metodos de raices

Humberto Camacho Guillen

3 de octubre del 2018

1 Método de bisección

El método de bisección en matemáticas es un método de búsqueda de raíces que divide un intervalo de forma repetida y luego selecciona un subintervalo en el que debe encontrarse una raíz para su posterior procesamiento. Es un método muy simple y robusto, pero también es relativamente lento, o en otras palabras, este método , que se utiliza para resolver ecuaciones de una variable, está basado en el "Teorema de los Valores Intermedios", en el cual se establece que toda función continua f, en un intervalo cerrado [a,b], toma todos los valores que se hallan entre f(a) y f(b), de tal forma que la ecuación f(x)=0 tiene una sola raíz que verifica f(a).f(b)¡0.

En pocas palabras, consiste en ir dividiendo la función que hemos visto arriba en subintervalos, y hallar los puntos medios de cada uno de ellos (m), quedándonos una cosa como en la figura 1

y entonces,¿como se realiza este método?

Primeramente elegimos dos valores iniciales en a1 y b1, de tal forma de que la función cambie de signo:

$$f(a_1)f(b_1) < 0 \tag{1}$$

Después se realiza la primera aproximación de la raíz, mediante la fórmula del punto medio:

$$\bar{X} = \frac{b_1 + a_1}{2} = m1 \tag{2}$$

Debemos de tener en cuenta que usammos los puntos de la función, NO la función.

Ahora determinaremos en que subinterbalo se encuentra la raíz:

tenemos que si f(a1)f(m1)¡0, entonces la raíz está en el subintervalo [a1,m1] y b1=m1

si f(a1)f(m1)¿0, entonces la raíz está en el subintervalo [b1, m1] y a1=m1

Si f(a1)f(m1)=0, entonces aquí se encuentra la raíz

Resumiendo esto, es ir hallando mitades sucesivamente hasta llegar al intervalo que nos de igual a cero.

en un cuarto paso para calcular la raiz de un polinomio con este método, tenemos que calcular una nueva aproximación de la raíz con la fórmula del punto medio.

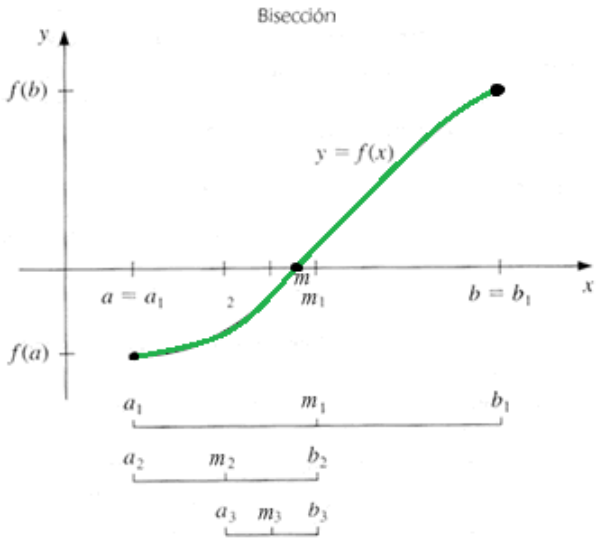

Figure 1: grafica

para despues evaluar el valor relativo aproximado con:

$$|(x_actual - \frac{xprevie}{x_actual} x100_porciento| < \epsilon_b \tag{3}$$

entonces, si:

$$|\epsilon_a| < \epsilon_b \tag{4}$$

se cumple el método y si no se cumple esta condición, se regresa a la tercera indicación y se vuelve a hacer el procedimiento.

1.1 Codigo de biseccion

```
program MBiseccion
implicit none
real:: x,y,z,error,h
error=1.0e-06
write(,)"Introduzca dos números a y b simultáneamente entre los que se encuentra la raíz"
10 read(,) x,y
15 if (f(x)*f(y) .lt. 0) then
z=(x+y)/2.0
else
write(,)"Probar con otros valores de x y y"
goto 10
```

```
end if
if (f(x)*f(z) .lt. 0) then
y=z
else
x=z
end if
if (abs(y-x) .gt. error) goto 15
write(,)"The root is",z
end program MBisección
real function f(x)
implicit none
real::xf=3*x+sin(x)-exp(x)
end function
```

2 Método de Newton

Este método, el cual es un método iterativo, es uno de los más usados y efectivos. A diferencia de los métodos anteriores, el método de Newton-Raphson no trabaja sobre un intervalo sino que basa su fórmula en un proceso iterativo.

