***INSTITUTO POLITÉCNICO NACIONAL ***

***ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA***

***INGENIERÍA ELÉCTRICA***

*SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN*

***“DISEÑO E IMPLEMENTACIÓN DE UN PROGRAMA PARA EL CALCULO DE eA”***

**TAREA 2**

***ALUMNO:***

***SEBASTIÁN CÁRDENAS DAVID JONATHAN***

***PROFESOR:***

***DR. DAVID ROMERO ROMERO***

**MÉXICO DF 04 OCTUBRE DE 2012**

**Formato y precisión de IEEE 754-2008**

El estándar (IEEE 754) es el estándar para el cálculo en punto flotante en las computadoras modernas. El estándar define formatos para la representación de números en punto flotante

IEEE 754 especifica tres formatos para la representación de valores en punto flotante: precisión simple (32 bits), precisión doble (64 bits), precisión cuádruple (128 bits) [1].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre IEEE | Nombre Coloquial | Dígitos binarios | Dígitos Decimales | Exponente Máximo |
| binary32 | Precisión Simple | 23+1 | 7.22 | 38.23 |
| binary64 | Precisión Doble | 52+1 | 15.95 | 307.95 |
| binary128 | Precisión  Cuádruple | 112+1 | 34.02 | 4931.77 |

El estándar IEEE fue establecido en 1985 como un estándar para aritmética de punto flotante, antes de cada fabricante tenia su propio modelo de punto flotante como el IBM 360, el cual también usaba 32 bits en precisión sencilla pero tenia rangos distintos de operación [2].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre IBM | Nombre Coloquial | Dígitos binarios | Dígitos Decimales | Exponente Mínimo | Exponente Máximo |
| binary32 | Precisión Simple | 24+1 | 6 | -79 | 75 |

**Discrepancias entre el algebra y la unidad de punto flotante**

Aunque se dan por hecho algunas leyes del algebra, en las unidades de punto flotante debido a su cantidad finita de dígitos se tiene que estas leyes no se cumplen, o existen errores de redondeo a continuación se describen algunas de ellas con ejemplos en binary32 [3]

*Ley de asociatividad*

(a + b) + c = a + (b + c) no siempre se cumple

a = 1234.567, b = 45.67834, c = 0.0004

(a + b) + c:

1280.24534 se redondea a 1280.245

a + (b + c):

1280.24574 se redondea a 1280.246 <---

*Ley de Distribuitividad*

(a + b) x c = a x c + b x c no siempre se cumple

*Y un ejemplos de redondeo*

A=2000000.0 B=.999999 C=1.0

*Algebra tradicional*

(A+B)\*C=2000000.999999

*Unidad de Punto Flotante*

(A+B)= 2000000.0\*C=2000000.0

**Serie de Taylor**

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Donde n tiende a infinito

**Formula de Tradicional**

Usando la característica de que cada término de la serie es una multiplicación de A por el producto anterior (Ecuación 2) y con el fin de reducir el tiempo de cálculo se obtienen las ecuaciones generales de solución por computadora

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

*Elementos de la iteración*

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

Donde x tiende a infinito o hasta que sea inferior al deseado

**Formula Considerando Unidad de Punto Flotante**

Las ecuaciones desarrolladas por el método tradicional son buenas hasta que se considera que la unidad de punto flotante tiene una cantidad finita de dígitos. Por lo que se obtienen nuevas ecuaciones de solución por computadora considerando el hardware. Es de notar que en la ecuación 16 las expresiones son algebraicamente iguales, sin embargo en la unidad de punto flotante no lo son y habrá que elegir la de menor ruido.

Condiciones iniciales

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

*Elementos de la iteración*

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

**Diseño de la función de tolerancia “Tol”**

Existen diversos métodos para el calculo dentro de los métodos numéricos, uno de ellos es la desviación media definida como

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

Usando una modificación de la formula para considerar todos los elementos de la matriz

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

Donde *in* e *ij* son el número de filas y columnas respectivamente.

Por lo que el error total de la matriz es

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

**Diseño de la de la función de cifras significativas**

Considerando que el estándar **IEEE 754-2008** considera un número finito de cifras significativas reales y que Fortran busca representar mas dígitos mediante redondeos sucesivos se limito la cantidad de dígitos mostrados en pantalla a la cantidad de cifras definidas por el estándar sustituyendo todas las cifras subsecuentes por cero.

