Práctica 1, (Errores)
Métodos numéricos
2n curso del grado en Matemáticas

Fecha de Reporte: **10/03/2019**
Alumno: *Graells Ricardo, Marc*
(NIU: *1388471*)

0. Resumen

En el presente documento se comentan las soluciones a los ejercicios propuestos en las sesiones de *Seminarios de Métodos Numéricos* de los días 18 y 25 de febrero; y 4 de marzo. Los ficheros `.c` con las implementaciones del código en lenguaje C se adjuntan tal como se detalla en documento en el mismo subdirectorio `README.txt`.

1. Primer ejercicio

1.1. Apartado a

Implementando el programa en lenguaje C obtenemos los resultados que aparecen en Tabla 1. Si pensamos en la función de una variable real definida a trozos del enunciado esperamos que esta sea continua. Esto se podemos ver ya que el limite en cero de la función es un medio (como sabemos por infinitesimales). Por tanto al ser continua en las proximidades debería tomar valores similares pero en el caso de la opción `float` obtenemos un valor extraño, cero.

Si lo pensamos esto tiene sentido ya que la función `fcos()` [1] que trabaja con precisión `float` al ser x muy próximo a cero el resultado de `fcos(x)` será equivalente a `fcos(0)` ≈ 1. Por tanto tendríamos cero entre un valor pequeño que seria de todos modos cero, comprobando que en esté caso un ahorro en la precisión tendría consecuencias fatales en los resultados. Este es un caso típico de **error de cancelación**.

También podemos observar que la precisión `double` da aparentemente un buen resultado al ser este muy próximo al esperado pero sin llegar en ningún caso al valor de un medio, ja que esto no puede ser como sabemos analíticamente, por lo menos de forma exacta.

1.2. Apartado b

Si utilizamos identidad trigonométrica del seno cuadrado[2] para el coseno podemos rescribir f como

$$f(x) = \begin{cases} \frac{1-\cos(x)}{x^2} & \text{si } x \neq 0, \\ \frac{1}{2} & \text{si } x = 0, \end{cases} \quad = \begin{cases} \frac{2\sin^2\left(\frac{x}{2}\right)}{x^2} & \text{si } x \neq 0, \\ \frac{1}{2} & \text{si } x = 0, \end{cases}$$

[1] Se debe destacar que usamos `cosf()` y `cos()` para la función cos(), ya que estas tienen precisiones diferente; simple y doble respectivamente.

[2] Recordemos la igualdad: $\sin^2(x) = \frac{1-\cos(2x)}{2}$

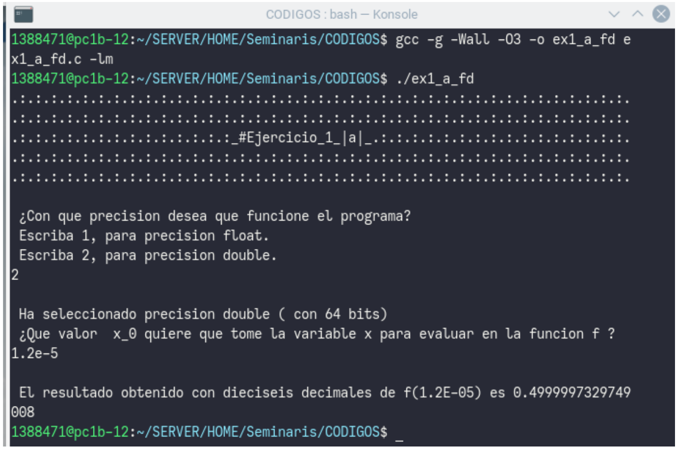

Figura 1: Vista de la terminal con el programa `ex1_a_fd` ejecutado

Con esta mejora tal como vemos en Tabla 1 se corrige el error de cancelación fatal en `float` y se mejora la aproximación en `double`.[3] Recordemos que 0.5 no es mejor aproximación, analíticamente sabemos que nuca sé llega al valor de un medio $\forall x \in \mathbb{R}$.

Programa	`float`	`double`
`ex1_a_fd`	0.0000000	0.49999973297490
`ex1_b_fd`	0.5000000	0.49999999999400

Tabla 1: Resultados obtenidos para las dos formulaciones de $f(x)$.

