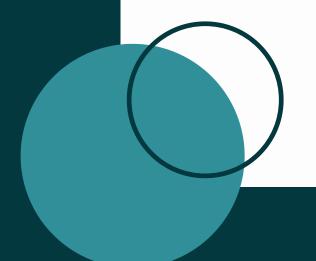
ANÁLISIS NUMÉRICO

Taler 2

David Andrés Ramírez Juan Felipe Arias C.



PUNTO 1 II

1. Dado el sistema:

i.
$$u - 8v - 2w = 1$$

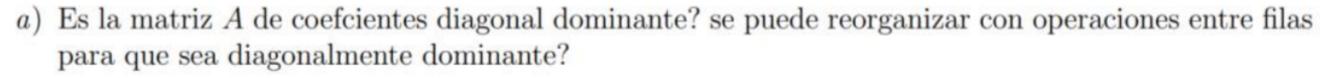
 $u + v + 5w = 4$
 $3u-v + w = -2$
 $u + 4v = 5$
ii. $v + w = 2$
 $2u + 3w = 0$

$$u + 4v = 5$$

ii. $v + w = 2$
 $2u + 3w = 0$

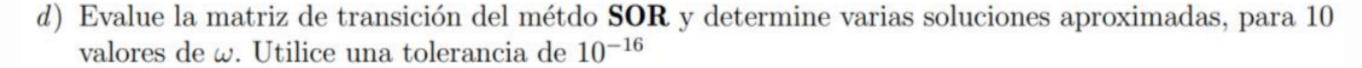
$$u + 3v - w = 18$$

iii. $4u-v + w = 27.34$
 $u + v + 7w = 16.2$

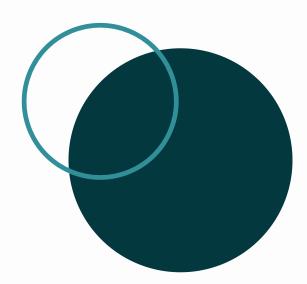




c) Compare la solución entre la solución de Jacobi y Gauss Seidel. Utilice una tolerancia de 10^{-6} , genere varias iteraciones



e) Construya una función $f(\omega)$ que determine el valor óptimo de ω para que el método**SOR** converja





```
# Si hay mas incognitas en el sistema de ecuaciones, calcula su convergencia y decide si ambas convergen o no.

# Si hay mas incognitas en el sistema de ecuaciones, calcula su convergencia y decide si ambas convergen o no.

# Si hay mas incognitas en el sistema de ecuaciones, indice + 1, cantidad_ecu, decimales)

# Si hay temp = convergencia(soluciones_temp, soluciones, indice + 1, cantidad_ecu, decimales)

# Si hag temp = convergencia(soluciones_temp, soluciones, indice + 1, cantidad_ecu, decimales)

# Si hag temp and flag

# Si hag temp and flag

# Si hag temp and flag
```







```
99
```





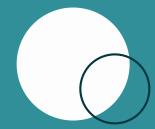


```
99
```

```
# Inicio del programa.
print("Bienvenido a GaussSei. Este programa resuelve un sistema de ecuaciones utilizando el metodo de Gauss-Seidel.")
# Se obtiene un numero correcto de ecuaciones.
cantidad_ecu = 0
while cantidad_ecu < 1:
   cantidad_ecu = int(input("\nEscribe el numero de ecuaciones (debe ser igual al numero de incognitas): "))
   if cantidad_ecu < 1:
       print("El valor debe ser mayor a 0. Intentalo de nuevo.")
# Inicializa dos vectores vacios. El primero contendra la matriz de coeficientes del sistema de ecuaciones, mientra que el segundo contendra las igualdades de cada ecuacion.
matriz = []
vector = []
# Se obtiene toda la informacion del sistema de ecuaciones y la almacena de forma correcta en ambos vectores.
for i in range(cantidad_ecu):
    print("\nEn la ecuacion " + str(i + 1) + ": ")
    fila = []
    for j in range(cantidad_ecu):
        fila.append(float(input("\tEscribe el coeficiente de la incognita " + str(j + 1) + ": ")))
    matriz.append(fila)
    vector.append(float(input("\tEscribe la igualdad de la ecuacion: ")))
```

