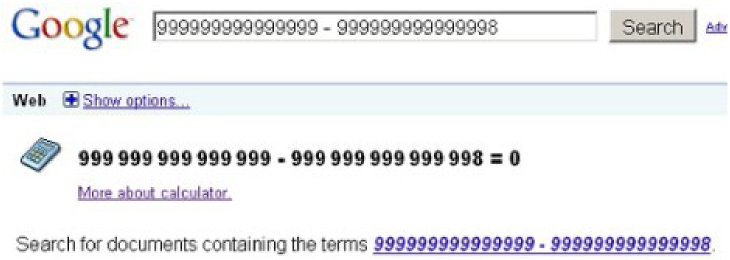
Errores Numéricos

Juan Manuel Rabasedas

20/09/2018



Introducción

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

Introducción

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

- Sistemas de cómputo simbólico

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

- Sistemas de cómputo simbólico
    - Maple, Mathematica, Maxima

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

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.

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

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

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

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.

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 Puto Flotante

- Scilab almacena los números reales en punto flotante de doble precisión de 64 bits (IEEE 754)

Números Reales y Puto Flotante

- Scilab almacena los números reales en punto flotante de doble precisión de 64 bits (IEEE 754)
- Tenemos infinitos no numerables números reales

Números Reales y Puto 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

Números Reales y Puto 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).

Números Reales y Puto 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 difernte s 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

Números Reales y Puto 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.

Números Reales y Puto 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 diferntes 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 $[-10^{308},\ -10^{-307}]$

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 $[10^{307}, \ 10^{308}]$

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)**

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 númro $0,1$ 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.

Errores de redondeo

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

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

Errores de redondeo

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
$(E)_2 = 01111111011$

Errores de redondeo

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$

Errores de redondeo

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011010$

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011010$
  $(M)_{10} = 1,6$

Errores de redondeo

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011010$
  $(M)_{10} = 1,6$
- Luego la representación es:
  $fl(0.1) = 1,6 \cdot 2^{1019-1023} = 1,6 \cdot 2^{-4}$

Errores de redondeo

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011010$
  $(M)_{10} = 1,6$
- Luego la representación es:
  $fl(0.1) = 1,6 \cdot 2^{1019-1023} = 1,6 \cdot 2^{-4}$
- Calcular en Scilab

Errores de redondeo

- La representación de $0,1$ no es exacta.
- Un núemro 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 mantiza:
  $(E)_2 = 01111111011$
  $(E)_{10} = 1019$
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011010$
  $(M)_{10} = 1,6$
- 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 mantiza:
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011001$

Errores de redondeo

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

Errores de redondeo

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

Errores de redondeo

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

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

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

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

- El número en punto flotante anterior al calculado tiene la siguiente mantiza:
  $(M)_2 = 1,1001100110011001100110011001100110011001100110011001$
  $(M)_{10} = 1,5999999999999999$
- 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:
  $|0,1 - 1,5999999999999999 \cdot 2^{-4}| = 8,33 \ldots 10^{-18}$
  $|0,1 - 1,6000000000000000 \cdot 2^{-4}| = 5,55 \ldots 10^{-18}$

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
// Evaluacion 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 $b > 0$

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

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