2. Segundo ejercicio

2.1. Apartado a y b

Demostración. Suponemos que $b^2 >> 4ac$, para la formula con +, entonces:

$$x_+ = \frac{-b+\sqrt{b^2-4ac}}{2a} \approx \frac{-b+\sqrt{b^2}}{2a} = \frac{-b+b}{2a} = 0$$

Observamos un claro error de cancelación. QED

La alternativa se basa en intentar eliminar el error de cancelación cambiando la expresión a calcular mediante manipulaciones algebraicas, en concreto multiplicar por el conjugado:

$$x_+ = \frac{-b+\sqrt{b^2-4ac}}{2a} \cdot \frac{-b-\sqrt{b^2-4ac}}{-b-\sqrt{b^2-4ac}} = \frac{2c}{-b-\sqrt{b^2-4ac}}$$

[3]A lo largo de la práctica se cogen 7 y 14 cifras significativas, para `IEEE simple` y `IEEE double` respectivamente. Esto tiene motivo de ser en que las precisiones respectivas son $2^{-23} < 10^{-7}$ y $2^{-52} < 10^{-14}$. Considerando el peor caso en que la maquina redondea y por tanto tiene peor precisión.

2.2. Apartado c

Probamos con dos casos similar al propuesto por el apartado **a**, $x^2 + 10000x + 1 = 0$ y $0.025x^2 + 70000x + 37 = 0$, ya que $(10000)^2 << 4$ y $(70000)^2 << 3.7$ respectivamente. En la

Programa	Ecuación	`float`	`double`
`ex2_a_fd`	$x^2 - x - 1 = 0$	$\begin{cases} x_+ &= 1.6180340, \\ x_- &= -0.6180340 \end{cases}$	$\begin{cases} x_+ &= 1.6180339887499, \\ x_- &= -0.61803398874989 \end{cases}$
	$x^2 + 10000x + 1 = 0$	$\begin{cases} x_+ &= 0.0000000, \\ x_- &= -1.00000 \times 10^4 \end{cases}$	$\begin{cases} x_+ &= -0.0001000000011, \\ x_- &= -9999.9999 \end{cases}$
	$0.025x^2 + 70000x + 37 = 0$	$\begin{cases} x_+ &= 0.0000000, \\ x_- &= -2.80000 \times 10^6 \end{cases}$	$\begin{cases} x_+ &= -0.52857154514 \times 10^{-3}, \\ x_- &= -2.79999999994 \times 10^6 \end{cases}$
`ex2_b_fd`	$x^2 - x - 1 = 0$	$\begin{cases} x_+ &= 1.6180340, \\ x_- &= -0.6180340 \end{cases}$	$\begin{cases} x_+ &= 1.6180339887499, \\ x_- &= -0.61803398874989 \end{cases}$
	$x^2 + 10000x + 1 = 0$	$\begin{cases} x_+ &= -0.0001000, \\ x_- &= 1.0000000 \times 10^4 \end{cases}$	$\begin{cases} x_+ &= -0.0001000000011, \\ x_- &= -0.99999999 \times 10^4 \end{cases}$
	$0.025x^2 + 70000x + 37 = 0$	$\begin{cases} x_+ &= -0.5286 \times 10^{-3}, \\ x_- &= -2.80000000 \times 10^6 \end{cases}$	$\begin{cases} x_+ &= -0.52857142867 \times 10^{-3}, \\ x_- &= -2.7999999994 \times 10^6 \end{cases}$

Tabla 2: Resultados obtenidos para las dos variaciones de los programas. Observamos que se corrigen (en verde) los errores de cancelación(en rojo). El primer ejemplo es solo de referencia, como solución conocida.

Tabla 2 se muestran los errores de cancelación y su corrección mediante la variación de la fórmula empleada para el cálculo. También cabe destacar que el código en `C` implementado aún puede fallar si $a \approx 0$ (este caso no lo consideramos por enunciado,ecuación de primer grado) en `ex3_a_fd`. En `ex2_b_fd` si $c \approx 0$ y $a \neq 0$ también anticipamos un mal resultado, que sé podría solucionar con un `if`. El caso de que el polinomio solo tenga raíces complejas el programa (en las dos variaciones) imprime un mensaje de error.

