Integración Numérica

Análisis Numérico

Joaquin Cavieres G.

Integración Numérica

Diferencias finitas

Ejemplo 1

```r
# Creamos el vector x y el vector fx (valores)
x  <- c(0.0, 0.2, 0.4)
fx <- c(0.00000, 0.74140, 1.3718)
```

La funcion que aproxima es desconocida, sin embargo, queremos aproximar la derivada de la funcion a los valores de x. Lo vamos a hacer mediante el metodo de forward y backward

```r
dif_finitas <- function(x, y) {
  if (length(x) != length(y)) {
    stop('x e y deben tener el mismo largo')
  }

  n <- length(x)

  # Inicializar el vector de largo n para guardar las derivadas aproximadas
  fdx <- vector(length = n)

  # Iterar a traves de los valores usando el método de forward
  for (i in 2:n) {
    fdx[i-1] <- (y[i-1] - y[i]) / (x[i-1] - x[i])
  }

  # Para el ultimo valor, y como no podemos usar forward ya que se conocen sólo los
  # n valores, usamos el backward.
  fdx[n] <- (y[n] - y[n - 1]) / (x[n] - x[n - 1])

  return(fdx)
}
```

Usando la función entonces realizamos la aproximación

```
result <- dif_finitas(x, fx)
result
```

```
## [1] 3.707 3.152 3.152
```

Ejemplo 2

Asumamos que los datos provienen de una función del estilo $e^x - 2x^2 + 3x - 1$

```
f <- function(x) {
  return(exp(x) - 2 * x^2 + 3 * x - 1)
}
```

Usando la ténica central de diferencias finitas entonces podemos aproximar la derivada en cada punto. Como h aproxima a 0 entonces demostraremos mediante varios valores de h como esta aproximacion converge

```
dif_central <- function(f, x) {
  steps <- c(0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001) # Valores de h
  n <- length(steps)

  fdx <- vector(length = n)

  for (h in 1:n) {
    fdx[h] <- (f(x + 0.5 * steps[h]) - f(x - 0.5 * steps[h])) / steps[h]
  }

  return(fdx)
}
```

Vemos las aproximaciones

```
for (i in x) {
  print(dif_central(f, i))
}
```

```
## [1] 4.000417 4.000004 4.000000 4.000000 4.000000 4.000000 4.000000
## [1] 3.421912 3.421408 3.421403 3.421403 3.421403 3.421403 3.421403
## [1] 2.892446 2.891831 2.891825 2.891825 2.891825 2.891825 2.891825
```

Vemos como la funcion aproxima rapidamente cuando h tiende a 0 y dando como resultado

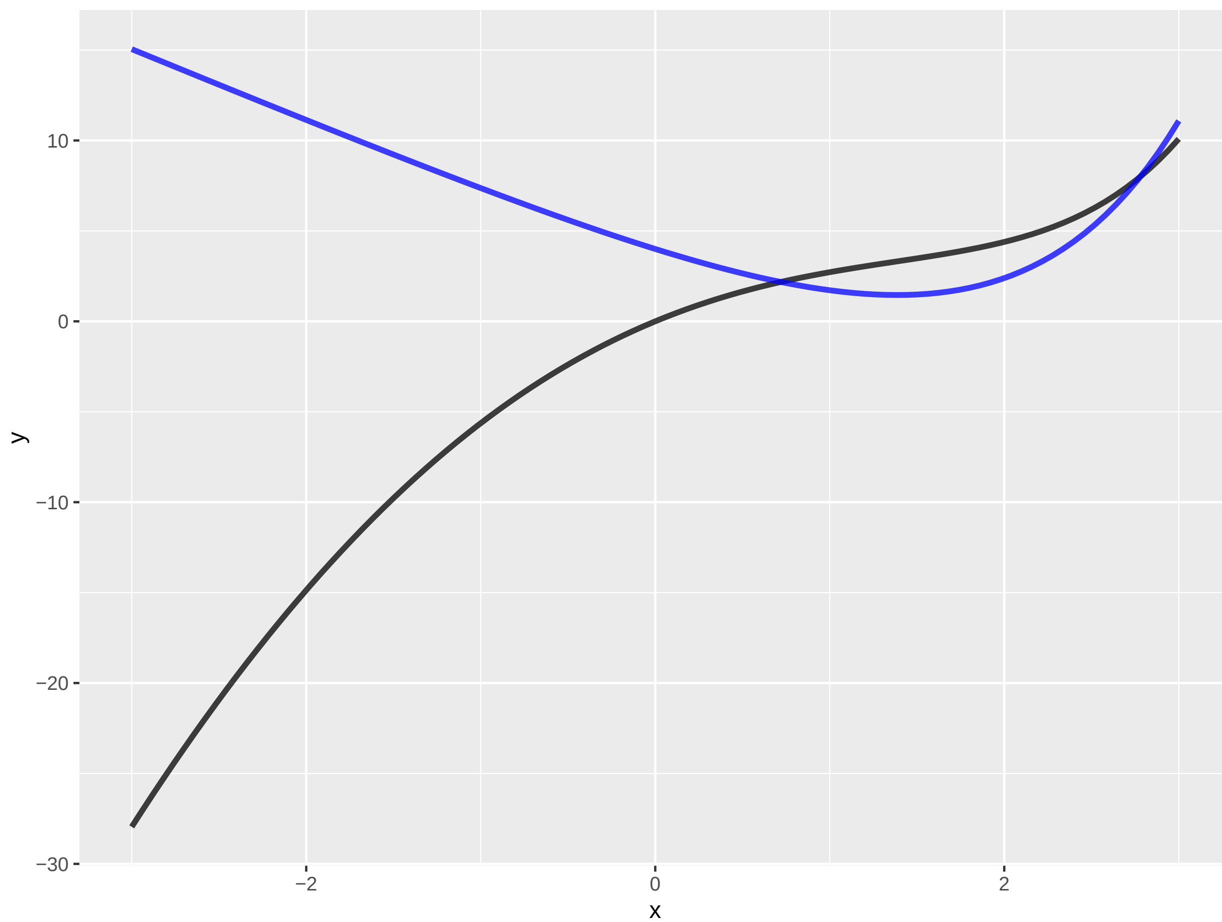
$$f'(0{,}0) \approx 4{,}00000 \qquad f'(0{,}2) \approx 3{,}421403 \qquad f'(0{,}4) \approx 2{,}891825$$

