**Taller sistemas de ecuaciones**

**Punto 1**

Un reloj en el medio

Descripción generada automáticamente con confianza media**A.** En el sistema de ecuaciones que nos presentan se puede evidenciar que la matriz no es diagonalmente dominante.

Un reloj en el medio

Descripción generada automáticamente con confianza mediaSe reorganizo de tal manera que cumpliera con la condición para lo cual se reorganizaron todas las filas quedando de la siguiente manera.

Texto

Descripción generada automáticamente

**B.** La matriz de transición encontrada a través del método de Jacobi es la siguiente:

La convergencia del método se verificó a través a través de 3 matrices las cuales son: la matriz diagonal, la matriz diagonal superior y la matriz diagonal inferior con lo cual se puede concluir que el método de Jacobi si converge.

**C.** Realizando la implementación de los métodos de Jacobi y Gauss-Seidel, se ha hecho la comparación de ambos métodos haciendo 50 iteraciones en cada uno y se ha podido llegar a la conclusión de que el método de Gauss-Seidel converge más rápido que el método de Jacobi.

Jacobi

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Gauss-Seidel

Texto

Descripción generada automáticamente

**D.**

Pantalla de computadora con letras

Descripción generada automáticamente con confianza mediaOmega 1.01 Omega 1.06

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Omega 1.15 Omega 1.23

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteOmega 1.3 Omega 1.37

Pantalla de computadora con letras

Descripción generada automáticamente con confianza bajaTexto

Descripción generada automáticamenteOmega 1.44 Omega 1.5

Omega 1.58 Omega 1.66

Texto

Descripción generada automáticamentePantalla de computadora con letras

Descripción generada automáticamente con confianza media

**E.** Para realizar el calculo del omega(w) óptimo la función utiliza un ciclo incrementando w en 0.01 iniciando en 0.01 y finalizando en 1.99 calculando la función sor\_solver en cada iteración y validando la cantidad de iteraciones utilizadas.

function [w] = w\_optimo (A, b , X0 , tol , num\_max\_it )

it\_aux = 100;

w= 0.01;

for i =0.01 : 0.01 :1.99

[ x , nit , err acum ] = sor\_solver (A, b ,i, X0 , tol ,T , num\_max\_it ) ;

if ( nit <it\_aux )

w=i ;

it\_aux=nit ;

endif

endfor

endfunction