Análisis Numérico - Taller Derivación Numérica

Juan Francisco Hamón, Diego Mauricio Bulla, Juan Diego Campos

27/10/2019

a)Genere una tabla para evaluar el valor aproximado $f'(x) \approx x\cos(x)$ de $f'(1.8) \approx$ para los siguientes valores de $h = 0.1{,}0.01{,}0.0011{,}0.0001$

```
library(pracma)
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
f = function(x){x*cos(x)}

fo = function(x,h){( ( f(x + h) - f(x) )/ h )}


#Con h=0,1

r11 = fo(1.8,0.1)
r1 = fderiv(f,1.8, h=0.1)

cat("Con h = 0.1 \n")

## Con h = 0.1

print(r1)

## [1] -1.976073

print(r11)

## [1] -2.052864

#Error absoluto
absoluto = abs(r1 - r11)
cat("Error absoluto = ", absoluto, "\n")

## Error absoluto =  0.07679138

#Error relativo
relativo = (abs(r1 - r11)) / (abs(r1))
cat("Error relativo = ", relativo, "\n")

## Error relativo =  0.03886061
```

```r
#Con h=0,01
r11 = fo(1.8,0.01)
r1 = fderiv(f,1.8, h=0.01)

cat("Con h = 0.01 \n")

## Con h = 0.01

print(r1)

## [1] -1.980087

print(r11)

## [1] -1.987781

#Error absoluto
absoluto = abs(r1 - r11)
cat("Error absoluto = ", absoluto, "\n")

## Error absoluto =  0.007693512

#Error relativo
relativo = (abs(r1 - r11)) / (abs(r1))
cat("Error relativo = ", relativo, "\n")

## Error relativo =  0.003885441

#Con h=0,0011
r11 = fo(1.8,0.0011)
r1 = fderiv(f,1.8, h=0.0011)

cat("Con h = 0.0011 \n")

## Con h = 0.0011

print(r1)

## [1] -1.980127

print(r11)

## [1] -1.980974

#Error absoluto
absoluto = abs(r1 - r11)
cat("Error absoluto = ", absoluto, "\n")

## Error absoluto =  0.0008463021

#Error relativo
relativo = (abs(r1 - r11)) / (abs(r1))
cat("Error relativo = ", relativo, "\n")
```

```
## Error relativo =  0.0004273978

#Con h=0,0001
r11 = fo(1.8,0.0001)
r1 = fderiv(f,1.8, h=0.0001)

cat("Con h = 0.0001 \n")

## Con h = 0.0001

print(r1)

## [1] -1.980128

print(r11)

## [1] -1.980205

#Error absoluto
absoluto = abs(r1 - r11)
cat("Error absoluto = ", absoluto, "\n")

## Error absoluto =  7.693657e-05

#Error relativo
relativo = (abs(r1 - r11)) / (abs(r1))
cat("Error relativo = ", relativo, "\n")

## Error relativo =  3.885435e-05
```

b)Estime el valor aproximado de las cotas del error para el problema anterior.

```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
#Derivadas de la función original
f = expression(x*cos(x))
dF1 = D(f,'x')
dF2 = D(dF1, 'x')
cat("Primera derivada: ")

## Primera derivada:

print(dF1)

## cos(x) - x * sin(x)

cat("\nSegunda derivada: ")
```

```
##
## Segunda derivada:

print(dF2)

## -(sin(x) + (sin(x) + x * cos(x)))

cat("\n")

funDF2 <- function(x) {eval(dF2)}
M = funDF2(1.8)

#Resultado para 0.1
resultado = abs(0.1*M)/2
cat("Resultado para h = 0.1 es: ",resultado,"\n")

## Resultado para h = 0.1 es:  0.07693657

#Resultado para 0.01
resultado = abs(0.01*M)/2
cat("Resultado para h = 0.01 es: ",resultado,"\n")

## Resultado para h = 0.01 es:  0.007693657

#Resultado para 0.0011
resultado = abs(0.0011*M)/2
cat("Resultado para h = 0.0011 es: ",resultado,"\n")

## Resultado para h = 0.0011 es:  0.0008463023

#Resultado para 0.0001
resultado = abs(0.0001*M)/2
cat("Resultado para h = 0.0001 es: ",resultado,"\n")

## Resultado para h = 0.0001 es:  7.693657e-05
```

