PROBLEMAS EN ARITMETICA DE PUNTO FLOTANTE Y OBTENCION DE CARACTERISTICAS INTERNAS DE COMPUTADORAS DIGITALES.

GUSTAVO RODRIGUEZ GOMEZ *
DAVID CARRASCO VILLARREAL **

RESUMEN

Se presentan algunas diferencias entre la "aritmética usual" y la de punto flotante, algunas limitaciones que se deben tomar en cuenta en el uso y diseño de códigos en métodos numéricos, y se da un algoritmo que revela algunas características internas de una computadora: base del sistema de punto flotante, épsilon de la máquina, número de digitos de la mantisa y tipo de error introducido.

I.1. INTRODUCCION

En la vida diaria se acostumbra a manejar los números en el sistema decimal bajo el supuesto de contar con una aritmética de precisión infinita, es decir, exacta (1). En contraste, una computadora digital, no trabaja (en general) en base diez y su aritmética es diferente a la que se usa, por ejemplo, si se almacena un número que tenga demasiados dígitos, la computadora lo "truncará" o "redondeará" para poder almacenarlo. Esto ocasiona ciertas limitaciones en los cálculos de la computadora que deben ser entendidos.

Los cálculos científicos se efectúan generalmente en aritmética de punto flotante. Un número en su representación en punto flotante que tenga n dígitos en base β tiene la forma:

$$x = \pm (.d_1 d_2 \cdots d_n)_\beta \beta^e$$

(1) (de aquí en lo sucesivo, se llamará aritmética de la vida diaria)

* Investigador del IIE (Departamento de Simulación)
** Becario del IIE (Departamento de Simulación)

donde $\beta > 1$

$0 \leq d_i \leq \beta - 1$, $i = 1, 2, \ldots, n$

$m \leq e \leq M$, $m \leq 0$, $M > m$

β, m, M, e son enteros.

Observe los siguientes elementos:

a) Un signo: + ó -
b) Una mantisa con n dígitos: $(.d_1 d_2 .. d_n)\beta$
c) Una base: β
d) Un exponente entero: e

Es claro que una computadora siempre tendrá un número finito de dígitos en la mantisa, ésta puede variar mucho de acuerdo a la computadora.

Por otra parte hay un elemento importante en la aritmética de punto flotante, es el llamado épsilon de la máquina, éste se define como el menor número positivo tal que $1 + \epsilon > 1$. Este concepto se desarrolla en la sección 1.2.

El objetivo de este trabajo es:

a) Presentar algunas diferencias entre la aritmética que usamos en la vida diaria y la empleada por una computadora.

b) Presentar ciertas restricciones que se deben tener en cuenta al diseñarse códigos en métodos numéricos.

c) Presentar un algoritmo general hecho por Malcom y corregido por Gentleman y Marovich que determina:

1) El número de dígitos en la mantisa: n
2) La base del sistema: β
3) El épsilon de la máquina: ϵ
4) El tipo de error introducido: truncamiento o redondeo

1.2. ARITMETICA DE PUNTO FLOTANTE.

Supóngase que se trabaja en un sistema numérico de punto flotante, con las siguientes características:

1. Una base β
2. n dígitos en la mantisa, $0 < n < +\infty$
3. Un exponente: $m \leq e \leq M$

donde β, n, m, e son enteros.

Supóngase que se tiene un número x en el sistema usual, al que se quiere representar en el sistema numérico anterior, se convendrá en denotar por $fl(x)$ al número representado en el sistema numérico de punto flotante, luego $fl(x)$ es de la forma:

$$\pm (.d_1 d_2 \ldots d_n)_\beta \beta^e \qquad (2.1)$$

Si $d_1 \neq 0$ se dirá que $fl(x)$ está en forma normalizada

Todos los números que la computadora almacena se encuentran en forma normalizada, con excepción del cero, en el cual

$$d_1 = d_2 = \cdots = d_n = 0$$

Hay dos maneras de traducir este número x en $fl(x)$:

Redondeado
o
Truncado

Cuando se redondea, $fl(x)$ se elige como el número de punto flotante normalizado más cercano a x , si hay empate se usa alguna regla especial, se toma el de la derecha. Si se trunca, $fl(x)$ se escoge como el número de punto flotante normalizado más cercano entre o y x, esto es, se toman algunas d's y se desprecian otras.

