Análisis de error

Análisis Numérico

Joaquin Cavieres G.

```r
#======================================
# Ejemplo 1 (error de máquina epsilon)
#======================================
.Machine$double.eps
```

```
## [1] 2.220446e-16
```

```r
#======================================
# Ejemplo 2
#======================================
# 1 + \epsilon
print(1 + .Machine$double.eps,     digits = 20)
```

```
## [1] 1.0000000000000002
```

```r
print(1 + .Machine$double.eps * 2, digits = 20)
```

```
## [1] 1.0000000000000004
```

```r
print(1 + .Machine$double.eps / 2, digits = 20)
```

```
## [1] 1
```

```r
#======================================
# Ejemplo 3
#======================================
# 1 - \epsilon
# Para ver el número
.Machine$double.neg.eps
```

```
## [1] 1.110223e-16
```

```r
# Para ver comoa actua
print(1 - .Machine$double.neg.eps,     digits = 20)
```

```
## [1] 0.99999999999999989
```

```r
print(1 - .Machine$double.neg.eps * 2, digits = 20)
```

```
## [1] 0.99999999999999978
```

```r
print(1 - .Machine$double.neg.eps / 2, digits = 20)
```

```
## [1] 1
```

```r
#=====================================
# Ejemplo 4
#=====================================
# Agregamos 1000 a epsilon
print(1000 + .Machine$double.eps, digits = 20)
```

```
## [1] 1000
```

Aquí se puede ver que no existe efecto en esta suma, pero realizando el mismo calculo pero mediante otra instrucción (debemos cargar la librería `pracma`) se tiene:

```r
library(pracma)
eps(1000)
```

```
## [1] 1.136868e-13
```

```r
eps(1000000)
```

```
## [1] 1.164153e-10
```

```r
eps(1000000000)
```

```
## [1] 1.192093e-07
```

Así el valor de ϵ_x crece con cada aumento en x y la magnitud de ϵ_x tiene una relacón linear con la magnitud de x.

```r
#=====================================
# Ejemplo 5 (perdida de significancia)
#=====================================
1 / 3 - 0.33333333333333
```

```
## [1] 3.330669e-15
```

```r
# Veamos la perdida de significancia al restar un número cerca de 1
1 - 0.999999999999
```

```
## [1] 9.999779e-13
```

La pérdida de significancia ocurrirá al restar dos números cercanos cuando al menos uno no está perfectamente representado en número binario.

```r
#======================================
# Ejemplo 6 (perdida de significancia)
#======================================
(1 - 0.999999999999) * 1000
```

```
## [1] 9.999779e-10
```

```r
# En la practica podemos remediar este tipo de problemas reooganizando las
# ecuaciones. Por ejemplo:
20.55 - 19.2 - 1.35
```

```
## [1] 1.332268e-15
```

```r
20.55 - 1.35 - 19.2
```

```
## [1] 0
```

Cuando hacemos estos dos cálculos a mano el valor es 0, pero una simple reorganización de la resta para permitir que la computadora haga las matemáticas, cambia el resultado de algo lo suficientemente cercano a 0 a un valor que es verdaderamente 0.

```r
#======================================
# Ejemplo 7 (perdida de significancia)
#======================================
# Función cuadratica
quadratic <- function (b2 , b1 , b0) {
  t1 <- sqrt (b1 ^2 - 4 * b2 * b0)
  t2 <- 2 * b2
  x1 <- - (b1 + t1) / t2
  x2 <- - (b1 - t1) / t2
  return (c(x1 , x2))
}
```

Si trabajamos con la formula clásica:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

la cual tiene una resta que sigue con una operación de raíz cuadrada y está sujeta a una división. Esto puede llevar a error cuando tenemos restas de iguales magnitudes (en este caso de $b^2 - 4ac$). Veamos una resolución simple de la ecuación si tenemos $24x^2 - 50x - 14$

$$
\begin{aligned}
\sqrt{b^2 - 4ac} &= 50^2 - 4(24)(14) \\
&= 2500 - 1344 \\
&= 1156,
\end{aligned}
$$

y la raíz cuadrada de 1156 es igual a 34. Ahora, veamos el siguiente ejemplo y considere:

$$
\begin{aligned}
x =& \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\
=& \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} * \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} \\
=& \frac{2c}{-b \pm \sqrt{b^2 - 4ac}}
\end{aligned}
$$

```r
quadratic2 <- function (b2 , b1 , b0) {
t1 <- sqrt (b1 ^2 - 4 * b2 * b0)
t2 <- 2 * b0
x1 <- t2 / (-b1 - t1)
x2 <- t2 / (-b1 + t1)
## Reverse the order so they come
## back the same as quadratic ()
return (c(x2 , x1))
}
```

La función anterior es equivalente a la función cuadrática ya representada matemáticamente. Generalmente los cálculos numéricos de doble precisión proporcionan suficiente significancia, sin embargo, es posible experimentar problemas en casos particulares. Por ejemplo:

$$
\begin{aligned}
a =& 94906265{,}625 \\
b =& 189812534{,}000 \\
c =& 94906268{,}375
\end{aligned}
$$

Estos valores del ejemplo dan resultados incorrectos y lo veremos a través de la función `print()` en `R`.

```r
b0 = 94906268.375
b1 = 189812534.000
b2 = 94906265.625
print(quadratic(b0, b1, b2), digits = 20)
```

```
## [1] -0.99999998551202129 -0.99999998551202129
```

Aquí el resultado corecto es 1 y 1.000000028975958… , sin embargo, incluso con una precisión doble, `R` no puede calcular correctamente la función cuadrática para los valores dados. Por otra parte, tampoco todas las x propuestas pueden encontrar todas las posibles soluciones al problema. Veamos con la función `quadratic2`:

```r
b0 = 94906268.375
b1 = 189812534.000
b2 = 94906265.625
print(quadratic2(b2, b1, b0), digits = 20)
```

```
## [1] -1.000000014487979 -1.000000014487979
```

Nuestro resultado es incorrecto, diferente y poco significativo.