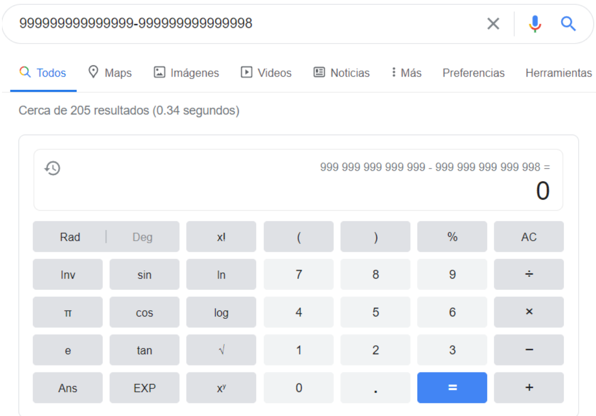
Errores Numéricos

Juan Manuel Rabasedas



Introducción

$$\int_0^1 \sqrt{1-x^2}dx = \frac{\pi}{4} \approx 0.78540$$

- Sistemas de cómputo simbólico
  - Maple, Mathematica, Maxima
  - Simbolos matemátematicos abstractos
  - No existe perdida de precisión mientas no se evalue.
- Sistema de cálculo numérico
  - Scilab, Matlab, Octave
  - Evaluan sus expresiones a números flotantes
  - Los valores, en general, poseen error de redondeo.

Números Reales y Punto Flotante

- Scilab almacena los números reales en punto flotante de doble precisión de 64 bits (IEEE 754)
- Tenemos infinitos no numerables números reales
- Sólo tenemos 2^{64} números difernetes en punto flotante de 64 bits
- Esto produce redondeo, desbordamiento a cero (underflow) y desbordamiento (overflow).
- Scilab tienen un epsilon igual a 2.22×10^{-16} %eps
- Este valor es independiente del hardware y del SO.
- Me indica que en el mejor de los casos tendré casi 16 cifras significativas.

Representaciones

- Los números negativos normalizados en punto flotante se encuentran en el rango $[-10^{308}, -10^{-307}]$
- Los números positivos normalizados están en el rango $[10^{-307}, 10^{308}]$
- Un número mayor que 10^{309} o menor que -10^{309} no es representable como doble, se almacena como `%inf`. **(overflow)**
- Un número menor a 10^{-324} no es representable como doble, se almacena como 0. **(underflow)**

Errores de redondeo

Calculamos el número $0,1$ de dos maneras equivalentes.

```
-->format(25)
-->0.1
 ans =
  0.1
-->1.0 - 0.9
 ans =
  0.0999999999999999780000
-->0.1 == 1.0 - 0.9
 ans =
 F
```


Errores de redondeo

- La representación de `0,1` no es exacta.
- Un número x se representa en punto flotante como

$$fl(x) = M \cdot \beta^{E-s}$$

- En Scilab $\beta = 2$, E es de 11 dígitos, M de 52 dígitos y $s = 1023$
- El número `0,1` se almacena con exponente y mantisa:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011010$
  $(M)_{10} = 1,6$
- Luego la representación es: $fl(0.1) = 1,6 \cdot 2^{1019-1023} = 1,6 \cdot 2^{-4}$
- Calcular en Scilab

```
-->1.6*2^(-4)
 ans =
  0.1
```


Errores de redondeo

- El número en punto flotante anterior al calculado tiene la siguiente mantisa:
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011001$
  $(M)_{10} = 1,5999999999999999$
- Calcular en Scilab

```
-->1.5999999999999999*2^(-4)
 ans =
  0.0999999999999999920000
```

- El número `0,1` exacto se encuentra entre dos números de punto flotante consecutivos:

$$1.5999999999999999 \cdot 2^{-4} < 0.1 < 1.6 \cdot 2^{-4}$$

- Scilab va a representar a $0,1$ con el número más cercano:
  $|0,1 - 1,5999999999999999 \cdot 2^{-4}| = 8,33 \ldots 10^{-18}$
  $|0,1 - 1,6000000000000000 \cdot 2^{-4}| = 5,55 \ldots 10^{-18}$

Errores de redondeo

La función *seno* es también aproximada, podemos calcular en Scilab:

```
-->format(25)
--> v = sin(0.0)
 v =
  0.0
-->a = sin(%pi)
 a =
  0.0
-->v == a
ans =  F
```

- La representación exacta del número π requiere de un número infinito de bits.
- Tenemos un error de representación de π
- También hay error de aproximación de la función seno.

