//This procedure approximates the values of unknowns in a system of equations

//Validation   
converge ← FALSE  
cont ← ‘N’  
diagonal\_zero = False

WHILE converge = False DO

//Get the matrix of the equation  
INPUT A  
  
//Get shape of matrix  
num\_rows, num\_cols ← A.shape

//First Column  
k = 0

FOR i = 0 TO num\_rows – 1

IF A[i][k] == 0

THEN

OUTPUT ("WARNING! 0 Is Detected In Diagonal Of The Position Of Row Number", i+1, "and Column Number", k+1, "Please Enter A New Matrix")  
diagonal\_zero ← True  
BREAK

ELSE

k ← k + 1  
 diagonal\_zero ← False

ENDIF

ENDFOR

IF diagonal\_zero = False

THEN

D ← np.diag(np.abs(A))  
 S = np.sum(np.abs(A), axis = 1)- D  
 IF np.all(D > S):

THEN

OUTPUT “Matrix is diagonally dominant

ELSE  
 OUTPUT “Matric is NOT diagonally dominant

D ← np.diag(np.diag(A))  
D\_inv ← np.diag(1/np.diag(D))  
LU ← A – D  
norm ← np.linalg.norm(np.dot(D\_inv,LU))  
  
IF norm >= 1

THEN

OUTPUT "WARNING! This Iteration MAY NOT Converge, The Frobenius Norm of ||D^-1 times M|| Is ", norm  
INPUT cont  
IF cont = ‘Y’

THEN

Converge ← TRUE

ENDIF

ELSE

OUTPUT "This Iteration Will Converge, The Frobenius Norm of ||D^-1 times M|| Is ", norm, " < 1\n”  
Converge ← TRUE

ENDIF

ENDWHILE

//Get the answer of each equation  
INPUT b

//Get the initial guess  
INPUT x\_init

//Get TOL  
INPUT episilon

//Get maximum number of iterations  
INPUT max\_iterations

//Function Pseudocode Is In Next Page  
x = jacobi(A, b, x\_init, episilon, max\_iterations)

OUTPUT x

OUTPUT “Computed b by substituting the computed x value:” np.dot(A,x)

OUTPUT “Real b values” b

OUTPUT ‘Actual Solution from built in functions = %s’% solve(A,b)

FUNCTION jacobi(A: ARRAY, b:ARRAY, x\_init: ARRAY, epsilon: FLOAT, max\_iterations: INTEGER) RETURNS ARRAY

//Obtain diagonal of matrix  
D ← np.diag(np.diag(A))  
OUTPUT D  
  
//Get matrix without diagonal values  
LU ← A – D  
OUTPUT LU

x ← x\_init

//Get inverse matrix of diagonal matrix  
D\_inv = np.diag(1/np.diag(D))

//Start iteration  
FOR i = 0 TO max\_iterations – 1

//Obtain new x value  
x\_new ← np.dot (D\_inv, b – np.dot (LU,x))  
 OUTPUT x\_old, x\_new, np.linalg.norm(n\_new – x)

#OUTPUT “New x Value After” i + 1 “Iterations”

#OUTPUT x\_new

#OUTPUT “The Norm of Matrix After Finding Matrix of xNew – xOld

is” np.linalg.norm(n\_new – x)

IF np.linalg.norm(x\_new – x) < epsilon

THEN

OUTPUT “The Jacobi Method Succeeded After” i + 1

“Iterations”

RETURN x\_new

ENDIF

x ← x\_new

ENDFOR

IF i+1 = max\_iterations AND np.linalg.norm(x\_new – x) > epsilon  
 THEN

OUTPUT “The Jacobi Method Has Failed After”

max\_iterations “Iterations”

ENDIF

RETURN x