Veamos el siguiente ejemplo: $\beta = 10,\ n = 4,\ x = 23456$

$$fl(x) = \begin{cases} 0.2345 \times 10^5 & \text{Truncado} \\ 0.2346 \times 10^5 & \text{Redondeado} \end{cases}$$

Es aquí donde se introduce un error en la representación, que obviamente repercutirá en las operaciones aritméticas.

Por otra parte en nuestra aritmética diaria, se trabaja bajo la suposición de tener una cantidad infinita de números, en contraste con un sistema numérico de punto flotante, en el cual se tiene un total de:

$$R = 2(M-m+1)(\beta-1)\beta^{n-1} + 1 \qquad (2.2)$$

números, donde se supone que todos los números están almacenados en forma normalizada; (2.2) se deduce fácilmente de (2.1).

De lo anterior se observa que, aunque R sea muy grande siempre existirá una infinidad de números que no pueden ser representados exactamente en una aritmérica de punto flotante.

La diferencia entre x y $fl(x)$ es llamada error de redondeo, sea

$$\delta(x) = \frac{fl(x) - x}{x}, \quad x \neq 0 \qquad (2.3)$$

luego $|\delta(x)|$ es el error relativo introducido. De (2.3) se obtiene

$$fl(x) = x[1 + \delta(x)] \qquad (2.4)$$

Es posible dar una cota independiente de x a $\delta(x)$, para esto obtendremos los siguientes resultados:

Note que dentro del sistema de punto flotante el número cuyo valor absoluto es el más pequeño, es

$$+(.10\cdots0)_\beta \beta^m = \beta^{m-1}$$
$$\longleftarrow n \longrightarrow$$

si se quiere obtener el sucesor inmediato de β^{m-1}, hay que sumarle

$$+(.00\cdots01)_\beta \beta^m = \beta^{m-n}$$

Es claro entonces que la distancia entro dos números consecutivos en el intervalo $[\beta^{m-1}, \beta^m]$ es:

$$\beta^{m-n}$$

Análogamente en el intervalo $[\beta^j, \beta^{j+1}]$ donde $m-1 \leq j \leq M-1$, la distancia entre dos números consecutivos ¡es siempre igual¡ a:

$$\beta^{j+1-n} \tag{2.5}$$

Luego si la computadora representa a los números por redondeo el error introducido es a lo más $\frac{1}{2}\beta^{j+1-n}$, por truncamiento es β^{j+1-n}, esto no da la medida del error absoluto. El error relativo se obtiene haciendo el cociente del error absoluto entre β^j ; este error relativo es:

a) Redondeo $\epsilon = \frac{1}{2}\beta^{1-n}$

b) Truncamiento $\epsilon = \beta^{1-n}$

Ya que $|\beta^j| \leq |x| \Rightarrow \frac{1}{|x|} \leq \frac{1}{|\beta^j|}$ luego

$$|\delta(x)| \leq \frac{1}{2}\beta^{1-n}$$ Redondeo

$$|\delta(x)| \leq \beta^{1-n}$$ Truncamiento

Al número ϵ se le llama el épsilon de la máquina.

Una característica importante del épsilon de la máquina es la siguiente: supóngase que se esta en el intervalo $[1, \beta]$ y se quiere calcular $1 \oplus \epsilon$, donde $\oplus$ denotará la suma en el sistema de punto flotante. De lo expuesto anteriormente conocemos que la distancia entre dos números consecutivos de este intervalo es β^{1-n} y $\epsilon = \frac{1}{2}\beta^{1-n}$ en caso que el error introducido sea por redondeo, entonces $1 + \epsilon$ quedaría localizado en la mitad del intervalo $[1, 1+\beta^{1-n}]$

Al almacenar la computadora $1 \oplus \epsilon$ de acuerdo a la regla dada en I.2 se toma $1 \oplus \epsilon = 1 + \beta^{1-n} > 1$ Sin embargo si $0 < \epsilon_1 < \epsilon$ obtenemos por la misma regla.

$$1 \oplus \epsilon_1 = 1$$

Análogamente en caso de truncamiento.