Ruido

```
clc // limpia la consola
clear // borra el contenido de la memoria

// Primera funcion
function y = P1(x)
 y = x.^7 - 7*x.^6 + 21*x.^5 - 35*x.^4 + 35*x.^3 - 21*x.^2 + 7*x - 1;
endfunction
// Segunda función
function y = P2(x)
  y = (x - 1).^7;
endfunction
// Evaluación de ambas funciones cerca de uno
x = linspace(1-1e-2,1+1e-2,2001);
y1 = P1(x);
y2 = P2(x);
// Gráfica de las funciones
plot(x,y1,'b');
plot(x,y2,'r','thickness',2)
legend(["$P1(x)$";"$P2(x)$"]);
```


Ecuación cuadrática

Sean $a, b, c \in \mathbb{R}$ coeficientes dados con $a \neq 0$. Consideremos la siguiente ecuación cuadrática: $ax^2 + bx + c = 0$
El *discriminante* de la ecuación $\Delta = b^2 - 4ac$ determina la índole y la cantidad de raíces.

- Si $\Delta > 0$ hay dos raíces reales y diferentes:

$$x_- = \frac{-b - \sqrt{\Delta}}{2a}, \tag{1}$$

$$x_+ = \frac{-b + \sqrt{\Delta}}{2a}. \tag{2}$$

- Si $\Delta = 0$ hay una raíz real doble:

$$x_\pm = -\frac{b}{2a}. \tag{3}$$

- Si $\Delta < 0$ hay dos raíces complejas conjugadas:

$$x_\pm = \frac{-b}{2a} \pm i\frac{\sqrt{-\Delta}}{2a}. \tag{4}$$

Ecuación cuadrática

```
function r = misraices(p)
  c = coeff(p,0);
  b = coeff(p,1);
  a = coeff(p,2);
  r(1) = (-b + sqrt(b^2-4*a*c))/(2*a);
  r(2) = (-b - sqrt(b^2-4*a*c))/(2*a);
endfunction
```

```
p = poly([-0.0001 10000.0 0.0001],"x","coeff");
e1 = 1e-8;
roots1 = misraices(p);
r1 = roots1(1);
roots2 = roots(p);
r2 = roots2(1);
error1 = abs(r1-e1)/e1;
error2 = abs(r2-e1)/e1;
printf("Esperado :  %e\n", e1);
printf("misraices (nuestro) :  %e (error=%e)\n", r1, error1);
printf("roots (Scilab) :  %e (error=%e)\n", r2, error2);
```

El script anterior produce la siguiente salida.

```
Esperado :  1.000000e-008
misraices (nuestro) :  9.094947e-009 (error = 9.050530e-002)
roots (Scilab) :  1.000000e-008 (error = 0.000000e-000)
```


Método robusto para calcular las raíces.

Supongamos que la ecuación cuadrática con coeficientes reales tiene un discriminante Δ positivo. Luego sabemos que:

$$x_- = \frac{-b - \sqrt{\Delta}}{2a}, \tag{5}$$

$$x_+ = \frac{-b + \sqrt{\Delta}}{2a}. \tag{6}$$

Dividiendo la ecuación cuadrática por $1/x^2$, suponiendo $x \neq 0$, obtenemos:

$$c(1/x)^2 + b(1/x) + a = 0 \tag{7}$$

Las dos raíces reales de la ecuación cuadrática son:

$$x_- = \frac{2c}{-b + \sqrt{b^2 - 4ac}}, \tag{8}$$

$$x_+ = \frac{2c}{-b - \sqrt{b^2 - 4ac}}. \tag{9}$$

Método robusto para calcular las raíces.

Cuando el discriminante Δ es positivo, el problema de supresión de dígitos significativos se puede separar en dos casos:

- Si $b < 0$, luego $-b - \sqrt{b^2 - 4ac}$ puede dar supresión de dígitos ya que $-b$ es positivo y $-\sqrt{b^2 - 4ac}$ es negativo,
- Si $b > 0$, luego $-b + \sqrt{b^2 - 4ac}$ puede dar supresión de dígitos ya que $-b$ es negativo y $\sqrt{b^2 - 4ac}$ es positivo.

Por lo tanto,

- Si $b < 0$ debemos usar la expresión $-b + \sqrt{b^2 - 4ac}$,
- Si $b > 0$ debemos usar la expresión $-b - \sqrt{b^2 - 4ac}$.

Método robusto para calcular las raíces.

- Si $b < 0$

$$x_- = \frac{2c}{-b + \sqrt{b^2 - 4ac}}, \tag{10}$$

$$x_+ = \frac{-b + \sqrt{\Delta}}{2a}. \tag{11}$$

- Si $b > 0$

$$x_- = \frac{-b - \sqrt{\Delta}}{2a}, \tag{12}$$

$$x_+ = \frac{2c}{-b - \sqrt{b^2 - 4ac}}. \tag{13}$$

Derivada numérica

- Para aproximar $f'(x)$ podemos utilizar la expresión en diferencias finitas de primer orden:

$$f'(x) \approxeq \frac{f(x+h) - f(x)}{h}$$

- Sabemos que cuando h tiende a cero, la fórmula nos da el valor exacto de la derivada $f'(x)$
- Podemos esperar que cuanto más pequeño sea h mejor será la aproximación.
- Debido a los errores de redondeo esto no es así.