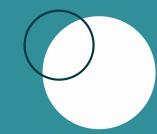






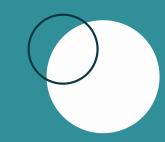
```
if revisar_matriz(matriz) == False:
       for i in range(len(matriz)):
           for j in range(i, len(matriz)):
               if abs(matriz[i][i]) < abs(matriz[j][i]):</pre>
                    fila = matriz[i][:]
                   matriz[i] = matriz[j][:]
                   matriz[j] = fila[:]
                    temp = vector[i]
                    vector[i] = vector[j]
                   vector[j] = temp
       # Si despues del arreglo de la matriz, esta continua siendo no diagonalmente dominante, entonces termina el programa.
       # El motivo es debido a que en esta instancia no se puede asegurar la convergencia del sistema.
       if revisar_matriz(matriz) == False:
           raise ValueError
except ValueError:
   print(
```





```
except ValueError:
  soluciones = []
  for i in range(cantidad_ecu):
      soluciones.append(0)
  soluciones_temp = []
  for i in range(cantidad_ecu):
      soluciones_temp.append(0)
  iteraciones = 0
  convergio = False
  # Itera el siguiente codigo hasta que los valores de cada incognita convergan en la precision deseada.
  while convergio == False:
      for i in range(cantidad_ecu):
           for j in range(cantidad_ecu):
              if (i != j):
                  if matriz[i][i] < 0:</pre>
                      suma = suma + (matriz[i][j] * soluciones_temp[j])
                       suma = suma + (-1 * (matriz[i][j]) * soluciones_temp[j])
          # Termina de calcular el despeje, y toma en cuenta los signos del coeficiente de la incognita actual.
```



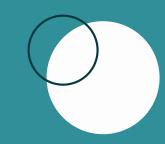


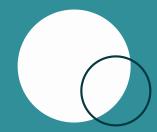


```
99
```

```
# Termina de calcular el despeje, y toma en cuenta los signos del coeficiente de la incognita actual.
       if matriz[i][i] < 0:</pre>
            suma = (suma + (-1 * vector[i])) / (-1 * matriz[i][i])
        else:
            suma = (suma + vector[i]) / (matriz[i][i])
       soluciones_temp[i] = suma
    # Calcula el error relativo entre el valor de las incognitas en la iteracion actual y la iteracion anteriror.
    iteraciones = iteraciones + 1
    indice = 0
   if convergencia(soluciones_temp, soluciones, indice, cantidad_ecu, decimales) == True:
        convergio = True
    else:
        soluciones = soluciones_temp[:]
for i in soluciones_temp:
   incognita_lista = [digito for digito in str(i)]
   incognita_cadena = ''
   for digito in incognita_lista[:incognita_lista.index('.') + 1 + decimales]:
        incognita_cadena = incognita_cadena + str(digito)
   print("\nEl resultado para la incognita " + str(soluciones_temp.index(i) + 1) + " es:", incognita_cadena)
print("\nLas iteraciones realizadas fueron:", (iteraciones))
```



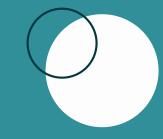




```
99
```

```
C:\Users\juanf\PycharmProjects\MetodoJacobi\venv\Scripts\python.exe C:/Users/juanf/PycharmProjects/MetodoJacobi/A.py
Bienvenido a GaussSei. Este programa resuelve un sistema de ecuaciones utilizando el metodo de Gauss-Seidel.
Escribe el numero de ecuaciones (debe ser igual al numero de incognitas):
En la ecuacion 1:
    Escribe el coeficiente de la incognita 1:
    Escribe el coeficiente de la incognita 2:
    Escribe el coeficiente de la incognita 3:
    Escribe la igualdad de la ecuacion:
En la ecuacion 2:
    Escribe el coeficiente de la incognita 1:
    Escribe el coeficiente de la incognita 2:
    Escribe el coeficiente de la incognita 3:
    Escribe la igualdad de la ecuacion:
En la ecuacion 3:
    Escribe el coeficiente de la incognita 1:
    Escribe el coeficiente de la incognita 2:
    Escribe el coeficiente de la incognita 3:
    Escribe la igualdad de la ecuacion:
Error: El sistema no es diagonalmente dominante. El programa ha terminado porque no se puede asegurar convergencia del metodo en esta instancia.
Process finished with exit code 0
```







