TEMA 1: INTRODUCCIÓN AL CÁLCULO NUMÉRICO

GENERALIDADES

1. INTRODUCCIÓN. ALGUNAS IDEAS Y CONCEPTOS INICIALES

Cuando se quiere resolver un problema matemático (e.g. calular una integral o resolver una ecuación diferencial) de forma numérica (no exacta, no analítica) se suele proceder de la siguiente forma:

1º) Se sustituye el problema inicial por un algoritmo de cálculo, que típicamente contiene un parámetro "n" ó "h".

2º) Probar la convergencia del algoritmo, es decir, asegurar que las aproximaciones x_n a la solución x del problema inicial, son tan próximas como se desee, estimando la rapidez o velocidad de convergencia.

NOTA:- 2º) no siempre es fácil en "problemas reales". En estos casos, la "bondad" de la aproximación numérica se ha de estimar atendiendo a consideraciones físicas, comparando con ejemplos "bien ", con datos experimentales, etc.

3º) Otra propiedad muy deseable en todo método o esquema numérico es que sea "estable", lo que significa que pequeñas modificaciones en los datos no ocasionen fuertes cambios en el resultado final.

4º) Finalmente, también es deseable que el esquema numérico sea "barato", desde un punto de vista computacional, es decir que minimice el número de operaciones efectuadas (directamente relacionado con el coste computacional) y la memoria requerida para almacenar datos.

En resumen, hemos de usar métodos numéricos estables y que produciendo errores dentro de márgenes admisibles, requieran de un coste computacional lo menor posible.

Veamos algunos ejemplos que ilustran estos criterios:

Ejemplo 1: coste computacional

Supongamos que queremos evaluar el polinomio

$$P(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

en un punto dado x_0.

Veamos un par de formas (métodos numéricos) de hacerlo.

Esquema 1: forma tradicional. Vamos a contar el número de operaciones:

Multiplicaciones: $a_0 + \underbrace{a_1 x_0}_{1} + \underbrace{a_2 x_0^2}_{2} + \dots + \underbrace{a_n x_0^n}_{2}$

$= 2n - 1$ multiplicaciones

Sumas $= n$.

Coste computacional del esquema $= 2n - 1 + n = 3n - 1$.

Esquema 2: algoritmo de Horner

Reescribimos $P_n(x)$ en la forma

$$P(x) = a_0 + x\,(a_1 + x\,(a_2 + x\,(a_3 + \dots + x\,(a_n)\dots))$$

que conduce al algoritmo

$P_n(x) = a_n$

$P_{n-1}(x) = a_{n-1} + x P_n(x)$ → 1 suma + 1 multiplicación

$P_{n-2}(x) = a_{n-2} + x P_{n-1}(x)$ → 1 suma + 1 multiplicación

$P(x) = P_0(x) = a_0 + x P_1(x)$ → 1 suma + 1 multip.

Por tanto, el algoritmo de Horner requiere $2n$ operaciones, más barato (computacionalmente) que el método tradicional, que requiere $3n-1$ operaciones.

Ejemplo 2: un algoritmo inestable

Supongamos que queremos calcular $\int_0^1 \frac{x^n}{10+x}\,dx$.

Para ello utilizamos la fórmula de reducción

$$y_n = \int_0^1 \frac{x^n}{10+x}\,dx = \int_0^1 \frac{x^{n-1}(10+x-10)}{10+x}\,dx$$

$$= \int_0^1 x^{n-1}\,dx - 10\int_0^1 \frac{x^{n-1}}{10+x}\,dx$$

$$= \frac{1}{n} - 10\,y_{n-1}.$$

Como $y_0 = \int_0^1 \frac{dx}{10+x} = \log(10+x)\Big|_0^1 = \log\left(\frac{11}{10}\right)$,

podemos usar el algoritmo:

$$\begin{cases} y_0 = \log(11/10) \\ y_n = \frac{1}{n} - 10\,y_{n-1}, & n \geq 1. \end{cases}$$

