Introducción al análisis numérico

Javier Segura

Universidad de Cantabria

Cálculo Numérico I. Tema 1

Contenidos:

Estructura de la presentación:

Tema 1

Sistemas de números y conversiones

Programación de un algoritmo numérico: sus limitaciones intrínsecas (precisión y rango de valores admisibles) están ligadas a la forma de representación de los números enteros y decimales en formato digital.

Sistemas de numeración en base q.

- El sistema de numeración decimal utiliza como base de numeración el 10 (dígitos 0...9).
- Un número en base q se denota como $(a_n a_{n-1} \ldots a_1 a_0 . b_1 b_2 \ldots b_k \ldots)_q$ donde a_i y b_j pertenecen al conjunto de los q dígitos elementales. Estos q dígitos representarán valores desde 0 hasta $q-1$.

Sistemas de números y conversiones

- Por definición:

$$\begin{aligned}(a_n a_{n-1}...a_1 a_0.b_1 b_2...b_k...)_q &= a_n q^n + a_{n-1} q^{n-1} + ... + a_1 q \\ &+ a_0 q^0 + b_1 q^{-1} + b_2 q^{-2} + ... \\ &+ b_k q^{-k} + ...\end{aligned}$$

- El sistema natural de numeración digital es el binario (base 2), utilizando sólo los dígitos 0 y 1.

Sistemas de números y conversiones

Ejemplos:

- Decimal:
  $(123.25)_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}.$
- Binario:

$$\begin{aligned}(1011.01)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} \\ &\quad +1 \times 2^{-2} = 11.25.\end{aligned}$$

Sistemas de números y conversiones

Conversión de base decimal a base q

- Conversión de la parte entera

$$(a_n...a_0)_q = (a_n...a_1)_q \times q + (.a_0)_q \times q = (a_n...a_1)_q \times q + (a_0)_q$$

Al dividir un número entero entre q el resto es el dígito menos significativo del número en base q. Dividiendo sucesivamente (hasta llegar al cociente 0) obtenemos los sucesivos dígitos del número en base q.

- Conversión de la parte fraccionaria

$$(.b_1 b_2...b_k)_q \times q = (b_1)_q + (.b_2...b_k)_q$$

La parte entera del resultado de multiplicar nuestro número por q es el primer dígito tras el punto decimal. Reteniendo la parte fraccionaria que resulta en cada paso y repitiendo sucesivamente el proceso, vamos obteniendo el resto de cifras tras el punto.

Sistemas de números y conversiones

Conversión de base decimal a base q

Ejemplo: Escribamos $(26.1)_{10}$ en base 2.

- Parte entera: Dividiendo sucesivamente, tenemos que:

$$26 = 2 \times 13 + \underline{0}\,;\ 13 = 2 \times 6 + \underline{1}\,;$$
$$6 = 2 \times 3 + \underline{0}\,;\ 3 = 2 \times 1 + \underline{1}\,;\ 1 = 2 \times 0 + \underline{1}$$

  Leyendo de izquierda a derecha los números subrayados:
  $(26)_{10} = (11010)_2$

- Parte fraccionaria: Multiplicando sucesivamente por dos y separando la parte fraccionaria:

$$0.1 \times 2 = \underline{0}.2\,;\ 0.2 \times 2 = \underline{0}.4\,;\ 0.4 \times 2 = \underline{0}.8\,;$$
$$0.8 \times 2 = \underline{1}.6\,;\ 0.6 \times 2 = \underline{1}.2\,;\ 0.2 \times 2 = \ldots$$

Sistemas de números y conversiones

Conversión de base decimal a base q

Leyendo de izquierda a derecha tenemos los dígitos de la parte fraccionaria (subrayados) luego:

$$(0.1)_{10} = (0.0\underline{0011})_2$$

donde las cifras subrayadas son las cifras periódicas.
Obsérvese que el número 0.1 tiene infinitas cifras distintas de cero cuando se escribe en base 2!
Sumando los resultados de la parte entera y la fraccionaria tenemos que

$$(26.1)_{10} = (11010.0\underline{0011})_2$$

Un programa desastroso:

```
x=0;
while x~=10
x=x+0.1;
end
```

Estándar IEEE

Números enteros (32-bits)

