Parcial

Andres David Gomez B

28 de febrero de 2018

## 1. Matriz triangular

a)Primer punto usa como ejemplo una matriz triangular de 3x3

rm(list=ls())  
cat("recuerde que la matriz debe ser cuadrada para que el algoritmo funcione","\n")

## recuerde que la matriz debe ser cuadrada para que el algoritmo funcione

superior <- function(a) {  
y<-(dim(a))  
i<-1  
sum<-0  
j<-1  
while(i<=y[c(1)])  
{  
 j<-i  
 while(j<=y[c(1)])  
 {  
 sum<-sum+a[(i),(j)]  
 j<-j+1  
 }  
 i<-i+1  
}  
cat(sum)  
}  
a<-matrix(c(1,1,1,0,1,1,0,0,1), nrow=3, byrow=T)  
superior(a)

## 6

1. Dada la formula basta con el siguiente codigo que depende del tamaño:

rm(list=ls())  
cat("Ingrese el tamaño de la matriz cuadrada","\n")

## Ingrese el tamaño de la matriz cuadrada

eficiencia<- function(a) {  
 j<-1  
 i<-1  
 sum<-0  
 while(i<=a){  
 j<-i  
 while(j<=a)  
 {  
 sum<-sum+1  
 j<-j+1  
 }   
 i<-i+1  
 }  
cat(sum)  
}  
eficiencia(4)

## 10

1. Se prueba la funcion con varios valores:

rm(list=ls())  
eficiencia<- function(a) {  
 j<-1  
 i<-1  
 sum<-0  
 while(i<=a){  
 j<-i  
 while(j<=a)  
 {  
 sum<-sum+1  
 j<-j+1  
 }   
 i<-i+1  
 }  
cat(sum)  
}  
  
i<-2  
while(i<=25){  
 cat(eficiencia(i),"\n")  
 i<-i+1  
}

## 3   
## 6   
## 10   
## 15   
## 21   
## 28   
## 36   
## 45   
## 55   
## 66   
## 78   
## 91   
## 105   
## 120   
## 136   
## 153   
## 171   
## 190   
## 210   
## 231   
## 253   
## 276   
## 300   
## 325

Con estos datos podemos concluir que la notacion O() mas cercana es : notacion O(nlog(n)) ## 2. Numeros Aitken Primero se evalua la sucesion mediante el metodo de biseccion:

# Remueve todos los objetos creados  
rm(list=ls())  
# Halla la raiz de Fx  
Aitken <- function(a,b,c) {  
 x<-0  
 error<-1  
 while (error > 1.e-4) {  
 #if (Fx(x) == 0) break  
 y<-x  
 x<-(a-(((b-a)^2)/(c-2\*b+a)))  
 a<-b  
 b<-c  
 c<-x  
 error<-abs(y-x)  
 cat("X=",x,"\tE=",error,"\n")  
 }  
}  
#Aitken(1,1,1)

## 3 Secante

Se usa el metodo de la secante

# Metodo de secante

# Remueve todos los objetos creados

rm(list=ls()) Fx <- function(x) log(x+2)-sin(x) F1x <- function(x) (1/(x+2))-cos(x) # Halla la raiz de Fx secante <- function(x0,x1) { x<-seq(-2,0,0.001) plot(x,Fx(x),type=“l”,col=“blue”) abline(h=0,col=“blue”) x<-(Fx(x1)*x0-Fx(x0)*x1)/(Fx(x1)-Fx(x0)) error <-1 while (error > 1.e-7) { x0<-x1 x1<-x x<-(Fx(x1)*x0-Fx(x0)*x1)/(Fx(x1)-Fx(x0)) if (Fx(x) == 0) break error<-abs(Fx(x)/F1x(x)) points(rbind(c(x,0)),pch=19,cex=0.7,col=“red”) cat(“X=”,x,“”,“E=”,error,“”) } } secante(-2,0)

# Metodo de Newton

rm(list=ls()) Fx <- function(x) log(x+2)-sin(x) F1x <- function(x) (1/(x+2))-cos(x) # Metodo de Newton # Halla la raiz de Newton <- function(x0) { x<-seq(0,20,1) plot(x,Fx(x),type=“l”,col=“blue”) abline(h=0,col=“blue”) x<-x0-(Fx(x0)/F1x(x0)) error <-1 while (error > 1.e-5) { x<-x-(Fx(x)/F1x(x)) if (Fx(x) == 0) break error<-abs(Fx(x)/F1x(x)) points(rbind(c(x,0)),pch=19,cex=0.7,col=“red”) cat(“X=”,x,“”,“E=”,error,“”) } } Newton(1)