Práctico 1 - Punto flotante y errores

Los ejercicios 10 y 12 son los "entregables" de este práctico. Todo estudiante inscripto en el curso va a tener asignado un único ejercicio del práctico 1, 2, o 3 para entregar antes del comienzo del primer período de parciales. **El 19 de setiembre vamos a avisar qué ejercicio le corresponde a cada estudiante, por lo que recomendamos fuertemente haber terminado y tener escrito los ejercicios 10 y 12 para esa fecha.**

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`

Ejercicio 4 (Del examen de febrero de 2024)**.** Consideremos la función $f\colon \mathbb{R} \setminus \{0\} \to \mathbb{R}$,

$$f(x) = \frac{1 - \cos(x)}{x^2}.$$

a) Verificar que $\lim_{x\to 0} f(x) = \frac{1}{2}$. En adelante, consideramos $f\colon \mathbb{R} \to \mathbb{R}$ como arriba, con $f(0) = \frac{1}{2}$ y podemos asumir que $f'(0) = 0$.

b) Calcular el número de condición de f en $x = 0$ y estimar el error inevitable al evaluar $f(x)$ para $x \approx 0$ usando aritmética de punto flotante con precisión doble y redondeo.

c) Se corre el siguiente código de Octave:

```
f = @(x) (1-cos(x))/(x^2);
valores = [f(1e-5) f(1e-6) f(1e-7) f(1e-8) f(1e-9)]
```

y se obtiene la salida

```
valores =

    0.5000    0.5000    0.4996         0         0
```

¿Por qué obtenemos resultados tan malos al evaluar $f(10^{-8})$ y $f(10^{-9})$?

d) ¿Cómo se puede reescribir $f(x)$ para que la se pueda evaluar de forma numéricamente estable con $x \approx 0$?

Errores

Ejercicio 5 (Raíces de polinomio)**.** Hallar las dos raíces del polinomio $x^2 + 5x + 10^{-12}$ con cuatro dígitos de precisión.

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 $e_{n+1} \leq Ce_n^2$. 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 (Otra recurrencia inestable)**.** Dado $n \in \mathbb{N}$, consideremos la integral

$$I_n = \int_0^1 \frac{x^n}{5+x}\, dx.$$

a) Verificar que $I_0 = \log(6/5)$ y que, para todo $n \geq 1$, se tiene $I_n \geq 0$ y se cumple la relación

$$I_n + 5I_{n-1} = \frac{1}{n}.$$

b) Escribir un programa de Octave que implemente la fórmula de recurrencia de la parte anterior y utilizarlo para calcular I_{30}. Analizar el resultado obtenido. ¿Es confiable? Justificar realizando un análisis de error hacia adelante.

c) Como alternativa, se propone comenzar aproximando $I_{100} = 0$ y utilizar la fórmula de recurrencia hacia atrás. Computar I_{30} de esta manera. Probar con otras aproximaciones iniciales para I_{100}, como por ejemplo $I_{100} = 0{,}1$, $I_{100} = 1$. Explicar qué ocurre y por qué.

Ejercicio 11 (Fórmula para el área de un triángulo)**.** El área de un triángulo T con lados a, b, c, se puede calcular mediante la fórmula

$$A(T) = \sqrt{p(p-a)(p-b)(p-c)},$$

donde $p = \frac{a+b+c}{2}$ es la mitad del perímetro de T.

a) Escribir una función de Octave `A = area(a,b,c)` que utilice esta fórmula para calcular el área del triángulo con lados a, b, c. Las entradas `a`, `b`, `c`, deben ser números positivos y tales que cualquiera de ellos es menor que la suma de los otros dos. En caso de que no se cumplan estas condiciones, la función debe arrojar un mensaje de error.

b) Sea $\delta \in (0, 1)$. Consideremos un triángulo rectángulo con catetos $a = 1$, $b = \delta$, e hipotenusa $c = \sqrt{1+\delta^2}$. Usar la función de la parte anterior para calcular el área de dicho triángulo para los valores $\delta = 10^{-n}$, $n = 1, \ldots, 16$. Computar los errores relativos en las áreas calculadas.

c) Repetir para el triángulo isósceles con lados $a = 2$, $b = c = \sqrt{1+\delta^2}$ que se obtiene "pegando" dos triángulos rectángulos de la parte anterior. Explicar la diferencia en los resultados obtenidos en las partes b) y c).

Ejercicio 12 (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 13 (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 14 (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 15 (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.