Formato Fortan real\*4

-0.735758758144753079636047750792090252

Formato mostrado en pantalla para real\*4

-0.735758800000000000000000000000000000

**Análisis de resultados**

Para el desarrollo de este ejemplo se consideraron los formatos **IEEE** **754-2008** binario 32 para short, binario 64 para doublé, binario 128 para cuádruple. Dentro del estándar fortran 95 usa el tipo de datos real\*4 para hacer referencia al binario32, real\*8 para hacer referencia al binario64 y real\*16 para hacer referencia al binario128

***Calculo de Considerado real***

Usando el método por eigenvalores y eigenvectores se obtuvo como resultado en binario128

-**0.7357587581447530796360477507920902**00 +0.**5518190996580977006157857604767754**00

-**1.471517599088260534975428694604734**000 +**1.103638240715572589083238117463828**000

***DATOS EN VERDE****: CIFRAS SIGNIFICATIVAS*

Comparando el resultado obtenido por el mismo método en binario64

**-0.7357587**76444260500000000000000000000 **+0.551819**113382729000000000000000000000

**-1.471517**635687277000000000000000000000 **+1.1036382**68164836000000000000000000000

***DATOS EN ROJO****: CIFRAS SIGNIFICATIVAS IGUALES A SOL. POR EIGENVALORES EN BINARIO128*

**Calculo de binario128**

*Por método tradicional*

**-0.735758758144753079636047750**354124500 **+0.551819099658097700615785760**355127900

**-1.47151759908826053497542869**2687048000 **+1.103638240715572589083238117**594390000

***DATOS EN ROJO****: CIFRAS SIGNIFICATIVAS IGUALES A SOL. POR EIGENVALORES EN BINARIO128*

Tolerancia 1e-34, 101 iteraciones

*Por método* FPAWARE

**-0.735758758144753079636047751**052681700 **+0.55181909965809770061578576**1119319100

**-1.471517599088260534975428694**001552000 **+1.103638240715572589083238117**647163000

***DATOS EN ROJO****: CIFRAS SIGNIFICATIVAS IGUALES A SOL. POR EIGENVALORES EN BINARIO128*

Tolerancia 1e-34, 101 iteraciones

**Calculo de binario64**

*Por método tradicional*

-**0.73575875**7930780900000000000000000000 +**0.55181909**7752925300000000000000000000

-**1.47151759**8511528000000000000000000000 +**1.103638241**171410000000000000000000000

***DATOS EN ROJO****: CIFRAS SIGNIFICATIVAS IGUALES A SOL. POR EIGENVALORES EN BINARIO128*

Tolerancia 1e-15, 74 iteraciones

*Por método* FPAWARE

**-0.73575876**2783774500000000000000000000 +**0.5518191**03125155500000000000000000000

**-1.4715176**03122790000000000000000000000 +**1.10363824**6380988000000000000000000000

***DATOS EN ROJO****: CIFRAS SIGNIFICATIVAS IGUALES A SOL. POR EIGENVALORES EN BINARIO128*

Tolerancia 1e-15, 74 iteraciones

**Calculo de binario32**

*Por método tradicional*

Causa una excepción al calcular un elemento de matriz que es demasiado grande para ser representado por un binario32

***DATOS EN ROJO****: CIFRAS SIGNIFICATIVAS IGUALES A SOL. POR EIGENVALORES EN BINARIO128*

Tolerancia 1e-7, 30 iteraciones

*Por método* FPAWARE con

+0.557182100000000000000000000000000000 -0.386570700000000000000000000000000000

-1.734312000000000000000000000000000000 +1.396859000000000000000000000000000000

Tolerancia 1e-7, 60 iteraciones, los resultados no se parecen a los esperados

*Por método* FPAWARE con

-1.534295000000000000000000000000000000 +1.278934000000000000000000000000000000

-0.191609200000000000000000000000000000 +0.351672100000000000000000000000000000

Tolerancia 1e-7, 60 iteraciones, los resultados no se parecen a los esperados

**Comportamiento del cambio de la matriz**

Iteraciones

A

**Conclusiones**

Como puede observarse en el análisis de resultados al usar una precisión mayor durante el calculo se obtendrá una mayor cantidad de cifras consideradas correctas, además de esto, una mayor precisión permite establecer tolerancias mas bajas, sin embargo esta tolerancia no debe de ser menor a la cantidad de cifras significativas que puede manejar el tipo de dato, pues nunca se podría comparar esa precisión.

Además de esto el usar una precisión mas alta permite al usuario olvidarse de que l a unidad de punto flotante es de carácter finito y permite hacer uso de la leyes del algebra, esto puede observarse en los métodos tradicionales y FPAWARE de 64 y 128 bits, sin embargo resulta obvio que al bajar la precisión se tienen que analizar el efecto del redondeo, tal es el caso de las funciones FPAWARE de 32 bits que arrojan resultados incorrectos entre si , aunque matemáticamente las formulas sean idénticas

# Bibliografía

|  |  |
| --- | --- |
| [1] | ANSI/IEEE, «IEEE Standard for Binary Floating-Point Arithmetic.,» ANSI/IEEE, New York, 1998. |
| [2] | S. F. E. J. G. Anderson, R. E. Goldschmidt y D. M. Powers, «The IBM System/360 Model 91: Floating-Point Execution Unit,» *IBM Journal of Research and Development,* vol. 11, nº 1, pp. 34-53, 1967. |
| [3] | J. L. H. David A. Patterson, Computer Architecture, Morgan Kaufmann, 2006. |