c)¿Cuál es el valor de h que proporciona una aproximación con una precisión de 10^{-4}?

```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
f2 <- function(x) -(sin(x) + (sin(x) + x * cos(x)))
M = f2(1.8)

cota = function(h) { abs((h*M)/2) -10^-4 }
respuesta = uniroot(cota, c(0,1))$root
cat("El valor de h que proporciona una aproximación con una estimación de 
10^-4 es de ",respuesta,"\n")
```

```
## El valor de h que proporciona una aproximación con una estimación de 
10^-4 es de  0.0001299772
```

d)Supóngase que se tienen tres puntos dados por: x_0; $x_1 = x_0 + h$; $x_2 = x_0 + 2h$. Encuentre la fórmula conocida de tres puntos para determinar una aproximación de $f'(x_0)$. Donde ξ_0 se encuentra entre x_0 y $x_0 + 2h$. Utilice esta fórmula para encontrar $\approx f'(1.8)$.

Para encontrar la fórmula de tres puntos, que es la que corresponde con los parámetros dados en el enunciado, se parte del polinomio de Taylor de segundo grado.

$$f(x) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2 + \frac{f'''(\xi)}{6}h^3$$

Despejando $f'(x_0)$ se obtiene:

$$f'(x_0) = \frac{f(x) - f(x_0)}{h} - \frac{f''(x_0)}{2}h - \frac{f'''(\xi)}{6}h^2$$

Ahora, aplicando el valores de x_1 en x dentro de la ecuación que se encontró anteriormente se obtiene:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{f''(x_0)}{2}h - \frac{f'''(\xi)}{6}h^2$$

Para determinar $f''(x_0)$ en terminos de x_0, x_1 y x_2, se construyen los polinomios de Taylor de segundo orden para $x_1 = x_0 + h$ y para $x_2 = x_0 + 2h$.

$f(x_0 + h) = f(x_0) + f'(x_0)h + \frac{f''(x_0)}{2}h^2$; Ecuación 1.

$f(x_0 + 2h) = f(x_0) + f'(x_0)2h + \frac{f''(x_0)}{2}(2h)^2$; Ecuación 2.

Ahora, multiplicando la ecuación 1 por -2 y sumándola con la ecuación 2 se obtiene:

$$f(x_0 + 2h) - 2f(x_0 + h) = -f(x_0) + f''(x_0)h^2$$

Al despejar $f''(x_0)$ de la ecuación anterior se tiene:

$f''(x_0) = \frac{f(x_0) - 2f(x_0 + h) + f(x_0 + 2h)}{h^2} + Oh$; Ecuación 3.

Al reemplazar la ecuación 3 dentro de la ecuación 1 se obtiene la fórmula de los tres puntos:

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h} + Oh^2$$

Ahora, tras haber encontrado la ecuación, se utilizará para encontrar el valor aproximado de $f'(1.8)$.

```r
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```r
x0 = 1.8

f = function(x) {x*cos(x)}

#Para 0.1
cat("Valor aproximado para h = 0.1: \n")

## Valor aproximado para h = 0.1:

h = 0.1

dFx = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx)

## [1] -1.989079

cat("\n")

#Para 0.01
cat("Valor aproximado para h = 0.01: \n")

## Valor aproximado para h = 0.01:

h = 0.01
dFx = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx)

## [1] -1.98021

cat("\n")

#Para 0.0011
cat("Valor aproximado para h = 0.0011: \n")

## Valor aproximado para h = 0.0011:

h = 0.0011
dFx = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx)

## [1] -1.980129

cat("\n")

#Para 0.0001
cat("Valor aproximado para h = 0.0001: \n")
```

```
## Valor aproximado para h = 0.0001:

h = 0.0001
dFx = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx)

## [1] -1.980128

cat("\n")
```

e)Realice una modificación de la fórmula de los tres puntos, tomando valores entre $(x_0 - h)$ y $(x_0 + h)$ y compare la magnitud del error con la fórmula de la parte d.

