrollo de Taylor

El método numérico dado por la inducción:

$$(P_n)\begin{cases} x_0 = f(a), \\ \\ x_k = x_{k-1} + \dfrac{f^{(k)}(a)}{k!}(x-a)^k,\ 1 \leq k \leq n, \end{cases}$$

es una discretización del cálculo de $f(x)$ mediante el desarrollo limitado de Taylor de orden n de f centrado en a. Se sabe que, dado $\varepsilon > 0$, existen n_0 tal que, bajo ciertas condiciones,

$$|f(x) - x_{n_0}| = |\frac{f^{n_0+1}(\alpha)}{(n_0+1)!}(x-a)^{n_0+1}| < \varepsilon,\ \alpha \in]a, x[$$

de donde, se tiene que el método numérico (P_n), es convergente.

Condicionamiento

Nos indica la sensibilidad de la solución ante pequeños cambios relativos de los datos. Un problema numérico se dice *bien condicionado* (*mal condicionado*) si pequeños cambios relativos en los datos de entrada producen pequeños cambios (grandes cambios) relativos en la solución.
Ejemplo de método mal condicionado: la obtención de las raíces de una ecuación polinómica de segundo grado mediante fórmulas usuales. La mayor raíz de

$$0.1x^2 - 20x + 20 = 0$$

es $x = 198.9949$ y, realizando un pequeño aumento de los coeficientes de 0.01; para la ecuación

$$0.11x^2 - 19.99x + 20.01 = 0$$

se obtiene $x = 180.7206$.

Proceso de evaluación de funciones

En lo que se refiere a la evaluación en un punto x de la función $y = f(x)$, que suponemos derivable con continuidad, del desarrollo de Taylor se deduce que

$$a_y \simeq f'(x)\ a_x$$
$$\delta_y \simeq \left|f'(x)\right|\ \delta_x$$

De forma análoga, si Y es una función de varias variables, digamos

$$Y = f\left(\overbrace{x, y, z, \ldots}^{\mathbf{X}}\right)$$

una cota del error absoluto sería

$$\delta_Y \simeq \left|f_x(\mathbf{X})\right|\ \delta_x + \left|f_y(\mathbf{X})\right|\ \delta_y + \left|f_z(\mathbf{X})\right|\ \delta_z + \cdots$$

Proceso inestable

Es aquel en el que los pequeños errores que se producen en una de sus etapas se agrandan en etapas posteriores y degradan seriamente la exactitud del cálculo en su conjunto. Los errores de redondeo pueden acumularse de tal modo que resten validez a los resultados (inestabilidad numérica).

No obstante, en general, una acotación del error cometido después de miles o millones de operaciones suele ser un valor muy pesimista, ya que supondría que el error es siempre de igual signo y se acumula, hecho que en la práctica rara vez ocurre.

Proceso inestable

Es aquel en el que los pequeños errores que se producen en una de sus etapas se agrandan en etapas posteriores y degradan seriamente la exactitud del cálculo en su conjunto. Los errores de redondeo pueden acumularse de tal modo que resten validez a los resultados (inestabilidad numérica).

No obstante, en general, una acotación del error cometido después de miles o millones de operaciones suele ser un valor muy pesimista, ya que supondría que el error es siempre de igual signo y se acumula, hecho que en la práctica rara vez ocurre.

Proceso inestable

Por ejemplo, la sucesión de término general $x_n = \left(\frac{1}{2}\right)^n$ puede ser también definida el siguiente proceso inductivo:

$$(I)\begin{cases} x_0 = 1,\ x_1 = \dfrac{1}{2}, \\ x_n = \dfrac{23}{2}x_{n-1} - \dfrac{11}{2}x_{n-2},\ n \geq 2, \end{cases}$$

ya que si, por hipótesis, $x_k = \left(\frac{1}{2}\right)^k$, para $k \leq n$, entonces

$$x_{n+1} = \frac{23}{2}x_n - \frac{11}{2}x_{n-1} = \left(\frac{23}{2} - 11\right)\left(\frac{1}{2}\right)^n = \left(\frac{1}{2}\right)^{n+1}.$$

Proceso inestable

Ahora bien, si considerásemos los cálculos sólo con seis cifras decimales, en coma fija, obtendríamos que $x_{15} = 0.000031$, y mediante la inducción (I), se tiene:

$$
\begin{aligned}
x_0 &= 1.000000, \\
x_1 &= 0.500000, \\
x_2 &= 0.250000, \\
&\vdots \\
x_{13} &= 0.936518, \\
x_{14} &= 10.300417, \\
x_{15} &= 113.303947,
\end{aligned}
$$

por lo que el error de redondeo en este caso sería bastante significativo y desvirtuaría la definición inductiva (I) de la sucesión.