3. Tercer ejercicio

3.1. Apartado a y b

En este caso cabe destacar que el código en `C`, lee directamente de un fichero y retorna el resultado de **(3)** y **(4)** de la hoja de enunciados. Observando los resultados de evaluar el vector $x = \{10000, 10001, 10002\}^T$ en la Tabla 3 nos encontramos con un nuevo error de cancelación en la fórmula (4), es decir, para $s_n^2 = \frac{1}{n-1}\left(\sum_{i=1}^n x_i^2 - \frac{1}{n}\left(\sum_{i=1}^n x_i\right)^2\right)$.

3.2. Apartado c y d

Implementando en `C` podemos hacer una lista de valores tan elaborada y extensa como queramos. Pero consideremos los ejemplos `vector_2.txt` y `vector_3.txt` generados

Fórmula	`float`	`double`
(3)	1.0000000	1
(4)	0.0000000	1

Tabla 3: Resultados obtenidos para las dos formulaciones diferentes con el programa `ex3_a_fd` para el vector $x = \{10000, 10001, 10002\}^T$.

por `ex3_a_fd.c`

$$L_{1\le i\le 100} = \begin{cases} 10.000.100 \text{ si } i \le 50, \\ 10.000.000 \text{ si } i > 50 \end{cases} \quad \text{y} \quad H_{1\le i\le 1000} = \begin{cases} 1.000.000.100 \text{ si } i \le 500, \\ 1.000.000.000 \text{ si } i > 500 \end{cases}$$

Los resultados para L son `float`: 50.2920761 y 8713.2734375, para `double`: 50.251890762961 en los dos caos; para fórmulas (3) (4) respectivamente. Los resultados para H son `float`: 9540.9863281 y $-NAN$, para `double`: 50.025018765639 y 59.518791036835; para fórmulas (3) (4) respectivamente. El valor esperado por construcción de los ejemplos es el mismo y es aproximadamente 50. El valor -NAN indica que sé ha intentado calcular la raíz de un numero negativo originado por error de cancelación. Concluimos que (4) es **numéricamente inestable**.

4. Cuarto ejercicio

4.1. Apartado a, b y c

Implementamos en `C` un nuevo programa que imprime la suma parcial enésima procediendo de forma creciente y decreciente con precisión simple o doble. Los resultados se muestran en Tabla 4. Observamos que con precisión simple, es decir con el uso de `float` , el resultado mejora si se suma en orden decreciente(comenzando por termino enésimo). Con las dos precisiones vemos que al aumentar n la diferencia disminuye, en `float` la mejora colapsa en n=10000 o antes, en `double` mejora hasta el máximo permitido por el programa realizado; con un total de 9 decimales correctos. Estás diferencias se explican ja que la suma en **aritmética de punto flotante** no es asociativa, al igual que las otras operaciones.

4.2. Apartado d

Consideramos la siguiente serie que sabemos por criterio de Leibniz que es convergente y procedemos a manipular-la: $2\cdot\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n^2} = \sum_{n=1}^{\infty}\frac{2}{n^2} + \sum_{n=1}^{\infty}\frac{-2}{(2n)^2} = \sum_{n=1}^{\infty}\frac{4-2}{2\cdot n^2} = \sum_{n=1}^{\infty}\frac{2}{2\cdot n^2}$. Probada la igualdad implementamos en `C` la nueva re-formulación de la serie en los dos ordenes (creciente y decreciente). Compilamos y ejecutamos la nueva versión del código `ex4_abd_fd.c` y obtenemos: para `float` y 5000 términos la diferencia queda por 10^{-7}, llegando al máximo de precisión para `float`. Para `double` con 5000 términos tenemos $d_{\triangle} \approx d_{\triangledown} = 3.999\cdot 10^{-8}$ y;con 10000 $d_{\triangle} \approx d_{\triangledown} = 3.999\cdot 10^{-9}$; con $n = 100000$ $d_{\triangle} \approx d_{\triangledown} = 3.999\cdot 10^{-10}$.

