PARCIAL 1

ANALISIS NUMERICO

1)

2)

**A)**



* Xn-1(inicial)=0; Xn-2(inicial)=0.5; tol=1.e-4

**i: 0**

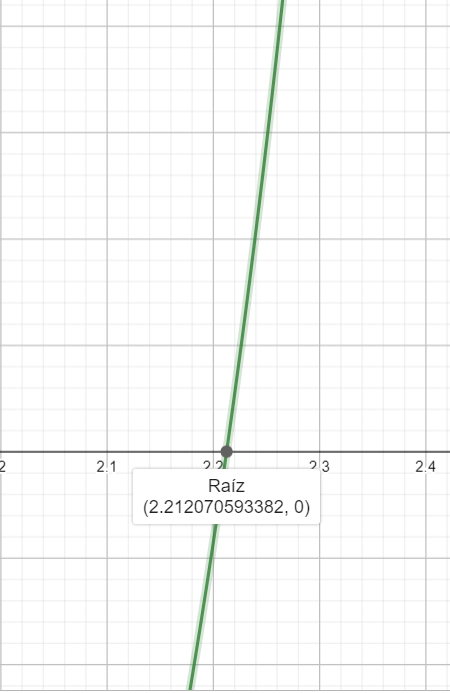
x= 2.202619 Error: 1.584022 **i: 1**

x= 2.213036 Error: 0.4707143 **i: 2**

x= 2.212049 Error: 0.04461944  **i: 3**

x= 2.212071 Error: 0.0009602412 **i: 4**

x= 2.212071 Error: 2.175155e-06

* Xn-1(inicial)=0; Xn-2(inicial)=0.5 ; tol=1.e-4

**i: 0**

x= 5.551115e-17 Error: 44.84681

**i: 1**

x= 0.2242341 Error: 100 **i: 2**

x= 0.2061048 Error: 8.796114 **i: 3**

x= 0.2119004 Error: 2.735038 **i: 4**

x= 0.212073 Error: 0.08137375 **i: 5**

x= 0.2120706 Error: 0.001118079 **i: 6**

x= 0.2120706 Error: 4.449772e-07

Código:

#Tx <- function(x) tan(pi\*x)

#Gx <- function(x) cos(pi\*x)

Fx<- function(x) tan(pi\*x)-cos(pi\*x)

sacarRaiz<-function(x1,x2,tol)

{

error<-tol+1

x0<-x1-((Fx(x1)\*(x1-x2))/(Fx(x1)-Fx(x2)))

x1<-x2

x2<-x0

i<-0

while(error>tol)

{

x3<-x1-((Fx(x1)\*(x1-x2))/(Fx(x1)-Fx(x2)))

error<-abs((x3-x2)/x3)\*100

cat("x=",x3,"\t Error:",error,"\ti:",i,"\n")

x1<-x2

x2<-x3

i<-i+1

}

}

sacarRaiz(2,2.3,1.e-4)

sacarRaiz(0,0.5,1.e-4)

**B)**

Utilicé el algoritmo de Newton (potenciado con Aitken)

Se puede apreciar que el algoritmo de Newton (potenciado con Aitken) converge más rápido.

Aunque en la primera raíz que se halló son iguales de rápidos, pero en la segunda raíz se puede evidenciar que el método de newton converge más rápido, utilizando 3 iteraciones menos.

sacarRaiz(2,2.3,1.e-4)

|  |  |
| --- | --- |
| iteracion | Error |
| 0 | 1,584022 |
| 1 | 0,4707143 |
| 2 | 0,04461944 |
| 3 | 0,0009602412 |
| 4 | 2,175155E-06 |

Método Newton Aitken

|  |  |
| --- | --- |
| Iteracion | Error |
| 0 | 0,159155 |
| 1 | 0,032728 |
| 2 | 0,012552 |
| 3 | 0,000996 |
| 4 | 0,000085 |

Código Newton (potenciado con Aitken)

newtonAitken<-function(r,tol){

i<-0

error<-1

while(i<=3 && error>tol){

if(r!=0){

bef=r

r<-r-((Fx(r))/Dx(r))

error<-(abs(bef-r))/abs(bef)

cat("R=",r,"\t Error:",error,"i:",i,"\n")

if(i==1)

{

x0=r

}

else

{

if(i==2)

{

x1=r

}

else

{

x2=r

}

}

}

i=i+1

}

while(error>tol){

if(r!=0){

x3=x2-(((x2-x1)^2)/(x2-(2\*x1)+x0))

x0=x1

x1=x2

error<-(abs(x2-x3))/abs(x2)

cat("R=",x3,"\t Error:",error,"i:",i,"\n")

r<-r-((Fx(r))/Dx(r))

x2=r

}

i=i+1

}

}