EL EPSILON DE LA MAQUINA ES EL MENOR NUMERO POSITIVO TAL QUE

$$1 \oplus \epsilon > 1$$

I.3. RESTRICCIONES EN CUANTO AL USO Y DISEÑO DE CODIGOS EN METODO NUMERICO.

3.1. Limitación de la suma Aritmética de Punto Flotante.

Una observación importante es la siguiente:

¿Dados dos números x,y arbitrarios cuando

$$x \oplus y = x \ ?$$

De acuerdo a lo expuesto en la sección anterior.

$$x \oplus y = x \quad , \qquad \epsilon = \text{épsilon de la máquir}$$

si $\frac{|y|}{|x|} < \epsilon$, es decir, $|y| < \epsilon |x|$ (3.1)

en otras palabras si y es "muy pequeño" comparado con x, sus dígitos no afectan a x en la suma.

3.2. Limitación del paso de integración.

Lo anterior da una limitación fundamental del paso de integración, en programas que se diseñan (o usen) para integrar ecuaciones diferenciales. Consideremos el siguiente ejemplo: supóngase que se quiere integrar

$$y'(t) = f(t, y) \ , \ t \in [t_0, t_{max}]$$
$$y(t_0) = y_0$$

Si elegimos el método de Euler, tenemos

$$y_{n+1} = y_n + h f(t_n, y_n)$$

donde h es el paso de integración propuesto y $t_{n+1} = t_n + h$ es el avance en el tiempo. Es claro que siempre se requerirá que $t_n \oplus h > t_n$, o de lo contrario no se puede proseguir. De (3.1) el paso propuesto debe satisfacer

$$h \geq \epsilon |t| \ , \ t \in [t_0, t_{max}]$$

Lo que da una cota inferior para el paso de integración en el método de Euler. En el caso que $|t|$ sea muy grande en comparación a h, hay que escalar el valor de t ó de h.

I.4. ALGORITMO PARA EL CALCULO DE LA BASE DEL SISTEMA, NUMERO DE DIGITOS DE LA MANTISA Y TIPO DE ERROR INTRODUCIDO, DE UNA COMPUTADORA.

A continuación se presenta un algoritmo desarrollado por Malcom, M.A.(2) y corregido por Gentleman y Marovich (1) que sirve para calcular la base del sistema, dígitos de la mantisa, tipo de error introducido de una computadora dada.

4.1. La base del sistema

De (2.5) se sigue que la distancia entre dos números consecutivos del intervalo $[\beta^j, \beta^{j+1}]$ es β^{j+1-n} luego si $j = n$, se tiene $\beta^{j+1-n} = \beta$
El algoritmo consiste en encontrar un número $a \in [\beta^n, \beta^{n+1}]$ y su sucesor b inmediato dentro de este intervalo, de tal manera que

$$\beta = b - a$$

El problema que queda por resolver es como encontrar a a. Hay que observar que en el esquema que se da, no se necesita conocer a β primero para encontrar a a.

Para esto es necesario demostrar que existe un entero positivo l tal que

$$\beta^n \leq 2^l < \beta^{n+1}$$

Demostración

Ya que $\beta \geq 2$, entonces $\log_2 \beta \geq 1$; la función mayor entero $[\cdot]$ se define para todo real x y es el mayor entero tal que

$$[x] \leq x < [x] + 1$$

Primer Caso $\beta = 2$

Basta con tomar $l = n$, ya que entonces

$$2^n \leq 2^n < 2^{n+1}$$

Segundo Caso $\beta > 2$

Basta con tomar $$l = [\log_2 \beta^n + 1] = [\log_2 \beta^n] + 1$$

entonces $$\log_2 \beta^n \leqq l \leqq \log_2 \beta^n + 1 < \log_2 \beta^n + \log_2$$

ya que $\beta > 2$ luego $$\log_2 \beta^n \leqq l < \log_2 \beta^{n+1}$$

por lo anterior $$\beta^n \leqq 2^l < \beta^{n+1}$$

que es lo que se quería demostrar.

