Programación Estadística: Bisección

Adrián Sosa
Universidad Veracruzana

Bisección

El método de bisección es uno de los más versátiles para determinar una raíz real en un intervalo de una ecuación dada, es fácil de comprender, aunque si se desea una mayor exactitud el número de cálculos que hay que realizar aumenta considerablemente.

El método de bisección se basa en el Teorema de Bolzano, el cual afirma que si se tiene una función real $y = f(x)$ continua en el intervalo **[a,b]** donde el signo de la función en el extremo **a** es distinto al signo de la función en el extremo **b** del intervalo, entonces existe al menos un $c \in [a, b] | f(c) = 0$, que es la raíz buscada.

Procedimiento

- Debe existir seguridad sobre la continuidad de $f(x)$ sobre el segmento $[a, b]$.
- Se verifica que $f(a)f(b) < 0$.
- Se calcula el punto medio m en el intervalo $[a, b]$ y se evalúa $f(m)$ si ese valor es igual a 0 ya se ha encontrado la raíz.
- En caso contrario se verifica si $f(m)$ tiene signo opuesto con $f(a)$ o con $f(b)$.
- se redefine el intervalo $[a, b]$ como$ [a,m]$ o $[m, b]$ según se haya determinado en cual de los casos haya el cambio de signo.
- Con este nuevo intervalo se continúa sucesivamente encerrando la solución en un intervalo cada vez más pequeño, hasta alcanzar la precisión deseada

Ventajas

- Es siempre convergente.
- Es óptimo para resolver una ecuación $f(x) = 0$ cuando no se sabe nada de f, excepto calcular su signo.
- Requiere que f sea continua en el intervalo especificado.
- Se puede establecer el límite de error.
- Es fácil de implementar.

Desventajas

- Converge muy lentamente.
- Permite encontrar solo una raíz, aunque existan más en el intervalo.
- Algunas veces la determinación del intervalo inicial no es muy fácil.
- No puede determinar raíces complejas.
- Es difícil generalizarlo para dimensiones superiores.

Codificación

El método de bisección es relativamente sencillo de implementar:

```
## function (Fx, limI, limS, tol = 1e-06)
## {
##     repeat {
##         x <- (limI + limS)/2
##         L <- (limS - limI)/2
##         if (Fx(x) == 0) {
##             return(x)
##             break
##         }
##         else if (Fx(limI) * Fx(x) < 0) {
##             limS <- x
##         }
##         else {
##             limI <- x
##         }
##         if (L < tol) {
##             return(x)
##             break
##         }
##     }
## }
```

Para utilizarlo debemos definir una función para encontrar su ráiz y un espacio de búsqueda:

```
fx1
```

```
## function (x)
## {
##     return(x^2 + 3 * x - 2)
## }
```

```
# se definen los límites
limS <- 10
limI <- -10
steps <- 0.01
# se crea el espacio de búsqueda
x <- seq(limI, limS, steps)
y <- c()
for (i in seq_along(x)){y[i] <- fx1(x[i])}
# se optiene la ráiz
ráiz <- biseccion(fx1, limI, limS)
```

El resultado se almacenara en la variable *ráiz*, para conocerlo basta con imprimirlo:

```
ráiz
```

```
## [1] -3.561553
```

Si deseamos podémos gráficar la función y nuestro resultado.

```
# se crea un dataframe
df <- data.frame(x,y)
ggplot(data= df, mapping=aes(x= x, y =y))+
  geom_line(color=4)+
  annotate(geom="text",x=ráiz-2, y=0, label="Ráiz ->")+
  annotate(geom="point", x=ráiz,y =0, size =3, shape=1, fill="transparent")
```