Derivada numérica

- La función `numderivative` de Scilab permite usar fórmulas de diferencias finitas de órdenes 1, 2, o 4.
- `numderivative(f,x,order)`
- `f`: es una función de scilab
  `x`: es el valor donde se evaluará f'
  `order: se usa para indicar el orden de la fórmula de diferencias finitas`

Ejemplo

```
clc // limpia la consola
clear  // borra el contenido de la memoria
xdel(winsid())  // cierra ventanas graficas

// Definicion de la funcion
function y = f(x)
  y = x.*x;
endfunction

// Calculo de la derivada utilizando diferencias finitas
function y = dfa(f,x,h)
  y = (f(x+h) - f(x))./h;
endfunction

x = 1; // Punto donde vamos a evaluar la derivada
ih = (0:16)';
h = (10.^-ih); // Vector con los valores de h

df_approx = dfa(f,x,h); // Evaluación de la derivada por diferencias finitas
df_scilab = numderivative(f,x,order=1); // Derivada obtenida por numderivative
df_true = 2; // Valor verdadero de la derivada en x = 1
```

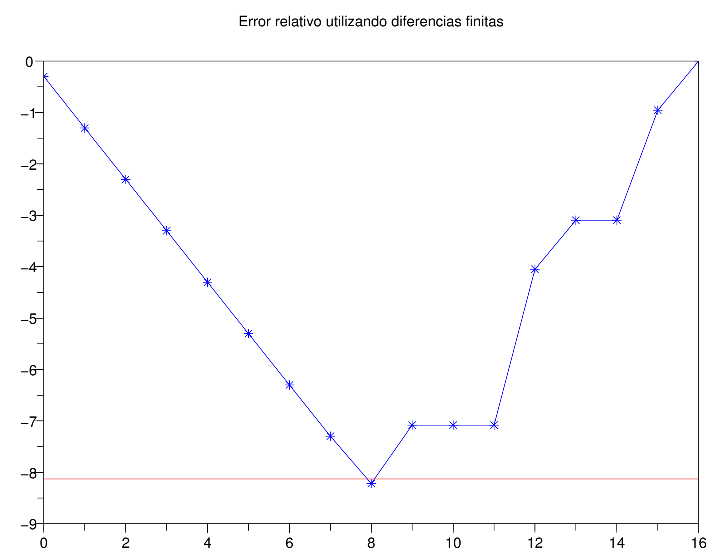

Ejemplo

```
// Errores absolutos y relativos
err_abs = abs(df_approx - df_true);
err_rel = err_abs./abs(df_true);
err_abs_sci = abs(df_scilab - df_true);
err_rel_sci = err_abs_sci/abs(df_true);

// Grafica
plot(ih,log10(err_rel),'b*-'); // Gráfica en escala logaritmica en el eje y
title('Error relativo utilizando diferencias finitas');
xlabel('i');
ylabel('$log_{10} (Err Rel)$');
plot(ih,log10(err_rel_sci),'r-');

// Impresion de resultados en pantalla
tablevalue = [ih,h,df_true*ones(length(h),1),df_approx,err_abs,err_rel];
mprintf('%s\n',strcat(repmat('-',1,80)));
mprintf('%4s  %8s  %12s  %18s  %14s  %14s\n',...
   'i', 'h','Der.  exact','Der approx','Abs.  error','Rel.  error');
mprintf('%s\n',strcat(repmat('-',1,80)));
mprintf('%4d  %8.1e  %9.6e  %18.10e  %14.5e  %14.5e\n',tablevalue);
mprintf('%s\n',strcat(repmat('-',1,80)));
mprintf('%4.1s  %8s  %9.6e  %18.10e  %14.5e  %14.5e\n',...
   ' ', 'Scilab',[df_true,df_scilab,err_abs_sci,err_rel_sci]);
mprintf('%s\n',strcat(repmat('-',1,80)));
```