Se toman potencias sucesivas de 2, almacenándose éstas en la variable a, hasta detectarse cuando se cae por primera vez en $[\beta^n, \beta^{n+1}]$ para esto se utiliza que en los intervalos anteriores a éste, a tiene representación exacta, lo cual implica que $(a \oplus 1) \ominus a = 1$, sin embargo cuando $a \in [\beta^n, \beta^{n+1}]$ por primera vez $a \oplus 1$ no se representa exactamente, ya que $1 < \beta$ luego $a \oplus 1$ se almacenará como a, dando en consecuencia que $(a \oplus 1) \ominus a = 0$. Una vez encontrado a pasamos a la localización de b, sumando a a potencia de 2, hasta que se adquiera un valor distinto de a, e inmediatament se calcula β como la diferencia : b- a

4.1.2. Tipo de Error introducido.

Con el valor de β , se procede a calcular $a + (\beta - 1)$, ya que $\beta > \beta - 1$, $a + (\beta - 1)$ no tiene representación exacta en $[\beta^n, \beta^{n+1}]$,

luego de acuerdo a si $\begin{cases} a \oplus (\beta - 1) = a, \text{hay truncamiento} \\ a \oplus (\beta - 1) \neq a, \text{hay redondeo} \end{cases}$

4.2.3. Número de dígitos de la mantisa.

Para el cálculo de n, observe que $\beta^n \oplus 1$ no se representa exactamente en $[\beta^n, \beta^{n+1}]$ y se almacena con el valor de β^n pero si $i<n$ $\beta^i \oplus 1$ tiene representación exacta en los intervalos $[\beta^i, \beta^{i+1}]$, $i=0,1,\cdots,n-1$ por tanto podemos diferenciar entre β^i y $\beta^i \oplus 1$. El algoritmo consiste entonces en almacenar potencias sucesivas de β en una variable a y poner un contador; en el momento que

$$(a \oplus 1) \ominus a = 0 \qquad (4.2.1)$$

se está en el intervalo $[\beta^n, \beta^{n+1}]$ en consecuencia n es el número de operaciones para que se cumpliera por primera vez. (4.2.1.)

Gentleman y Marovich encontraron una ligera falla del algoritmo para ciertas computadoras e hicieron la siguiente observación:
En el algoritmo de Malcom se compara el valor de expresiones aritméticas con el valor de una variable tales como:

```
IF ( (A+B) .EQ. A) ....
ó
IF ( A+ FLOAT ( BETA-1) .EQ. A ) ...
```

ya que la primera se encuentra en la unidad aritmética y la segunda en la memoria, se da el caso, que se representen de manera diferente y esto ocasiona problemas. Para evitar esto se debe almacenar el valor de la expresión aritmética en la memoria, esto se hace mediante un COMMON.

A continuación se presentan los resultados obtenidos en las computadoras VAX/VMS y SEL 32/7780.

Al lector interesado en los listados en FORTRAN-IV plus del algoritmo presentado escriba directamente a los autores.

Sistema VAX/VMS

Aritmética de punto flotante en Precisión Simple

Base del sistema = 2

Dígitos de la mantisa = 24

Tipo de error = Redondeo

Epsilon de la máquina = 5.9604644775E-08

Sistema VAX/VMS

Aritmética de punto flotante en Precisión Doble

Base del sistema = 2

Dígitos de la mantisa = 56

Tipo de error = Redondeo

Epsilon de la máquina = 1.3877787808E-17

Sistema SEL 32/7780

Aritmética de punto flotante en Precisión Simple

Base del sistema = 16

Dígitos de la mantisa = 6

Tipo de error = Redondeo

Epsilon de la máquina = 4.7683715820E-07

Sistema SEL 32/7780

Aritmética de punto flotante en Precisión Doble

Base del sistema = 16

Dígitos de la mantisa = 14

Tipo de error = Truncamiento

Epsilon de la máquina = 2.2204459224E-16

BIBLIOGRAFIA.

a) Libros.

Henrici Peter - Elementos de Análisis Numérico, ed. Trillas, 1977.

S.D. Conte - Carl de Boor - Análisis Numérico, ed. Mc Graw Hill, 1974.

b) Artículos.

1. Gentleman y Marovich - More on algorithmic that reveal properties of floating-point arithmetic units. Comm. ACM 17,5 pgs. 276-277, 1974.

2. Malcom, M.A. - Algorithms to reveal properties of floating - point arithmetic. Comm. ACM 15, 11, pag. 949-951, 1972.