Al realizar los cambios correspondientes en el despeje y reemplazo de las ecuaciones encontradas en el punto anterior, se llega a la fórmula de los tres puntos centrada:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
x0 = 1.8
fe = expression(x*cos(x))
DFe = D(fe, 'x')
dff = function(x) {eval(DFe)}
valorReal = dff(1.8)


f = function(x) {x*cos(x)}

#Para 0.1
cat("Valor aproximado para h = 0.1 por tres puntos: \n")

## Valor aproximado para h = 0.1 por tres puntos:

h = 0.1
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.989079

cat("\n")

cat("Valor aproximado para h = 0.1 por tres puntos centrada: \n")

## Valor aproximado para h = 0.1 por tres puntos centrada:
```

```r
dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.976073

cat("\n")

cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 0.4520308

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 0.2047922

cat("\n")

#Para 0.01
cat("Valor aproximado para h = 0.01 por tres puntos: \n")

## Valor aproximado para h = 0.01 por tres puntos:

h = 0.01
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.98021

cat("\n")

cat("Valor aproximado para h = 0.01 por tres puntos centrada: \n")

## Valor aproximado para h = 0.01 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.980087

cat("\n")

cat("Error estimado por tres puntos: \n")
```

```
## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 0.004142121

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 0.002049125

cat("\n")

#Para 0.0011
cat("Valor aproximado para h = 0.0011 por tres puntos: \n")

## Valor aproximado para h = 0.0011 por tres puntos:

h = 0.0011
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.980129

cat("\n")

cat("Valor aproximado para h = 0.0011 por tres puntos centrada: \n")

## Valor aproximado para h = 0.0011 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.980127

cat("\n")

cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 4.96477e-05

cat("\n")
```

```r
cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 2.479455e-05

cat("\n")

#Para 0.0001
cat("Valor aproximado para h = 0.0001 por tres puntos: \n")

## Valor aproximado para h = 0.0001 por tres puntos:

h = 0.0001
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.980128

cat("\n")

cat("Valor aproximado para h = 0.0001 por tres puntos centrada: \n")

## Valor aproximado para h = 0.0001 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.980128

cat("\n")

cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 4.098902e-07

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 2.049122e-07
```

```
cat("\n")
```

f)Utilice la fórmula para cinco puntos alrededor x_0 y aplíquela y compárela con todas las fórmulas anteriores.

```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
x0 = 1.8
fe = expression(x*cos(x))
DFe = D(fe, 'x')
dff = function(x) {eval(DFe)}
valorReal = dff(1.8)

f = function(x) {x*cos(x)}

#Para 0.1
cat("Valor aproximado para h = 0.1 por tres puntos: \n")

## Valor aproximado para h = 0.1 por tres puntos:

h = 0.1
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.989079

cat("\n")

cat("Valor aproximado para h = 0.1 por tres puntos centrada: \n")

## Valor aproximado para h = 0.1 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.976073

cat("\n")

cat("Valor aproximado para h = 0.1 por cinco puntos: \n")

## Valor aproximado para h = 0.1 por cinco puntos:

dFx3 = (f(x0-2*h)-8*f(x0-h)+8*f(x0+h)-f(x0+2*h))/(12*h)
print(dFx3)

## [1] -1.980118

cat("\n")
```

```r
cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 0.4520308

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 0.2047922

cat("\n")

cat("Error estimado por cinco puntos: \n")

## Error estimado por cinco puntos:

error5 = abs(((valorReal - dFx3)/valorReal)*100)
print(error5)

## [1] 0.0004856519

cat("\n")

#Para 0.01
cat("Valor aproximado para h = 0.01 por tres puntos: \n")

## Valor aproximado para h = 0.01 por tres puntos:

h = 0.01
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.98021

cat("\n")

cat("Valor aproximado para h = 0.01 por tres puntos centrada: \n")

## Valor aproximado para h = 0.01 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.980087
```

```r
cat("\n")

cat("Valor aproximado para h = 0.01 por cinco puntos: \n")

## Valor aproximado para h = 0.01 por cinco puntos:

dFx3 = (f(x0-2*h)-8*f(x0-h)+8*f(x0+h)-f(x0+2*h))/(12*h)
print(dFx3)

## [1] -1.980128

cat("\n")

cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 0.004142121

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 0.002049125

cat("\n")

cat("Error estimado por cinco puntos: \n")

## Error estimado por cinco puntos:

error5 = abs(((valorReal - dFx3)/valorReal)*100)
print(error5)

## [1] 4.86312e-08

cat("\n")

#Para 0.0011
cat("Valor aproximado para h = 0.0011 por tres puntos: \n")

## Valor aproximado para h = 0.0011 por tres puntos:

h = 0.0011
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)
```

```
## [1] -1.980129

cat("\n")

cat("Valor aproximado para h = 0.0011 por tres puntos centrada: \n")

## Valor aproximado para h = 0.0011 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.980127

cat("\n")

cat("Valor aproximado para h = 0.0011 por cinco puntos: \n")