Orden de aproximación

Muchas veces disponemos de una expresión que nos da, para cualquier $h > 0$, un valor $A(h)$ como aproximación del valor exacto $s \in \mathbb{R}$ y se sabe que existe una constante $C > 0$ tal que

$$\lim_{h \to 0} \left| \frac{A(h) - s}{h^n} \right| = C.$$

Entonces se dice que el error absoluto $|A(h) - s|$ es un infinitésimo de orden n, y se expresa

$$|A(h) - s| = O\left(h^n\right).$$

Para valores positivos y pequeños de h podemos considerar

$$|A(h) - s| \simeq Ch^n.$$

Orden de exactitud

En ocasiones la expresión $A \equiv A(f,h) \equiv A(f)$ es una combinación de datos relativos a una función f (por ejemplo, una combinación de valores de f y de algunas de sus derivadas) y puede depender de h o no, mientras que el valor s corresponde a una manipulación de f que sólo puede realizarse de forma exacta para ciertas funciones.

En tal caso, los polinomios constituyen unas funciones test para medir la bondad de la aproximación, y se dice que A tiene orden de exactitud n si es exacta para para las funciones $1, x, \ldots, x^n$ y comete error distinto de cero para la función x^{n+1}

Ejerc. 5.- Obtener el orden de exactitud y el de aproximación de la fórmula de derivación numérica $f''(a) \simeq \frac{f(a-h)-2f(a)+f(a+h)}{h^2}$.

Ejerc. 6.- Obtener el orden de exactitud de la fórmula de integración numérica $\int_0^1 f(x)\,dx \simeq \frac{1}{6}f(\frac{1}{4}) + \frac{2}{3}f(\frac{1}{2}) + \frac{1}{6}f(\frac{3}{4})$.

Orden de exactitud

En ocasiones la expresión $A \equiv A(f,h) \equiv A(f)$ es una combinación de datos relativos a una función f (por ejemplo, una combinación de valores de f y de algunas de sus derivadas) y puede depender de h o no, mientras que el valor s corresponde a una manipulación de f que sólo puede realizarse de forma exacta para ciertas funciones.

En tal caso, los polinomios constituyen unas funciones test para medir la bondad de la aproximación, y se dice que A tiene orden de exactitud n si es exacta para para las funciones $1, x, \ldots, x^n$ y comete error distinto de cero para la función x^{n+1}

Ejerc. 5.- Obtener el orden de exactitud y el de aproximación de la fórmula de derivación numérica $f''(a) \simeq \frac{f(a-h)-2f(a)+f(a+h)}{h^2}$.

Ejerc. 6.- Obtener el orden de exactitud de la fórmula de integración numérica $\int_0^1 f(x)\,dx \simeq \frac{1}{6}f(\frac{1}{4}) + \frac{2}{3}f(\frac{1}{2}) + \frac{1}{6}f(\frac{3}{4})$.

Orden de exactitud

En ocasiones la expresión $A \equiv A(f,h) \equiv A(f)$ es una combinación de datos relativos a una función f (por ejemplo, una combinación de valores de f y de algunas de sus derivadas) y puede depender de h o no, mientras que el valor s corresponde a una manipulación de f que sólo puede realizarse de forma exacta para ciertas funciones.

En tal caso, los polinomios constituyen unas funciones test para medir la bondad de la aproximación, y se dice que A tiene orden de exactitud n si es exacta para para las funciones $1, x, \ldots, x^n$ y comete error distinto de cero para la función x^{n+1}

Ejerc. 5.- Obtener el orden de exactitud y el de aproximación de la fórmula de derivación numérica $f''(a) \simeq \frac{f(a-h)-2f(a)+f(a+h)}{h^2}$.

Ejerc. 6.- Obtener el orden de exactitud de la fórmula de integración numérica $\int_0^1 f(x)\,dx \simeq \frac{1}{6}f(\frac{1}{4}) + \frac{2}{3}f(\frac{1}{2}) + \frac{1}{6}f(\frac{3}{4})$.

