

# Taller sistemas de ecuaciones

1. Dado el sistema:

$$x + 3y - z = 18$$

$$4x - y + z = 27.3$$

$$x + y + 7z = 16.2$$

- Es la matriz  $A$  de coeficientes diagonal dominante? se puede reorganizar con operaciones entre filas para que sea diagonalmente dominante?
- Encuentre la matriz de transición por el método de Jacobi y determine si el método converge.
- Compare la solución entre la solución de Jacobi y Gauss Seidel. Utilice una tolerancia de  $10^{-6}$ , genere varias iteraciones
- Evalue la matriz de transición del método **SOR** y determine varias soluciones aproximadas, para 10 valores de  $\omega$ . Utilice una tolerancia de  $10^{-16}$
- Construya una función  $f(\omega)$  que determine el valor óptimo de  $\omega$  para que el método **SOR** converja

A continuación, una breve descripción del método SOR general, investigando un poco, tenemos que el método de SOR es una mejora a la iteración de Gauss-Seidel, donde se escoge un parámetro  $\omega$  de relación tal que: si  $\omega = 1$ , La solución dada es equivalente a la solución que proporciona Gauss-Seidel y se dice que no hay relajación. si  $0 < \omega < 1$ , El valor de la aproximación estará más cercano a la iteración  $k + 1$  o a la actual, y se dice que hay subrelación. Generalmente se usa para la convergencia de sistemas que no convergen por Gauss-Seidel. si  $1 < \omega < 2$ , El valor de la aproximación será cercano a la iteración  $k + 1$  y se dice que hay sobre-relación y acelera la convergencia de sistemas convergentes y lentos como Gauss-Seidel.

Las librerías que fueron utilizadas fueron las siguientes:

```
library(pracma)
```

```
library(pracma)
```

```
library(Matrix)
```

La implementación del código fue la siguiente:

```
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)

```

2. Dado el sistema lineal de la forma  $AX = b$  donde la matriz de coeficientes inicialmente esta dado por:

- a) Si  $A = \begin{bmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix}$ , es diagonalmente dominante
- b) Calcule el radio espectral  $\rho(\lambda)$  de la matriz de transición por el método de Gauss-Seidel.
- c) Utilice el método de Gauss-Seidel para aproximar la solución con una tolerancia de  $10^{-16}$ , determine el número de máximo de iteraciones. Tenga en cuenta que  $b = \begin{bmatrix} 0.254 \\ -1.425 \\ 2.978 \end{bmatrix}$
- d) Que pasa con la solución anterior si  $a_{13} = -2$ , explique su respuesta
- e) Evalúe la matriz de transición del método **SOR** y determine varias soluciones aproximadas, para 10 valores de  $\omega$ . Utilice una tolerancia de  $10^{-5}$

Sistemas de ecuaciones lineales  $AX = B$

Ecuación recurrente equivalente sustituyendo  $A = LDU$

$X = D^{-1}B - D^{-1}LX - D^{-1}UX$ , siempre que  $D^{-1}$  exista

Ecuación recurrente iterativa de Gauss-Seidel  $X^{k+1} = D^{-1}B - D^{-1}LX^{k+1} - D^{-1}UX^k$

Restar las ecuaciones para aplicar la definición de convergencia:  $E^{k+1} = TE^k$   $X - X^{k+1} = D^{-1}B - D^{-1}L(X - X^{k+1}) - D^{-1}U(X - X^k)$   $E^{k+1} = -D^{-1}LE^{k+1} - D^{-1}UE^k$   $E^{k+1}(I + D^{-1}L) = -D^{-1}UE^k$   $E^{k+1} = (I + D^{-1}L)^{-1}(-D^{-1}U)E^k$

Matriz de transición:  $T = (I + D^{-1}L)^{-1}(-D^{-1}U)$

La implementación para este punto fue el siguiente:

```
N <- 3
##Definicion de la matriz
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)
##Solucion mediante el metodo de Gauss
itersolve(A, b, tol=1e-9, method = "Gauss-Seidel")
##Operacion sobre diagonales
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

```

3. Suponga que en el siguiente modelo  $f(x)$  describe la cantidad de personas que son infectadas por un virus, en donde  $t$  es el tiempo en días

$$f(t) = k_1 t + k_2 t^2 + k_3 e^{0.13t}$$

Se conocen los siguientes datos:  $f(10) = 25$ ;  $f(15) = 130$ ;  $f(20) = 650$

Determine de forma aproximada el día más cercano donde la cantidad de personas infectadas supera los 1500;1800;2000.

La implementación para este punto fue el siguiente:

```

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)
#Vector con valores iniciales
sol$par

```

5. Sea  $I$  una imagen en blanco y negro, digamos con valores en una gama de 0 a 1 de  $800 \times 600$  píxeles. Se considera la transformación de desenfoque que consiste en que el valor de gris de cada píxel se cambia por una combinación lineal de los valores de los píxeles adyacentes y el mismo, según la caja

$a_{11}$	$a_{12}$	$a_{13}$
$a_{21}$	$a_{22}$	$a_{23}$
$a_{31}$	$a_{32}$	$a_{33}$

Figura 2:  $I$

Donde se supone que  $a_{22}$  (la ponderación del propio píxel) es mayor que la suma de todos los demás valores  $a_{ij}$  en valor absoluto. Se pide:

- Si se desea realizar la operación inversa (enfocar), ¿se puede utilizar el algoritmo de Gauss-Seidel o el de Jacobi? ¿Piensas que es mejor usar uno de estos (si es que se puede) o, por ejemplo, la factorización LU? ¿Por qué?
- ¿Qué condiciones se han de dar para que la matriz de la transformación sea simétrica? ¿Y definida positiva?

La implementación para este punto fue el siguiente:

```
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)
}

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")))

```