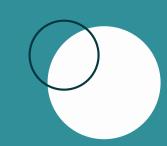
b) Encuentre la matriz de transición por el método de Jacobi y determine si el método converge.

```
99
```

```
def matrizTransicionGS(a):
 #d (diagonal).
 d = np.diag(a)
 d = np.diag(d)
 if(not(np.linalg.det(d))): #Evaluando que tenga inversa...
   return
 dInv = np.linalg.inv(d)
 #l (lower): matriz inferior triangular.
 l = np.tril(a)
 #u (upper): matriz superior triangular.
 u = np.triu(a)
 ident = np.identity(len(a))
 dInv_l = np.dot(dInv, l)
 _dInv_u = np.dot(-dInv, u)
 if(not(np.linalg.det(_dInv_u))): #Evaluando que tenga inversa...
   return
 else:
   if(np.linalg.det(ident + dInv_l)): #Evaluando que tenga inversa...
     t = np.dot(np.linalg.inv(ident + dInv_l), _dInv_u)
     return(t)
   return
```

```
def jacobi(A, b, x, n):
  y=[]
  D = np.diag(A)
  R = A-np.diagflat(D)
  for i in range(n):
      y=x
      x = (b-np.dot(R_x))/D
      print("({})->{}".format(i+1, x))
     if(i > 0):
        e1 = (x-y)
        print("Vector error: "_abs(e1))
  return x
```







b) Encuentre la matriz de transición por el método de Jacobi y determine si el método converge.



----- JACOBI -----(1)->[-1.4365 -0.35625 1.2445] (2)->[0.3306875 1.03225 0.6554375] Vector error: [1.7671875 1.3885 0.5890625] (3)->[-0.7106875 -0.44040625 1.0025625] Vector error: [1.041375 1.47265625 0.347125] (4)->[0.39380469 0.42740625 0.63439844] Vector error: [1.10449219 0.8678125 0.36816406] (5)->[-0.25705469 -0.49300391 0.85135156] Vector error: [0.65085937 0.92041016 0.21695312] (6)->[0.43325293 0.04937891 0.62124902] Vector error: [0.69030762 0.54238281 0.23010254] (7)->[0.02646582 -0.52587744 0.75684473] Vector error: [0.40678711 0.57525635 0.1355957] (8)->[0.45790808 -0.18688818 0.61303064] Vector error: [0.43144226 0.33898926 0.14381409] (9)->[0.20366614 -0.5464234 0.69777795] Vector error: [0.25424194 0.35953522 0.08474731] (10)->[0.47331755 -0.33455511 0.60789415] Vector error: [0.26965141 0.21186829 0.0898838] (11)->[0.31441634 -0.55926463 0.66086122] Vector error: [0.15890121 0.22470951 0.05296707] (12)->[0.48294847 -0.42684695 0.60468384] Vector error: [0.16853213 0.13241768 0.05617738] $(13) \rightarrow [0.38363521 - 0.56729039 0.63778826]$ Vector error: [0.09931326 0.14044344 0.03310442] (14)->[0.48896779 -0.48452934 0.6026774] Vector error: [0.10533258 0.08276105 0.03511086] (15)->[0.42689701 -0.57230649 0.62336766] Vector error: [0.06207079 0.08777715 0.02069026] (16)->[0.49272987 -0.52058084 0.60142338] Vector error: [0.06583286 0.05172566 0.02194429] $(17) \rightarrow [0.45393563 - 0.57544156 0.61435479]$ Vector error: [0.03879424 0.05486072 0.01293141] $(18) \rightarrow [0.49508117 - 0.54311302 0.60063961]$ Vector error: [0.04114554 0.03232853 0.01371518] $(19) \rightarrow [0.47083477 - 0.57740097 0.60872174]$ Vector error: [0.0242464 0.03428795 0.00808213] (20)->[0.49655073 -0.55719564 0.60014976] Vector error: [0.02571596 0.02020533 0.00857199] (21)->[0.48139673 -0.57862561 0.60520109]