Supongamos que tenemos la aproximación Xi a la raíz Xy de f(x),
Trazamos la recta tangente a la curva en el punto (Xi,f(x)) ; ésta cruza al eje X en un punto Xi+1 que será nuestra siguiente aproximación a la raíz Xy.
Para calcular el punto Xi+1, calculamos primero la ecuación de la recta tangente. Sabemos que tiene pendiente:

$$m = f(x) \tag{5}$$

Por lo que la ecuación de la recta tangente es:

$$y - f(x_i) = f(x_i)(x - x_i) \tag{6}$$

Hacemos y=0 :

$$-f(x_i) = f(x_i)(x - x_i) \tag{7}$$

Despejamos X:

$$x = x_i - \frac{f(x_i)}{f(x_i)} \tag{8}$$

Que es la fómula iterativa de Newton-Raphson para calcular la siguiente aproximación:

$$x_i + 1 = x_i - \frac{f(x_i)}{f(x_i)}, si f(x_i) \neq 0 \tag{9}$$

Note que el método de Newton-Raphson no trabaja con intervalos donde nos asegure que encontraremos la raíz, y de hecho no tenemos ninguna garantía de que nos aproximaremos a dicha raíz. Desde luego, existen ejemplos donde

este método no converge a la raíz, en cuyo caso se dice que el método diverge. Sin embargo, en los casos donde si converge a la raíz lo hace con una rapidez impresionante, por lo cual es uno de los métodos preferidos por excelencia.
También observe que en el caso de que f(xi)=0, el método no se puede aplicar. De hecho, vemos geométricamente que esto significa que la recta tangente es horizontal y por lo tanto no intersecta al eje X en ningún punto, a menos que coincida con éste, en cuyo caso xi mismo es una raíz de f(x).

2.1 Codigo Método de Newton

```
module newton
! module parameters:
implicit none
integer, parameter :: maxiter = 20
real(kind=8), parameter :: tol = 1.d-14
contains
subroutine solve(f, fp, x0, x, iters, debug)
! Estimate the zero of f(x) using Newton's method.
! Input:
! f: the function to find a root of
! fp: function returning the derivative f'
! x0: the initial guess
! debug: logical, prints iterations if debug=.true.
! Returns:
! the estimate x satisfying f(x)=0 (assumes Newton converged!)
! the number of iterations iter
implicit none
real(kind=8), intent(in) :: x0
real(kind=8), external :: f, fp
logical, intent(in) :: debug
real(kind=8), intent(out) :: xínteger, intent(out) :: iters
! Declare any local variables:
real(kind=8) :: deltax, fx, fxprime
integer :: k
! initial guess
x = x0
if (debug) then
print 11, x
format('Initial guess: x = ', e22.15)
endif
! Newton iteration to find a zero of f(x)
do k=1,maxiter
! evaluate function and its derivative:
fx = f(x)
fxprime = fp(x)
if (abs(fx) ¡ tol) then
```

exit ! jump out of do loop
end if
! compute Newton increment x:
deltax = fx/fxprime
! update x:
x = x - deltax
if (debug) then
print 12, k,x
format('After', i3, ' iterations, x = ', e22.15)
end if
end do
if (k ¿ maxiter) then
! might not have converged
fx = f(x)
if (abs(fx) ¿ tol) then
print , '** Warning: has not yet converged'
end if
endi f
! number of iterations taken:
iters = k-1
end subroutine solve
end module newton
program test1
use newton, only: solve
use functions, only: $f_sqrt, fprime_sqrt$
$implicitnone$
$real(kind = 8) :: x, x0, fx$
$real(kind = 8) :: x0vals(3)$
$integer :: iters, itest$
$logical :: debug!setto.true.or.false.$
$print*,"Testroutineforcomputingzerooff"$
$debug = .true.$
$!valuestotestasx0 :$
$x0vals = (/1.d0, 2.d0, 100.d0/)$
$doitest = 1, 3$
$x0 = x0vals(itest)$
$print*,''!blankline$
$callsolve(f_sqrt, fprime_sqrt, x0, x, iters, debug)$
$print11, x, iters$
$format('solverreturnsx =', e22.15,' after', i3,' iterations')$
$fx = f_sqrt(x)$
$print12, fx$
$format('thevalueoff(x)is', e22.15)$
$if(abs(x - 2.d0) > 1d - 14)then$
$print13, x$
$format('*Unexpectedresult : x =', e22.15)$

$endifenddo$
end program tes
module functions
conins
real(kind=8) function $f_sqrt(x)$
$implicitnone$
$real(kind = 8), intent(in) :: x$
$f_sqrt = x * *2 - 4.d0$
end function f_sqrt
real(kind=8) function $fprime_sqrt(x)$
$implicitnone$
$real(kind = 8), intent(in) :: x$
$fprime_sqrt = 2.d0 * x$
end function $fprime_sqrt$
end module functions