- Asumimos que cada palabra tiene una longitud de 32 bits
- Asignamos uno de los 32 bits para designar el signo
- Las restantes 31 posiciones son utilizadas para representar los dígitos del módulo del número entero

Estándar IEEE

Números no enteros

- Números con parte fraccionaria: utilizamos el formato de punto (o coma) flotante
- Esta representación es la correspondiente versión binaria de la conocida notación científica o exponencial para los números decimales:

$$\pm M \times 10^E\,,\ 1 \leq M < 10$$

pero utilizando base 2:

$$x = \pm M \times 2^E\,,\ 1 \leq M < 2$$

donde M es la mantisa y E el exponente.

Estándar IEEE

El número de dígitos que se pueden almacenar de la mantisa y exponente es limitado!

Hay dos tipos de precisión (simple y doble): difieren en el número de bits de los que se dispone para almacenar las cifras
Precisión simple: 32 bits, de los cuales

- 1 corresponde al signo,
- 8 al exponente, luego -126 $\leq$ E $\leq$ 127
- 23 a la mantisa

Precisión doble: 64 bits, de los cuales

- 1 corresponde al signo,
- $\approx$11 al exponente: -1022 $\leq$ E $\leq$ 1023 ($2^{11} = 2 \times 1024$)
- 52 a la mantisa

La especificiación del número de bits para almacenar la mantisa y el exponente determinan los límites de overflow y underflow y el denominado ϵ-máquina .

Estructura de la presentación:

Errores

Definición:
Si x_A es una aproximación del verdadero valor x_T, definimos entonces:

$$\text{Error absoluto:} \quad E_{abs} = |x_T - x_A|$$

$$\text{Error relativo:} \quad E_{rel} = |1 - \frac{x_A}{x_T}|, \ \text{ si } \ x_T \neq 0.$$

El error relativo mide el número de cifras significativas exactas de x_A. Así, si

$$2E_{rel} < 10^{-m}\,,\ m \in \mathbb{N}$$

se dirá que x_A tiene m cifras significativas exactas.

Fuentes de error

- Los datos de entrada de nuestro algoritmo están afectados de cierto error. Importante estudiar como se propaga el error.
- Errores de punto flotante: errores de redondeo, pérdida de cifras significativas por cancelaciones, problemas de underflow/overflow
- Errores de truncamiento o discretización

Fuentes de error

Un ejemplo ... catastrófico

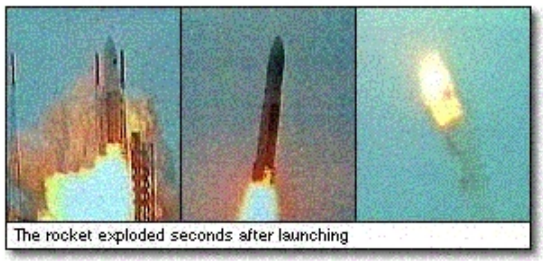
Unmanned European rocket explodes on first flight

June 4, 1996
Web posted at: 12:00 p.m. EDT (1600 GMT)

KOUROU, French Guiana (CNN) -- Europe's newest unmanned satellite-launching rocket, the Ariane 5, intentionally was blown up Tuesday just seconds after taking off on its maiden flight.

A spokesman for Arianespace said the rocket was destroyed by its controllers.

"It's been confirmed that the vehicle was deliberately destroyed by the safety people," said spokesman Ian Pryke. "Why? I don't know. There were no, repeat no, injuries. Most of the debris came down in the mangroves and the sea, and the winds carried the fumes away from the people out over the sea."


The rocket exploded seconds after launching

RELATED SITES & STORIES

El matemático Jacques-Louis Lions del Collège de France fue encargado de dirigir la investigación.

Más información en: SIAM News, Vol. 29. Number 8, October 1996

Propagacion de errores

Condición

La condición de una función $f(x)$ mide la sensibilidad de los valores de $f(x)$ a pequeños cambios en x y se define como

$$C = \left| \frac{E_{rel}(f(x))}{E_{rel}(x)} \right|$$

donde $E_{rel}(f(x))$ es el error relativo de $f(x)$ para un error relativo $E_{rel}(x)$ en x.
Para funciones $f(x)$ de una variable real definimos los números de condición como

$$C(x) = \left| x \frac{f'(x)}{f(x)} \right|$$

Propagación de errores

Condición

Cuando para un x dado $0 < C(x) < 1$ para ese x se dirá que el problema (cálculo de f) está bien condicionado (y cuanto menor sea C mejor condicionado), mientras que **si $C(x) > 1$ el problema estará mal condicionado**. Si $C(x) = 1$, el error relativo se mantiene.