(20) -> [0.49655073 -0.55719564 0.60014976] Vector error: [0.02571596 0.02020533 0.00857199] (21)->[0.48139673 -0.57862561 0.60520109] Vector error: [0.015154 0.02142997 0.00505133] (22)->[0.49746921 -0.56599728 0.5998436] Vector error: [0.01607248 0.01262833 0.00535749] (23)->[0.48799796 -0.57939101 0.60300068] Vector error: [0.00947125 0.01339373 0.00315708] (24)->[0.49804325 -0.5714983 0.59965225] Vector error: [0.0100453 0.00789271 0.00334843] (25)->[0.49212372 -0.57986938 0.60162543] Vector error: [0.00591953 0.00837108 0.00197318] (26)->[0.49840203 -0.57493644 0.59953266] Vector error: [0.00627831 0.00493294 0.00209277] $(27) \rightarrow [0.49470233 - 0.58016836 0.60076589]$ Vector error: [0.00369971 0.00523193 0.00123324] (28)->[0.49862627 -0.57708527 0.59945791] Vector error: [0.00392394 0.00308309 0.00130798] (29)->[0.49631395 -0.58035523 0.60022868] Vector error: [0.00231232 0.00326995 0.00077077] (30)->[0.49876642 -0.5784283 0.59941119] Vector error: [0.00245247 0.00192693 0.00081749] RESULTADO -> [0.499 -0.58066667 0.59933333]

No converge







c) Compare la solución entre la solución de Jacobi y Gauss Seidel. Utilice una tolerancia de 10⁻⁶, genere varias iteraciones

```
99
```

```
ldef seidel(a, x, b):
    \underline{n} = len(a)
    for j in range(0, n):
        d = b[j]
        for i in range(0, n):
             if(j != i):
                 d-=a[j][i] * x[i]
        x[j] = d / a[j][j]
        para=(b[j]-d)/ a[j][j]
        if x[j]==para:
           print("FIN.")
    return x
```

```
def jacobi(A, b, x, n):
  <u>y=[]</u>
  D = np.diag(A)
  R = A-np.diagflat(D)
  for i in range(n):
      y=x
      x = (b-np.dot(R,x))/D
      print("({})->{}".format(i+1, x))
      if(i > 0):
        e1 = (x-y)
        print("Vector error: ",abs(e1))
  return x
```







c) Compare la solución entre la solución de Jacobi y Gauss Seidel. Utilice una tolerancia de 10⁻⁶, genere varias iteraciones

99

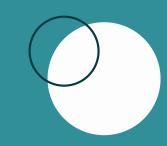
Seidel

```
----- SIDEL ------
0 [0.7986652720263248, -0.8054154812223759, 0.543146129694406]
1 [0.939134675763985, -0.9248144743993872, 0.5132963814001532]
2 [1.013759046499617, -0.9882451895246743, 0.4974387026188315]
3 [1.0534032434529215, -1.0219427569349833, 0.48901431076625423]
4 [1.0744642230843646, -1.03984458962171, 0.4845388525945726]
5 [1.0856528685135687, -1.0493549382365335, 0.4821612654408667]
6 [1.0915968363978334, -1.0544073109381582, 0.4808981722654605]
7 [1.094754569336349, -1.0570913839358966, 0.4802271540160259]
8 [1.0964321149599354, -1.0585172977159452, 0.47987067557101376]
9 [1.0973233110724656, -1.0592748144115958, 0.4796812963971011]
10 [1.0977967590072475, -1.0596772451561605, 0.47958068871095993]
11 [1.0980482782226004, -1.0598910364892105, 0.47952724087769744]
12 [1.0981818978057567, -1.0600046131348932, 0.47949884671627674]
13 [1.0982528832093084, -1.060064950727912, 0.47948376231802203]
14 [1.098290594204945, -1.0600970050742031, 0.47947574873144927]
15 [1.098310628171377, -1.0601140339456705, 0.4794714915135824]
16 [1.098321271216044, -1.0601230805336375, 0.4794692298665907]
17 [1.0983269253335235, -1.0601278865334949, 0.47946802836662633]
18 [1.0983299290834343, -1.0601304397209192, 0.47946739006977024]
19 [1.0983315248255745, -1.0601317961017385, 0.47946705097456543]
20 [1.0983323725635865, -1.0601325166790485, 0.4794668708302379]
21 [1.0983328229244054, -1.0601328994857446, 0.4794667751285639]
22 [1.0983330621785903, -1.0601331028518017, 0.4794667242870496]
23 [1.0983331892823762, -1.0601332108900199, 0.4794666972774951]
24 [1.0983332568062625, -1.060133268285323, 0.4794666829286693]
```