S_n	`float`	diferencia
n=5000	$\begin{cases} x_\triangle &= 1.6447253, \\ x_\triangledown &= 1.6447340 \end{cases}$	$\begin{cases} d_\triangle &\approx 0.0002087, \\ d_\triangledown &\approx 0.0002000 \end{cases}$
n=10000	$\begin{cases} x_\triangle &= 1.6447253, \\ x_\triangledown &= 1.6448340 \end{cases}$	$\begin{cases} d_\triangle &\approx 0.0002087, \\ d_\triangledown &\approx 0.0001000 \end{cases}$
n=100000	$\begin{cases} x_\triangle &= 1.6447253, \\ x_\triangledown &= 1.6448340 \end{cases}$	$\begin{cases} d_\triangle &\approx 0.0002087, \\ d_\triangledown &\approx 0.0001000 \end{cases}$
S_n	`double`	diferencia
n=5000	$\begin{cases} x_\triangle &= 1.6447340868469, \\ x_\triangledown &= 1.6447340868469 \end{cases}$	$\begin{cases} d_\triangle &\approx 1.99980 \times 10^{-4}, \\ d_\triangledown &\approx 1.99980 \times 10^{-4} \end{cases}$
n=10000	$\begin{cases} x_\triangle &= 1.6448340718481, \\ x_\triangledown &= 1.6448340718481 \end{cases}$	$\begin{cases} d_\triangle &\approx 9.99950 \times 10^{-5}, \\ d_\triangledown &\approx 9.99950 \times 10^{-5} \end{cases}$
n=100000	$\begin{cases} x_\triangle &= 1.6449240668982, \\ x_\triangledown &= 1.6449240668982 \end{cases}$	$\begin{cases} d_\triangle &\approx 9.99994 \times 10^{-6}, \\ d_\triangledown &\approx 9.99994 \times 10^{-6} \end{cases}$
n=4.294.967.295 máximo `unsigned int`	$\begin{cases} x_\triangle &= 1.6449340578346, \\ x_\triangledown &= 1.6449340666154 \end{cases}$	$\begin{cases} d_\triangle &\approx 9.01365 \times 10^{-9}, \\ d_\triangledown &\approx 2.32830 \times 10^{-10} \end{cases}$

Tabla 4: Resultados obtenidos con `ex4_ab_fd`. Notación: $\triangle$ indica orden creciente, $\triangledown$ indica orden decreciente. x_{orden} indica valor de la suma, d_{orden} indica diferencia con valor de enunciado (7 o 14 decimales).

Y finalmente en poco más de 2 minutos con n=4.294.967.295 conseguimos 15 decimales correctos con $d_\triangledown \approx 4.44 \cdot 10^{-16}$. Obtenemos $x_\triangledown = 1.644934066848226$. [4]

5. Quinto ejercicio

5.1. Apartado a, b y c

Demostración. Sea $p = \frac{(a+b+c)}{2}$ y $A = \sqrt{p(p-a)(p-b)(p-c)}$. Suponemos que $a \approx b+c$, entonces:

$$A = \sqrt{p\frac{(a+b+c-2a)}{2}(p-b)(p-c)} \approx \sqrt{\frac{p}{2}(2a-2a)(p-b)(p-c)} = 0$$

Observamos un claro error de cancelación suponiendo condiciones. QED

Si consideramos[5] [6] y sin el cumplimiento de la propiedad asociativa, con $a \leq b \leq c$. Por Ejemplo $T_1 = (10^8, 10^8, 0.01)$ o $T_2 = (10^7, 10^7, 2)$ Si introducimos estos valores en `ex5_bc_fd` obtenemos para fórmula inicial 0.0000000 y 499999.52, para fórmula alternativa 500000.00 y 499999.99; para `float` y `double` respectivamente. Observamos que se corrigen (en verde) los errores de cancelación (en rojo).

[4] Mirar `Resultados_4d.txt`

[5] $A = \frac{1}{4}\sqrt{(a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c)}$

[6] Nótese que con este *orden* evitamos que las operaciones generen un error de cancelación, es la fórmula de Heron re-adaptada.