## Valor aproximado para h = 0.0011 por cinco puntos:

dFx3 = (f(x0-2*h)-8*f(x0-h)+8*f(x0+h)-f(x0+2*h))/(12*h)
print(dFx3)

## [1] -1.980128

cat("\n")

cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 4.96477e-05

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)

## [1] 2.479455e-05

cat("\n")

cat("Error estimado por cinco puntos: \n")

## Error estimado por cinco puntos:

error5 = abs(((valorReal - dFx3)/valorReal)*100)
print(error5)

## [1] 5.618039e-12
```

```r
cat("\n")

#Para 0.0001
cat("Valor aproximado para h = 0.0001 por tres puntos: \n")

## Valor aproximado para h = 0.0001 por tres puntos:

h = 0.0001
dFx1 = (-f(x0+2*h)+4*f(x0+h)-3*f(x0))/(2*h)
print(dFx1)

## [1] -1.980128

cat("\n")

cat("Valor aproximado para h = 0.0001 por tres puntos centrada: \n")

## Valor aproximado para h = 0.0001 por tres puntos centrada:

dFx2 = (1/(2*h) ) * ( f(x0+h) - f(x0-h) )
print(dFx2)

## [1] -1.980128

cat("\n")

cat("Valor aproximado para h = 0.0001 por cinco puntos: \n")

## Valor aproximado para h = 0.0001 por cinco puntos:

dFx3 = (f(x0-2*h)-8*f(x0-h)+8*f(x0+h)-f(x0+2*h))/(12*h)
print(dFx3)

## [1] -1.980128

cat("\n")

cat("Error estimado por tres puntos: \n")

## Error estimado por tres puntos:

error3 = abs(((valorReal - dFx1)/valorReal)*100)
print(error3)

## [1] 4.098902e-07

cat("\n")

cat("Error estimado por tres puntos centrada: \n")

## Error estimado por tres puntos centrada:

error3c = abs(((valorReal - dFx2)/valorReal)*100)
print(error3c)
```

```
## [1] 2.049122e-07

cat("\n")

cat("Error estimado por cinco puntos: \n")

## Error estimado por cinco puntos:

error5 = abs(((valorReal - dFx3)/valorReal)*100)
print(error5)

## [1] 7.950478e-12

cat("\n")
```

g)Aplique la fórmula adecuada para aproximar $f''(1.8)$. Justifique su respuesta.

```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
x0 = 1.8
fe = expression(x*cos(x))
DFe = D(fe, 'x')
DF2e = D(DFe, 'x')
dff = function(x) {eval(DF2e)}
valorReal = dff(1.8)
valorReal

## [1] -1.538731

f = function(x) {x*cos(x)}
h = 0.1

#3 Puntos centrada
cat("Resultado usando fórmula de 3 puntos centrada: \n")

## Resultado usando fórmula de 3 puntos centrada:

dFx = ( (1/(h^2) ) * ( (f(x0+h)) - (2*f(x0)) + (f(x0-h)) ) )
print(dFx)

## [1] -1.535828

cat("\n")

Error = abs(((valorReal - dFx)/valorReal)*100)
cat("Error estimado para fórmula de 3 puntos centrada: \n")

## Error estimado para fórmula de 3 puntos centrada:
```

```
print(Error)

