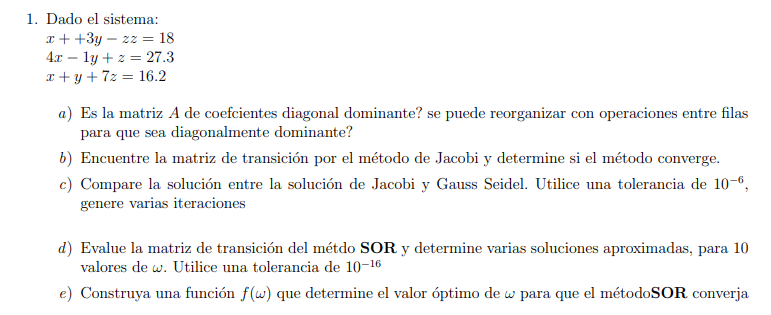
**Taller sistemas de ecuaciones**



EL método de SOR es una mejora a la iteración de Gauss-Seidel, donde se escoge un parametro W de ralajación tal que: si w = 1, La solición dada es equivalente a la solición que proporciona Gauss-Seidel y se dice que no hay relajación. si 0 < w < 1, El valor de la aproximación estará más cercano a la iteración k + 1 o a la actual, y se dice que hay subrelajación. Generalmente se usa para la convergencia de sistemas que no convergen por Gauss-Seidel. si 1 < w < 2, El valor de la aproximación será cercano a la iteración k + 1 y se dice que hay sobre-relajación y acelera la convergencia de sistemas convergentes y lentos como Gauss-Seidel.

**library(pracma)**

**library(pracma)**

**library(Matrix)**

**#Punto 1(xx)**

**n=3**

**a = matrix(c(1,3,-1,**

**4,-1,1,**

**1,1,7), nrow=n, byrow=TRUE)**

**print("a")**

**print(a)**

**b = matrix(c(18, 27.3, 16.2), nrow=n, byrow=TRUE)**

**print("b")**

**print(b)**

**diag1 <- function(M) {**

**M[col(M)!=row(M)] <- 0**

**return(M)**

**}**

**#T == matriz de transicion(jacobi)**

**#T = -D^-1(L + U)**

**D = diag1(A)**

**L = tril(A,k=-1,diag = FALSE)#triangular inferior**

**U = triu(A,k=1,diag = FALSE)#triangular superior**

**T = (-solve(D))%\*%(L+U)**

**print("T")**

**print(T)**

**print("Norma")**

**print(norm(T,"F"))**

**print("Gauss-Seidel:")**

**tol = 1e-9**

**sol = itersolve(A, b, x0=c(1,2,1,1), tol=0.0000000000000001 , method = "Gauss-Seidel")**

**print(sol)**

**jacobiPr <- function(A,b, x0, tol)**

**{**

**x\_k = matrix(x0)**

**it = 0**

**repeat**

**{**

**inn = matrix(b-((L+U)%\*%x\_k))**

**D1 = (solve(D))**

**xk1 = D1%\*%inn**

**cat("Error ",it," ",norm(xk1-x\_k,"F")/norm(x\_k),"\n")**

**x\_k = xk1**

**it = it + 1**

**if(it == tol)**

**break**

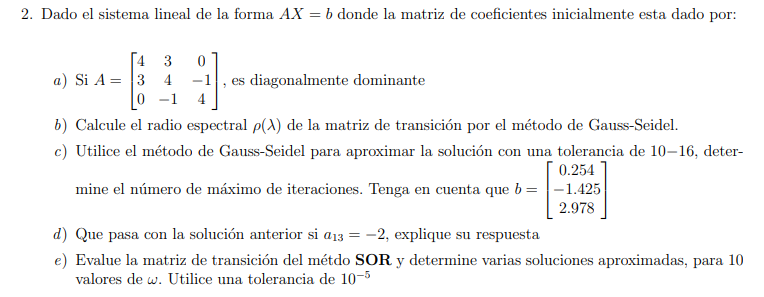
**}**

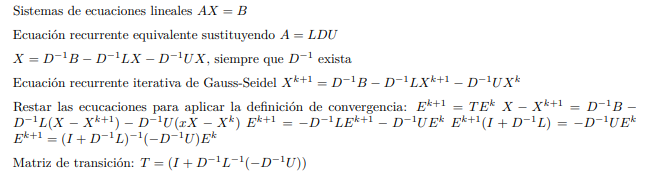
**cat("Solucion a 5 iteraciones: ",x\_k,"\n")**

**}**

**x0 = c(1,2,1,1)**

**jacobiPr(A, b, x0, 5)**





N <- 3

A = matrix(c(-8.1, -7, 6.123, -2,

-1, 4,-3, -1,

0, -1, -5, 0.6,

-1, 0.33, 6, 1/2), nrow=4, byrow=TRUE)

A

x0 <- rep(0, N)

b = c(4,5,6,8)

itersolve(A, b, tol=1e-9 , method = "Gauss-Seidel")

L = tril(A,k=-1,diag = FALSE)

U = triu(A,k=1,diag = FALSE)

L[lower.tri(L,diag=TRUE)] <- 0

U[upper.tri(U, diag = TRUE)] <- 0

#print (A)

D = diag(diag(A))

I=diag(1,nrow = nrow(A)) # Matriz identidad de 3x3

D1 <- solve(D,I) # Matriz inversa de A

T1 = D1 %\*% U

T2 = (I + (L %\*% D1))

T2<- solve(T2,I) # Matriz inversa de A

MatTG = T1+T2

MatTG

T3 = -solve(D)