Ejemplo:

La función $f(x) = \sqrt{x}$ está bien condicionada, pues $C(x) = 1/2$, luego el error relativo se reduce.

En cambio $f(x) = x^2 - 1$ está mal condicionada para $x \simeq 1$ pues $C(x) = \left|\frac{2x^2}{x^2-1}\right|$

El concepto de condicionamiento se puede extender a situaciones más generales que la de una función de una variable.

Propagación de errores

Estabilidad

Un concepto relacionado pero diferente es el de **(in)estabilidad de un algoritmo**: describe la sensibilidad de un método numérico específico respecto a los errores de redondeo cometidos durante su ejecución en aritmética de precisión finita.

Observación: **la condición no depende de errores de redondeo** mientras que **la estabilidad de un algoritmo sí depende del condicionamiento de la función que queramos evaluar**.

Propagación de errores

Estabilidad

Ejemplo:

Dada la función $f(x) = \sqrt{x+1} - \sqrt{x}$, su número de condición es:

$$C(x) = \left| x \frac{f'(x)}{f(x)} \right| = \frac{x}{2\sqrt{x}\sqrt{x+1}}$$

y vemos que $C(x) < 1/2$ para $x > 0$, luego la función está bien condicionada. Sin embargo, el algoritmo para calcular x consistente en ir realizando las operaciones implicadas en $f(x) = \sqrt{x+1} - \sqrt{x}$, a saber:

1. Input: x
2. $y = x + 1$
3. $f_1 = \sqrt{x+1}$
4. $f_2 = \sqrt{x}$
5. $f = f_1 - f_2$

es inestable para x grande por culpa del paso 5 (hay cancelaciones entre números similares).

Estructura de la presentación:

Eficiencia de un algoritmo numérico

Un ejemplo práctico

Supongamos que nos planteamos evaluar el polinomio

$$P(x) = 2 + 4x - 5x^2 + 2x^3 - 6x^4 + 8x^5 + 10x^6$$

Contando con que cada potencia de exponente k entero cuente como $k-1$ productos, tendríamos que el número total de productos para evaluar el polinomio de forma directa es:

$$1 + 2 + 3 + 4 + 5 + 6 = 21$$

y el número de sumas es 6.
Una forma mejor de proceder es ir calculando primero las potencias de forma sucesiva:

$$x^2 = x\,x\,,\ x^3 = x\,x^2\,,\ x^4 = x\,x^3\,,\ x^5 = x\,x^4\,,\ x^6 = x\,x^5$$

con lo que sólo se añade una nueva multiplicación por potencia, haciendo un total de

$$1 + 2 + 2 + 2 + 2 + 2 = 11$$

Eficiencia de un algoritmo numérico

Un ejemplo práctico

Pero aún se puede hacer mejor reescribiendo

$$P(x) = 2 + x(4 + x(-5 + x(2 + x(-6 + x(8 + x10)))))$$

con lo que sólo necesitamos 6 multiplicaciones (y el número de sumas no cambia).

Por tanto para evaluar un polinomio de grado n en el que ninguno de los coeficientes es cero, se necesitan $n(n+1)/2$ multiplicaciones por el primer método, $2n-1$ por el segundo y n para el tercero.

Este método es además sencillo de programar: es el **algoritmo de Horner o de división sintética**.

Eficiencia de un algoritmo numérico

Un ejemplo práctico

Algoritmo de Horner

Inputs: z, $p(x) = a_0 + a_1\, x + ... + a_n\, x^n\,,\ \ a_n \neq 0$
Output: $p(z)$.

(1) $b = a_n$

(2) Repetir mientras $n > 0$

(3) $n = n - 1$

(4) $b = a_n + z * b$

(5) Volver a **(2)**

(6) $p(z) = b$.