Práctico 1 - Punto flotante y errores

El Ejercicio 6 es el "entregable" de este práctico. Todo estudiante inscripto en el curso va a tener asignado un ejercicio del práctico 1, 2, o 3 para entregar el 8 de octubre a las 23:59. **El 6 de octubre vamos a avisar qué ejercicio le corresponde a cada estudiante, por lo que recomendamos fuertemente haber terminado y tener escrito el Ejercicio 6 para esa fecha.**

La primer entrega de ejercicio tiene un puntaje máximo de 15 puntos y la segunda un máximo de 20 puntos. Para la aprobación del curso se necesita un mínimo de 15 puntos en las entregas de ejercicios y 60 puntos en el total (cuestionarios+ejercicios+obligatorio).

Punto flotante

Ejercicio 1 (Constantes de calculadora)**.** Encontrar experimentalmente los siguientes valores de tu calculadora:

a) El valor ε_M.

b) El mayor número representable.

c) El menor número positivo representable.

Ejercicio 2 (Precisión simple)**.** El formato de precisión simple es uno de 32 bits. Los números en dicho formato están definidos por

$$x = \pm(1+f)\cdot 2^e, \qquad (*)$$

donde

- el signo ocupa 1 bit;
- la mantisa ocupa 23 bits, esto es, $0 \leq f < 1$, y $2^{23}f$ es un número natural;
- el exponente ocupa 8 bits, esto es, $-126 \leq e \leq 127$.

¿Cuál es ε_M para este formato? ¿Cuáles son el número mayor y menor (en valor absoluto) que pueden ser representados con $(*)$?
Se considera $g\colon \mathbb{R} \to \mathbb{R}$,

$$g(x) = \frac{x^3}{x - \operatorname{sen}(x)},$$

y se toma $x = 5 \cdot 10^{-4}$. Computar $g(x)$ en Octave.
Por defecto, Octave trabaja con números de precisión doble; el comando `single` convierte una variable a precisión simple. Definir `y=single(x)` y computar $g(y)$ en Octave. ¿Qué resultado se obtiene? Explicar qué está ocurriendo.

Ejercicio 3 (Problema 1.35 en Moler)**.** ¿Qué realizan cada uno de estos programas? ¿Cuántas líneas de salida devuelve cada uno de los programas? ¿Cuáles son los últimos dos valores de `x` que muestran?

- `x = 1;  while 1+x > 1, x = x/2, pause(.02), end`
- `x = 1; while x+x > x, x = 2*x, pause(.02), end`
- `x = 1; while x+x > x, x = x/2, pause(.02), end`

Errores

Ejercicio 4 (Errores relativo y absoluto)**.**

a) Al determinar una constante C se obtuvo el valor 92,34 con un error relativo de un 0,1 %. ¿En qué intervalo se encuentra C? ¿Cuál es el error absoluto?

b) ¿Cuántos dígitos del número $\sqrt{22}$ deben darse para determinarlo con un error relativo no mayor al 0.1 %?

c) En una medición se obtiene el valor $v = 17261$. Se sabe que el error relativo es del 1 %. ¿Cómo debería escribirse v para reflejar este hecho?

Ejercicio 5 (Errores en operaciones)**.**

a) El diámetro interior de un tanque de agua esférico es de $1{,}5 \pm 0{,}05$ metros. Calcular su volumen (con el error correspondiente) aproximando $\pi \simeq 3{,}1416$.

b) Un campo rectangular mide aproximadamente 2000 por 3000 metros. ¿Con qué error deberían medirse los lados para obtener el área con un error inferior a un metro cuadrado?

Ejercicio 6 (Aproximación de la derivada con un cociente incremental)**.** Dada una función $f : I \to \mathbb{R}$ de clase C^∞, donde $I \subseteq \mathbb{R}$ es un intervalo, se desea calcular la derivada $f'(a)$ en un punto $a \in I$ usando un *cociente incremental centrado*,

$$\delta_a(h) = \frac{f(a+h) - f(a-h)}{2h}.$$

a) Usar un desarrollo de Taylor para estimar el error $|\delta_a(h) - f'(a)|$.

b) Aproximar la derivada de la función $f(x) = \operatorname{sen}(5x)$ en el punto $a = 1$, con $\delta_a(h)$ y usando $h = 0{,}1^k$ con $k = 0, 1, \ldots, 20$. Graficar, usando escala logarítmica (función `loglog` en Octave), el error absoluto cometido en función de h. Explicar el comportamiento observado.

c) Usando los resultados vistos en clase sobre los errores de truncamiento y de redondeo, estimar el valor de h óptimo para el cálculo anterior. Cotejar este valor de h con el resultado obtenido en la parte anterior.