## [1] 0.1887169

cat("\n")
```

La razón para escoger la fórmula de 3 puntos centrada es debido a que tiene un orden de complejidad $O(h^2)$. Así mismo, presenta un error bajo al realizar la operación para encontrar la aproximación deseada.

h)Teniendo en cuenta que el error total $(h) = $error de redondeo + error de truncamiento dado por: $e(h) = \frac{\xi}{h} + \frac{h^2}{6}M$ como una función del tamaño del paso. Encuentre el tamaño óptimo del paso.

Teniendo la ecuación $e(h) = \frac{\xi}{h} + \frac{h^2}{6}M$, se puede encontrar su primera derivada para obtener:

$$e'(h) = -\frac{\xi}{h^2} + \frac{h}{3}M$$

Ahora, al igualar la ecuación a 0 se forma el siguiente sistema:

$$\frac{\xi}{h^2} = \frac{h}{3}M$$

Por último, al despejar h se puede determinar el tamaño del paso óptimo.

$$h = \sqrt[3]{\frac{3\xi}{M}}$$

i)El siguiente código está dado para aproximar $f'(1)$; $f(x) = xe^x$, realice una gráfica que muestre como varía la precisión en función de h.

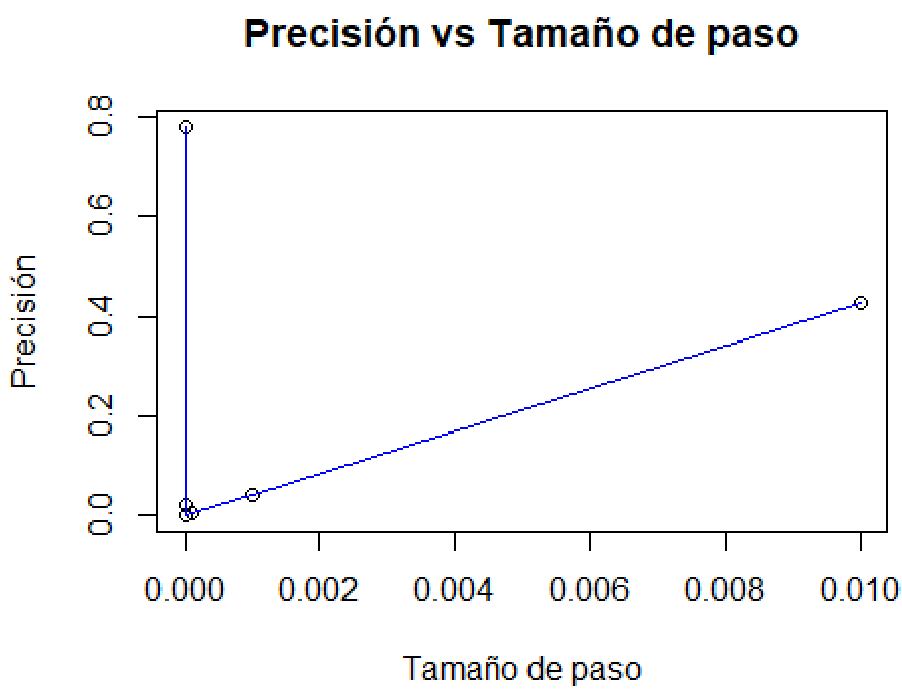
```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
f = function(x) {x*exp(x)}
r = 5.436563656918091
h = 0.1
x = c()
y = c()
i=0

while(i < 15){
  d = (f(1+h)-f(1))/h
  e = abs(r-d)
```

```
  h = h/10
  i = i+1
  x[i] = h
  y[i] = e
}

plot(x, y, main = "Precisión vs Tamaño de paso", xlab = "Tamaño de paso",
ylab = "Precisión")
lines(x, y, col = "blue")
```



j)Aplicación: En un circuito con un voltaje $E(t)$ y una inductancia L se tiene que: $E(t) = L\frac{di}{dt} + Ri$ donde R es la resistencia e i es la corriente. La siguiente tabla muestra la media de la corriente para varios instantes de tiempo (segundos), con $R = 0.142$ ohms y $L = 0.98$ henries. Aproxime el voltaje para los valores de t.

```
#Se limpian los elementos creados con anterioridad
rm(list=ls())
#Se limpia la consola para una mejor visualizacion
cat("\014")
```

```
E <- function(L,R,i,di,dt) (L*(di/dt)+R*i)
valR <- 0.142
valL <- 0.98

#Para 1.01
```

```r
diffT = 1.01 - 1.00
diffI = 3.12 - 3.10
cat("Voltaje aproximado en 1.01: ",E(valL, valR, 3.12, diffI,
diffT),"\n")

## Voltaje aproximado en 1.01:  2.40304

#Para 1.02
diffT = 1.02 - 1.01
diffI = 3.14 - 3.12
cat("Voltaje aproximado en 1.02: ",E(valL, valR, 3.14, diffI,
diffT),"\n")

## Voltaje aproximado en 1.02:  2.40588

#Para 1.03
diffT = 1.03 - 1.02
diffI = 3.18 - 3.14
cat("Voltaje aproximado en 1.03: ",E(valL, valR, 3.18, diffI,
diffT),"\n")

## Voltaje aproximado en 1.03:  4.37156

#Para 1.04
diffT = 1.04 - 1.03
diffI = 3.24 - 3.18
cat("Voltaje aproximado en 1.04: ",E(valL, valR, 3.24, diffI,
diffT),"\n")

## Voltaje aproximado en 1.04:  6.34008
```