Orden de exactitud: ejemplo

Por ejemplo, si usamos

$$A(f,h) = \frac{f(a+h) - f(a)}{h}$$

para aproximar el valor $s = f'(a)$, cuando $f(x) = 1$ se obtiene $A(h) = 0 = f'(a)$ y para $f(x) = x$ resulta que

$$A(h) = 1 = f'(a)$$

mientras que si $f(x) = x^2$, se verifica que

$$A(h) = \frac{2ah + h^2}{h} = 2a + h \neq 2a = f'(a).$$

Por tanto el orden de exactitud es 1.

Orden de exactitud: ejemplo

Por ejemplo, si usamos

$$A(f,h) = \frac{f(a+h) - f(a)}{h}$$

para aproximar el valor $s = f'(a)$, cuando $f(x) = 1$ se obtiene $A(h) = 0 = f'(a)$ y para $f(x) = x$ resulta que

$$A(h) = 1 = f'(a)$$

mientras que si $f(x) = x^2$, se verifica que

$$A(h) = \frac{2ah + h^2}{h} = 2a + h \neq 2a = f'(a).$$

Por tanto el orden de exactitud es 1.

Orden de exactitud: ejemplo

Por ejemplo, si usamos

$$A(f,h) = \frac{f(a+h) - f(a)}{h}$$

para aproximar el valor $s = f'(a)$, cuando $f(x) = 1$ se obtiene $A(h) = 0 = f'(a)$ y para $f(x) = x$ resulta que

$$A(h) = 1 = f'(a)$$

mientras que si $f(x) = x^2$, se verifica que

$$A(h) = \frac{2ah + h^2}{h} = 2a + h \neq 2a = f'(a).$$

Por tanto el orden de exactitud es 1.

Orden de exactitud: ejemplo

Por ejemplo, si usamos

$$A(f,h) = \frac{f(a+h) - f(a)}{h}$$

para aproximar el valor $s = f'(a)$, cuando $f(x) = 1$ se obtiene $A(h) = 0 = f'(a)$ y para $f(x) = x$ resulta que

$$A(h) = 1 = f'(a)$$

mientras que si $f(x) = x^2$, se verifica que

$$A(h) = \frac{2ah + h^2}{h} = 2a + h \neq 2a = f'(a).$$

Por tanto el orden de exactitud es 1.

Orden de exactitud: ejemplo

Por ejemplo, si usamos

$$A(f,h) = \frac{f(a+h) - f(a)}{h}$$

para aproximar el valor $s = f'(a)$, cuando $f(x) = 1$ se obtiene $A(h) = 0 = f'(a)$ y para $f(x) = x$ resulta que

$$A(h) = 1 = f'(a)$$

mientras que si $f(x) = x^2$, se verifica que

$$A(h) = \frac{2ah + h^2}{h} = 2a + h \neq 2a = f'(a).$$

Por tanto el orden de exactitud es 1.

Rapidez y coste computacional

Una vez establecida la convergencia de un método, surge el problema de estimar la rapidez con que las soluciones aproximadas tienden a la solución exacta.

Si el método converge lentamente, el número de cálculos necesarios para alcanzar una precisión aceptable puede llegar a ser enorme, haciendo prohibitiva la aplicación del método.

Aun cuando la rapidez de convergencia sea un mérito, a la hora de comparar métodos numéricos hay que tener en cuenta otros aspectos, como pueda ser la complejidad de los cálculos que el método requiere.

El *coste computacional* de un método numérico puede ser medido también en función de la cantidad de memoria del ordenador que se precisa para almacenar los datos iniciales, resultados y cómputos intermedios.

Rapidez y coste computacional

Una vez establecida la convergencia de un método, surge el problema de estimar la rapidez con que las soluciones aproximadas tienden a la solución exacta.

Si el método converge lentamente, el número de cálculos necesarios para alcanzar una precisión aceptable puede llegar a ser enorme, haciendo prohibitiva la aplicación del método.

Aun cuando la rapidez de convergencia sea un mérito, a la hora de comparar métodos numéricos hay que tener en cuenta otros aspectos, como pueda ser la complejidad de los cálculos que el método requiere.

El *coste computacional* de un método numérico puede ser medido también en función de la cantidad de memoria del ordenador que se precisa para almacenar los datos iniciales, resultados y cómputos intermedios.

Rapidez y coste computacional

Una vez establecida la convergencia de un método, surge el problema de estimar la rapidez con que las soluciones aproximadas tienden a la solución exacta.

Si el método converge lentamente, el número de cálculos necesarios para alcanzar una precisión aceptable puede llegar a ser enorme, haciendo prohibitiva la aplicación del método.

Aun cuando la rapidez de convergencia sea un mérito, a la hora de comparar métodos numéricos hay que tener en cuenta otros aspectos, como pueda ser la complejidad de los cálculos que el método requiere.