Ejercicio 7 (Algoritmo babilonio)**.** Se desea estimar $\sqrt{2}$ con una alta cantidad de acierto en dígitos decimales. Para esto se propone la sucesión $\{a_n\}_{n\geq 0}$ tal que $a_0 = 2$ y

$$a_{n+1} = \frac{a_n}{2} + \frac{1}{a_n}.$$

a) Demostrar por inducción completa que $a_n > 0 \ \forall n \in \mathbb{N}$.

b) Probar que a_n está acotada inferiormente por $\sqrt{2}$ y que es decreciente. Concluir que a_n es convergente y que $\lim_n a_n = \sqrt{2}$.

c) Se considera la sucesión de errores absolutos (en magnitud) $e_n = |a_n - \sqrt{2}|$. Buscar una constante $0 < C < 1$ tal que $e_{n+1} \geq C e_n^2$. Notar que esto implica que el error decrece al menos en forma cuadrática de una iteración a otra.

d) Demostrar por inducción que $a_n \in \mathbb{Q} \ \forall n \in \mathbb{N}$. Por lo tanto, $a_n \neq \sqrt{2} \ \forall n \in \mathbb{N}$.

e) Implementar un programa que calcule a_n usando la recursión inicial pero escrita en la forma:

$$a_{n+1} = \left(\frac{a_n^2 + 2}{2}\right) \cdot \frac{1}{a_n}.$$

Con este programa:

i) Analizar la evolución del error e_n.

ii) Hallar el primer entero positivo n_0 tal que la máquina confunde a_{n_0} con $\sqrt{2}$.

Ejercicio 8 (Cancelación catastrófica y desborde)**.**

a) Se desea calcular numéricamente $\lim\limits_{n \to +\infty} \int_n^{n+1} \log(x)\, dx$. ¿Cómo puede reescribirse dicha integral para evitar efectos de cancelación catastrófica?

b) Reescribir la expresión $\frac{e^x}{e^x+1}$ para poder evaluarla en valores grandes de x evitando efectos de desborde.

c) Comentar los inconvenientes que pueden surgir al implementar un programa para calcular la derivada de $\cos(x)$ utilizando el cociente incremental $\frac{\cos(x+h)-\cos(x)}{h}$. ¿Cómo se podría reescribir dicho cociente para aproximar esta derivada de forma más estable?

Ejercicio 9 (Una recurrencia inestable)**.** Los números $p_n = \int_0^1 x^n e^x dx$ satisfacen $p_1 > p_2 > \ldots > 0$ y la relación de recurrencia (donde $e = 2,718\ldots$ es el número de Euler)

$$p_{n+1} = e - (n+1)\, p_n, \qquad p_1 = 1.$$

a) Demostrar la relación de recurrencia.

b) Escribir un programa de Octave que genere los primeros 20 valores de p_n y explicar por qué los resultados obtenidos no satisfacen la desigualdad de arriba.

c) Tomar $p_{20} = 1/8$ y usar la relación de recurrencia para computar $p_{19}, \ldots, p_1$. Los números obtenidos, ¿satisfacen las desigualdades $1 = p_1 > p_2 > \ldots > 0$? Explicar la diferencia entre los dos procedimientos en términos de su estabilidad numérica. Repetir con $p_{20} = 20$ y $p_{20} = 100$, y explicar qué ocurre y por qué.

Ejercicio 10 (Extrapolación de Richardson)**.** La Extrapolación de Richardson permite mejorar el orden del error de truncamiento de una estimación numérica. Supongamos que p^* es un número a estimar y tenemos una estimación $p_0(h)$ que cumple

$$p^* = p_0(h) + a_0 h^{k_0} + a_1 h^{k_1} + \dots, \tag{1}$$

donde las $a_i \in \mathbb{R}$ son constantes desconocidas y las potencias $k_i > 0$ son conocidas y cumplen $k_0 < k_1 < k_2 < \dots$. La identidad de arriba puede ser interpretada como que el error de truncamiento al aproximar p^* mediante $p_0(h)$ es del orden de h^{k_0}. Buscamos ahora reutilizar (1) para producir una nueva aproximación $p_1(h)$ de p^* cuyo error de truncamiento sea del orden de h^{k_1}. Con este fin, dado $t > 0$ consideramos el estimador p_0 evaluado en h/t y utilizamos (1):

$$p^* = p_0(h/t) + a_0 \frac{h^{k_0}}{t^{k_0}} + a_1 \frac{h^{k_1}}{t^{k_1}} + \dots.$$