La derivada de la función $e^x - 2x^2 + 3x - 1$ es $e^x - 4x + 3$. Comparemos nuestros valores aproximados con los valores reales de la derivada en x.

```
fdx <- function(x) {
  return(exp(x) - 4 * x + 3)
}
```

y gráficamos la función y su derivada e los valores de x.

```
library(ggplot2)
ggplot(data = data.frame(x = 0), mapping = aes(x = x)) +
  stat_function(fun = f, size = 1.25, alpha = 0.75) +
  stat_function(fun = fdx, size = 1.25, color = 'blue', alpha = 0.75) +
  xlim(-3,3)
```



Finalmente, mostramos los valores de la derivada calculada para cada x y los comparamos con los valores aproximados de nuestra función.

```
actual <- vector(length = length(x))
cen.approx <- c(4.00000, 3.421403, 2.891825)
for (i in 1:length(x)) {
  actual[i] <- fdx(x[i])
}
df <- data.frame(cbind(actual, cen.approx, actual - cen.approx, result, actual - result))
colnames(df) <- c('Valores actuales', 'Diferencia central', 'Error diferencia central',
                  'Diferencias finitas', 'Error diferencias finitas')
df
```

```
##   Valores actuales Diferencia central Error diferencia central
## 1         4.000000           4.000000             0.000000e+00
## 2         3.421403           3.421403            -2.418398e-07
## 3         2.891825           2.891825            -3.023587e-07
##   Diferencias finitas Error diferencias finitas
## 1               3.707                 0.2930000
## 2               3.152                 0.2694028
## 3               3.152                -0.2601753
```

Integración Gaussiana

Creamos una función para poder aplicar el método

```
integra_gauss <- function (f, x, w) {
y <- f(x)
return (sum(y * w))
}
```

Esta función conecta los puntos de integración con los ponderadores, es decir, la función `integra_gauss` requiere como opciones tanto los puntos de evaluación como sus ponderadores asociados. La función acepta argumentos vectoriales y devolverá un vector de valores y.

Para un $n = 2$ a través de la función $f(x) = x^3 + x + 1$ y una regla de dos evaluaciones para los calculos.

```
trap <- function (f, a, b, m = 100) {
x = seq(a, b, length.out = m + 1)
y = f(x)
p.area = sum ((y [2:( m+1)] + y[1:m]))
p.area = p.area * abs(b - a) / (2 * m)
return (p.area )
}
```

```r
w = c(1, 1)
x = c(-1 / sqrt(3), 1 / sqrt(3))
f <- function(x) { x^3 + x + 1 }
integra_gauss(f, x, w)
```

```
## [1] 2
```

```r
trap(f, -1, 1, m = 1)
```

```
## [1] 2
```

Veamos la función *cos*()

```r
integra_gauss(cos, x, w)
```

```
## [1] 1.675824
```

```r
trap(cos, -1, 1, m = 1)
```

```
## [1] 1.080605
```

Veamos los resultados de la misma función coseno para una función creada con el método de Gauss-Legendre

```r
gauss_legendre <- function (f, m = 5) {
p <- paste ("gauss_legendre", m, sep = "")
params <- eval ( parse ( text = p))
return (integra_gauss(f, params $x, params $w))
}
```