Ejemplo

i	h	Der. exact	Der approx	Abs. error	Rel. error
0	1.0e+000	2.000000e+000	3.0000000000e+000	1.00000e+000	5.00000e-001
1	1.0e-001	2.000000e+000	2.1000000000e+000	1.00000e-001	5.00000e-002
2	1.0e-002	2.000000e+000	2.0100000000e+000	1.00000e-002	5.00000e-003
3	1.0e-003	2.000000e+000	2.0010000000e+000	1.00000e-003	5.00000e-004
4	1.0e-004	2.000000e+000	2.0001000000e+000	1.00000e-004	5.00000e-005
5	1.0e-005	2.000000e+000	2.0000100000e+000	1.00000e-005	5.00001e-006
6	1.0e-006	2.000000e+000	2.0000009999e+000	9.99924e-007	4.99962e-007
7	1.0e-007	2.000000e+000	2.0000001011e+000	1.01088e-007	5.05439e-008
8	1.0e-008	2.000000e+000	1.9999999878e+000	1.21549e-008	6.07747e-009
9	1.0e-009	2.000000e+000	2.0000001655e+000	1.65481e-007	8.27404e-008
10	1.0e-010	2.000000e+000	2.0000001655e+000	1.65481e-007	8.27404e-008
11	1.0e-011	2.000000e+000	2.0000001655e+000	1.65481e-007	8.27404e-008
12	1.0e-012	2.000000e+000	2.0001778012e+000	1.77801e-004	8.89006e-005
13	1.0e-013	2.000000e+000	1.9984014443e+000	1.59856e-003	7.99278e-004
14	1.0e-014	2.000000e+000	1.9984014443e+000	1.59856e-003	7.99278e-004
15	1.0e-015	2.000000e+000	2.2204460493e+000	2.20446e-001	1.10223e-001
16	1.0e-016	2.000000e+000	0.0000000000e+000	2.00000e+000	1.00000e+000
	Scilab	2.000000e+000	2.0000000149e+000	1.49012e-008	7.45058e-009

Ejemplo



Ejercicios

Ejercicio (2)

Usando aritmética de cuatro dígitos de precisión (mantisa decimal de 4 dígitos con redondeo), sume la siguiente expresión

$$0,1025 \cdot 10^4 + (-0,9123) \cdot 10^3 + (-0,9663) \cdot 10^2 + (-0,9315) \cdot 10^1$$

tanto ordenando los números de mayor a menor (en valor absoluto), como de menor a mayor. Realiza cada operación de forma separada, primero igualando exponentes y luego normalizando el resultado en cada paso. ¿Cuál de las dos posibilidades es más exacta? Justifique los resultados que encuentre.

Ejercicios

La suma exacta es $1025 - 912,3 - 96,63 - 9,315 = 6,755$
Sumar de mayor a menor, requiere evaluar con el orden

$$(((0,1025 \cdot 10^4 + (-0,9123) \cdot 10^3) + (-0,9663) \cdot 10^2) + (-0,9315) \cdot 10^1)$$

$$\begin{aligned}
s_1 &= 0,1025 \cdot 10^4 \\
s_2 &= s_1 - 0,0912 \cdot 10^4 \\
&= 0,0113 \cdot 10^4 \\
&= 0,1130 \cdot 10^3 \\
s_3 &= s_2 - 0,09663 \cdot 10^3 \\
&= 0,1130 \cdot 10^3 - 0,0966 \cdot 10^3 \\
&= 0,0164 \cdot 10^3 = 0,1640 \cdot 10^2 \\
s_4 &= s_3 - 0,09315 \cdot 10^2 \\
&= 0,1640 \cdot 10^2 - 0,0932 \cdot 10^2 \\
&= 0,0708 \cdot 10^2 = 0,7080 \cdot 10^1 = 7,080
\end{aligned}$$

$$E_r = \frac{7,08 - 6,755}{6,755} = 0,048$$

Ejercicios

Sumar de menor a mayor requiere evaluar con el orden:

$$(((-0,9315) \cdot 10^1 + (-0,9663) \cdot 10^2) + (-0,9123) \cdot 10^3) + 0,1025 \cdot 10^4$$

$$\begin{aligned}
z_1 &= -0,9315 \cdot 10^1 \\
z_2 &= z_1 - 0,9663 \cdot 10^2 \\
&= -0,09315 \cdot 10^2 - 0,9663 \cdot 10^2 \approx -0,0932 \cdot 10^2 - 0,9663 \cdot 10^2 \\
&= -1,0595 \cdot 10^2 \\
&= -0,1060 \cdot 10^3 \\
z_3 &= z_2 - 0,9123 \cdot 10^3 \\
&= -0,1060 \cdot 10^3 - 0,9123 \cdot 10^3 \\
&= -1,0183 \cdot 10^3 \\
&= -0,1018 \cdot 10^4 \\
z_4 &= z_3 + 0,1025 \cdot 10^4 \\
&= -0,1018 \cdot 10^4 + 0,1025 \cdot 10^4 \\
&= 0,0007 \cdot 10^4 = 0,7000 \cdot 10^1 = 7
\end{aligned}$$

$$E_r = \frac{7 - 6,755}{6.755} = 0,036$$