El *coste computacional* de un método numérico puede ser medido también en función de la cantidad de memoria del ordenador que se precisa para almacenar los datos iniciales, resultados y cómputos intermedios.

Rapidez y coste computacional

Una vez establecida la convergencia de un método, surge el problema de estimar la rapidez con que las soluciones aproximadas tienden a la solución exacta.

Si el método converge lentamente, el número de cálculos necesarios para alcanzar una precisión aceptable puede llegar a ser enorme, haciendo prohibitiva la aplicación del método.

Aun cuando la rapidez de convergencia sea un mérito, a la hora de comparar métodos numéricos hay que tener en cuenta otros aspectos, como pueda ser la complejidad de los cálculos que el método requiere.

El *coste computacional* de un método numérico puede ser medido también en función de la cantidad de memoria del ordenador que se precisa para almacenar los datos iniciales, resultados y cómputos intermedios.

Ejercicios

Calcular el número de operaciones necesarias para resolver cada uno de los siguientes problemas:

7.- Hallar $\sum_{i=1}^{n} x_i$, con $x_i \in \mathbb{R}$, para $i = 1, \ldots, n$.

8.- Resolver el sistema de ecuaciones lineales con coeficientes reales:

$$\begin{aligned} ax + by &= e, \\ cx + dy &= f, \end{aligned}$$

por el método de sustitución, si $ad - bc \neq 0$.

9.- Resolver el sistema anterior por el método de reducción.

10.- Hallar el producto escalar de dos vectores de $\mathbb{R}^n$.

Analisis a posteriori del error

Uno de los procedimientos más simples para estimar la veracidad de los resultados es el *método de permutación-perturbación*, que consiste en resolver el mismo problema varias veces, permutando las operaciones (o parte del proceso) que sea posible y redondeando los datos iniciales y los resultados intermedios cada vez, por exceso o por defecto, de forma aleatoria.

En la práctica, si después de efectuar tres o cuatro permutaciones-perturbaciones los resultados difieren poco en términos relativos es casi seguro que son fiables.

A veces el condicionamiento es tan malo que al aplicar varias perturbaciones los resultados son tan dispares que resulta evidente.

Veremos ejemplos de todo ello en la correspondiente práctica de ordenador con MATHEMATICA.

Analisis a posteriori del error

Uno de los procedimientos más simples para estimar la veracidad de los resultados es el *método de permutación-perturbación*, que consiste en resolver el mismo problema varias veces, permutando las operaciones (o parte del proceso) que sea posible y redondeando los datos iniciales y los resultados intermedios cada vez, por exceso o por defecto, de forma aleatoria.

En la práctica, si después de efectuar tres o cuatro permutaciones-perturbaciones los resultados difieren poco en términos relativos es casi seguro que son fiables.

A veces el condicionamiento es tan malo que al aplicar varias perturbaciones los resultados son tan dispares que resulta evidente.

Veremos ejemplos de todo ello en la correspondiente práctica de ordenador con MATHEMATICA.

Analisis a posteriori del error

Uno de los procedimientos más simples para estimar la veracidad de los resultados es el *método de permutación-perturbación*, que consiste en resolver el mismo problema varias veces, permutando las operaciones (o parte del proceso) que sea posible y redondeando los datos iniciales y los resultados intermedios cada vez, por exceso o por defecto, de forma aleatoria.

En la práctica, si después de efectuar tres o cuatro permutaciones-perturbaciones los resultados difieren poco en términos relativos es casi seguro que son fiables.

A veces el condicionamiento es tan malo que al aplicar varias perturbaciones los resultados son tan dispares que resulta evidente.

Veremos ejemplos de todo ello en la correspondiente práctica de ordenador con MATHEMATICA.

Analisis a posteriori del error

Uno de los procedimientos más simples para estimar la veracidad de los resultados es el *método de permutación-perturbación*, que consiste en resolver el mismo problema varias veces, permutando las operaciones (o parte del proceso) que sea posible y redondeando los datos iniciales y los resultados intermedios cada vez, por exceso o por defecto, de forma aleatoria.

En la práctica, si después de efectuar tres o cuatro permutaciones-perturbaciones los resultados difieren poco en términos relativos es casi seguro que son fiables.

A veces el condicionamiento es tan malo que al aplicar varias perturbaciones los resultados son tan dispares que resulta evidente.

Veremos ejemplos de todo ello en la correspondiente práctica de ordenador con MATHEMATICA.