La extrapolación de Richardson consiste en combinar esta identidad con (1), para obtener un nuevo estimador $p_1(h)$ de p^*,

$$p_1(h) = \alpha p_0(h) + \beta p_0(h/t).$$

de modo que el error de truncamiento al aproximar p^* mediante $p_1(h)$ sea del orden de h^{k_1}, esto es,

$$p^* = p_1(h) + \tilde{a}_1 h^{k_1} + \dots,$$

a) Hallar α y β y verificar que es

$$p_1(h) = \frac{t^{k_0} p_0(h/t) - p_0(h)}{t^{k_0} - 1}.$$

b) Se sabe que $\lim_{h\to 0}(1+h)^{1/h} = e$, y que ésta es una fórmula de primer orden,

$$e = (1+h)^{1/h} + a_0 h + \dots,$$

donde el número a_0 es desconocido. Usar la extrapolación de Richardson con $t = 2$ para obtener una aproximación del número e de segundo orden.

c) Verificar los resultados de la parte anterior en Octave: para $h = 10^{-2}, 10^{-3}, 10^{-4}$, comparar los errores cometidos por la aproximación $e \simeq (1+h)^{1/h}$ con los cometidos al usar el método de la parte anterior.

Ejercicios opcionales

Ejercicio 11 (Derivada con complejos)**.** El objetivo de este ejercicio es mostrar un truco que, si uno puede realizar cómputos con números complejos, permite evaluar derivadas de forma numéricamente estable.

Sea $f : I \to \mathbb{R}$ una función de clase C^∞ (infinitamente derivable), donde $I \subseteq \mathbb{R}$ es un intervalo. Se desea aproximar $f'(a)$.

a) Sea i la unidad imaginaria compleja. Dado $h > 0$, aproximar $f(a + ih)$ mediante un desarrollo de Taylor.

b) Se aproxima $f'(a)$ por

$$\Delta_a(h) = \mathrm{Im}\left(\frac{f(a+ih)}{h}\right),$$

donde Im denota la parte imaginaria. Estimar el error $|\Delta_a(h) - f'(a)|$.

c) Repetir las partes b) y c) del Ejercicio 6 para $\Delta_a(h)$. ¿Qué hace que esta aproximación sea más robusta computacionalmente que la del otro ejercicio?

Ejercicio 12 (Serie exponencial)**.** Se desea calcular los valores de la función exponencial a partir de su desarrollo en serie

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

a) Usar el siguiente programa para efectuar la suma anterior hasta $n = 100$, para un rango de valores de x:

```
x=-20:20;
sum=ones(size(x));
t=x; n=1;
while n<100
    sum=sum+t;
    n=n+1;
    t=t.*x/n;
end
```

b) Investigar qué sucede con el error relativo en los resultados numéricos obtenidos. Usar la función `exp` y grafique con `semilogy`. ¿Dónde se dan los peores resultados? Justificar por qué ocurre esto.

c) Buscar una forma alternativa y más precisa de hacer el cálculo en los valores de x problemáticos en la parte anterior.

Ejercicio 13 (Overflow en sumas)**.**

a) Consideremos el siguiente código para calcular una suma $\sum_{k=1}^{n} x_k$, donde $x_1, \ldots, x_n$ son conocidos:

```
suma = 0;
for k = 1:n
    suma = suma + x(k);
end
```

Explicar en qué casos podría darse que este código dé lugar a problemas computacionales como cancelaciones catastróficas u overflow. Generar un ejemplo en el que se produce cada uno de esos problemas.

b) Consideremos ahora el siguiente código[1]:

[1] Dado un vector x, la primera línea de este código halla $m = \max_k |x_k|$.

```
[~, ~, m] = find(max(abs(x)));
x = x/m;
suma = 0;
for k = 1:n
    suma = suma + x(k);
end
suma = m*suma;
```

Explicar en qué casos y por qué este código puede evitar que se produzca un overflow.

c) Utilizar la idea de la parte anterior para escribir un código que compute la norma euclídea de vectores $\mathtt{x} \in \mathbb{R}^n$ (con n "grande") de forma estable.