Supongamos que al calcular y_0 cometemos un pequeño error e_0 de modo que en realidad calculamos

$$\bar{y}_0 = y_0 + e_0.$$

Ese pequeño error lo vamos arrastrando y así, en el paso n-ésimo tendremos

$$\bar{y}_n = \frac{1}{n} - 10\,\bar{y}_{n-1}.$$

El error acumulado en el paso n-ésimo es

$$e_n = \bar{y}_n - y_n = \frac{1}{n} - 10\,\bar{y}_{n-1} - \frac{1}{n} + 10\,y_{n-1}$$

$$= -10\,(\bar{y}_{n-1} - y_{n-1})$$

$$= -10\,e_{n-1}$$

$$= (-10)\cdot(-10)\,e_{n-2}$$

$$= \cdots\cdots$$

$$= (-10)^n e_0.$$

Por tanto, en cada paso el error se multiplica por (-10).

Este es un ejemplo típico de un algoritmo inestable, el cual, obviamente no debe usarse en Cálculo Numérico.

2. ERRORES EN LOS MÉTODOS NUMÉRICOS

Tipos de errores:

a) Errores en los datos iniciales, por ejemplo si son resultado de alguna medida con algún instrumento.

b) Errores de redondeo, debidos al hecho de que el ordenador maneja sólo un número finito de dígitos.

c) Errores de truncatura o discretización, que provienen de sustituir un problema continuo por uno discreto, por ejemplo una derivada por un cociente incremental, o una integral por una suma de un número finito de términos.

En Cálculo Numérico consideramos únicamente los errores de tipo b) y c). De forma indirecta también tendremos en cuenta los de tipo a) ya que buscaremos algoritmos numéricos estables.

Obtenida una aproximación $\bar{x}$ del verdadero valor x de una determinada magnitud, se llama:

- error absoluto: $x - \bar{x}$

Si $\bar{x} > x$ se dice que la aproximación es por exceso y si $\bar{x} < x$ por defecto.

Generalmente, interesa el error sin signo

$$e = |x - \bar{x}|.$$

- error relativo: $\frac{|x-\bar{x}|}{|x|}$ si $x \neq 0$.

- error porcentual = error relativo $\times 100$.

Errores de redondeo en los ordenadores

Los ordenadores trabajan con un número finito de números, los cuales pueden ser representados de diversas formas:

- Aritmética de punto flotante decimal, donde cada número x se expresa como

$$x = s \cdot m \cdot 10^{e}$$

  * $s \in \{-1, 1\}$ es el signo del número
  * $m \in [1, 10]$ es la mantisa
  * e es el exponente.

Por ejemplo,

$$-\frac{3237}{126} = -2.569047619047619 \times 10^{1}$$

donde $s = -1$, $m = 2.569047619047619$, $e = 1$.

· Aritmética de punto flotante binaria, donde

$$x = \sigma \cdot \bar{x} \cdot 2^e$$

$\sigma \in \{-1, 1\}$ es el signo

$\bar{x} \in [1_2, 10_2]$ representado en base binaria

$e \in [-1022, 1023]$ es el exponente

Así,

$$-\frac{3237}{126} = -1.10011011000011000011000011_2 \cdot 2^4$$

Es frecuente trabajar en "doble precisión", que es un estándar de la IEEE (Institute of Electrical and Electronics Engineers) que consiste en el uso de 64 bits para representar un número en un ordenador. De ellos:

(i) Un bit se usa para el signo

(ii) 53 bits se usan para almacenar la mantisa

(iii) 11 bits se usan para el exponente.

. La forma de medir la precisión con la cual los números son almacenados en un ordenador es a través del llamado "épsilon de la máquina", denotado eps. Se define como la distancia entre 1 y el siguiente número (mayor que 1) que puede ser almacenado en el formato elegido.

Por ejemplo, en doble precisión $eps = 2^{-52} \simeq 2.22 \cdot 10^{-16}$

lo que significa que los números racionales pueden ser almacenados con exactitud hasta 16 decimales.

Denotemos por τ un número dado y por t el número representable en un ordenador más cercano a τ. Por tanto

$$(1-eps)t \leq \tau \leq (1+eps)t$$

de donde $$-eps \leq \frac{\tau - t}{t} \leq eps.$$

Es decir, eps es una medida del error relativo que se comete al "redondear" un número en un ordenador.