T4 = T3%\*%U

T5= solve(D)

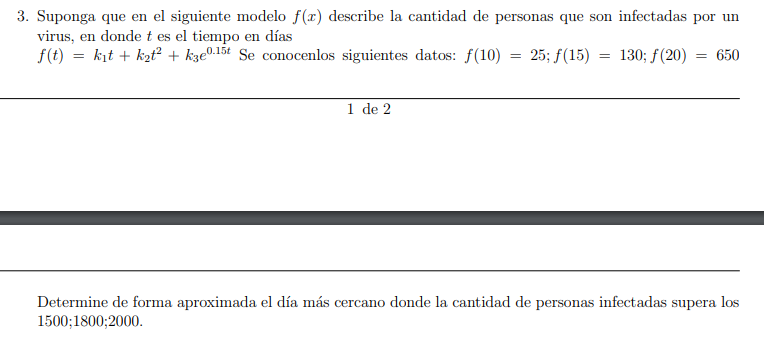
T6 = L%\*%T5

T7 = I + T6

T8 = solve(T7)

T = T8+T4#T4%\*%T8

T





trigexp = function(x) {

#Tamaño del vector que llega por parámetro

n = length(x)

#se crea un vector F vacío

F = rep(NA, n)

#Se enuncian las ecuaciones del sistema

F[10] = 25

F[15] = 130

F[20] = 650

#Se crea una secuencia de 2 hasta n-1

tn1 = 2:(n-1)

#Se evalúan tn1 ecuaciones

F[tn1] = -x[tn1-1] \* exp(x[tn1-1] - x[tn1]) + x[tn1] \*

( 4 + 3\*x[tn1]^2) + 2 \* x[tn1 + 1] + sin(x[tn1] -

x[tn1 + 1]) \* sin(x[tn1] + x[tn1 + 1]) - 8

#Se evalúa la última ecuación n

F[n] = -x[n-1] \* exp(x[n-1] - x[n]) + 4\*x[n] - 3

#Se retorna F

F

}

n = 10000

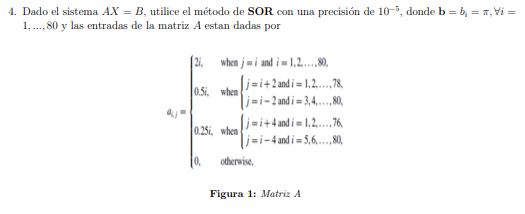
p0 = runif(n)

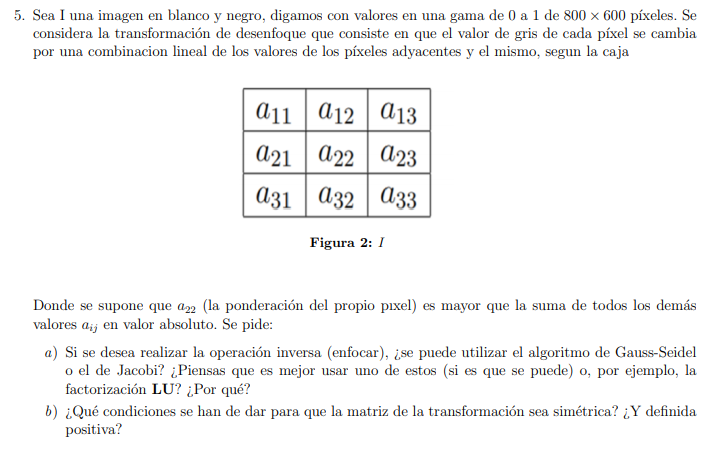
#se halla la solución del sistema trigexp usando BBsolve de la librería BB, utilizando n valores iniciales

sol = BBsolve(par=p0, fn=trigexp)

#Muestra el vector solución del sistema para cada n valores iniciales

sol$par





library(pracma)

library(Matrix)

diag1 <- function(M)

{

M[col(M)!=row(M)] <- 0

return(M)

}

crearMatrix = function()

{

datos = sample(1:20,36,replace=T) ## DAtos de la matrix aleatorios

A = matrix(datos,nrow = 6,ncol = 6)

while(1/rcond(A) < 1000)

{

datos = sample(1:20,36,replace=T) ## DAtos de la matrix aleatorios

A = matrix(datos,nrow = 6,ncol = 6)

}

return(A)

}

#1

A = crearMatrix()

A

b = c(1,5,2,3,4,5)

b

D = diag1(A)

L = tril(A,k=-1)#triangular inferior

U = triu(A,k=1)#triangular superior

I=diag(1,nrow = nrow(A))

T3 = -solve(D)

T4 = T3%\*%U

T5= solve(D)

T6 = L%\*%T5

T7 = I + T6

T8 = solve(T7)

MatTG = T4%\*%T8

normaG = norm(MatTG, type = c( "I"))

print("Convergencia Gauss")

print(normaG)

MatTJ = (-solve(D))%\*%(L+U)

normaJ = norm(MatTJ, type = c("I"))

print("Convergencia Jacobi")

print(normaJ)

print("Matriz transicion Gauss")

print(MatTG)

print("Matriz transicion Jacobi")

print (MatTJ)

X <- itersolve(A, b, method = "Jacobi")

print(X)

X <- itersolve(A, b, tol = 1e-9 , method = "Gauss-Seidel")

print(X)

solucion<- solve(A,b)

print(solucion)

#sor

w = 2

Qw <- D/w + L

IQw <- solve(Qw)

Transc <- eye(6) - IQw%\*%A

Transc

print(norm(Transc,type = c("I")))