Jacobi

```
(20) \rightarrow [0.49655073 - 0.55719564 0.60014976]
Vector error: [0.02571596 0.02020533 0.00857199]
(21) \rightarrow [0.48139673 - 0.57862561 0.60520109]
Vector error: [0.015154 0.02142997 0.00505133]
(22)->[ 0.49746921 -0.56599728  0.5998436 ]
Vector error: [0.01607248 0.01262833 0.00535749]
(23) \rightarrow [0.48799796 - 0.57939101 0.60300068]
Vector error: [0.00947125 0.01339373 0.00315708]
(24)->[ 0.49804325 -0.5714983  0.59965225]
Vector error: [0.0100453 0.00789271 0.00334843]
(25)->[ 0.49212372 -0.57986938  0.60162543]
Vector error: [0.00591953 0.00837108 0.00197318]
(26) -> [0.49840203 - 0.57493644 0.59953266]
Vector error: [0.00627831 0.00493294 0.00209277]
(27) \rightarrow [0.49470233 - 0.58016836 0.60076589]
Vector error: [0.00369971 0.00523193 0.00123324]
(28)->[ 0.49862627 -0.57708527 0.59945791]
Vector error: [0.00392394 0.00308309 0.00130798]
(29)->[ 0.49631395 -0.58035523  0.60022868]
Vector error: [0.00231232 0.00326995 0.00077077]
(30) \rightarrow [0.49876642 - 0.5784283 0.59941119]
Vector error: [0.00245247 0.00192693 0.00081749]
RESULTADO -> [ 0.499
                          -0.58066667 0.59933333]
```







d) Evalue la matriz de transición del métdo ${\bf SOR}$ y determine varias soluciones aproximadas, para 10 valores de ω . Utilice una tolerancia de 10^{-16}

SOR

```
def sor(a,b,w,x):
    for i in range(len(a)):
        sigma=0
        for j in range(len(a)):
            if j != i:
                sigma+=a[i][j]*x[j]
        if(a[i][i]!=0):
                x[i] += w*(((b[i]-sigma)/a[i][i])-x[i])
        return x
```

```
print("----- SOR
w≅0
for j in range(0,15):
  w + = 0.125
 print("W = "_w)
 for i in range(0, 25):
     x = sor(a_x b_x w_x x)
     print(i,x)
  print(" ")
print('\n\n')
```





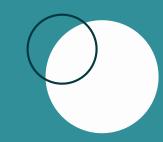


d) Evalue la matriz de transición del mét
do SOR y determine varias soluciones aproximadas, para 10 valores de ω . Utilice una tolerancia de 10^{-16}

```
99
```

```
W = 0.375
                                                                    W = 0.25
W = 0.125
                                                                                                                                      0 [0.4574615240038942, -0.5385518532225587, 0.6121997982745402]
                                                                    0 [0.3012479369324207, -0.38310765332030616, 0.6424644916476507]
0 [0.0079375, -0.045275390625, 0.09164764404296875]
                                                                    1 [0.3136436376968729, -0.3950473913304151, 0.6432829067775871]
                                                                                                                                       1 [0.46119366122127853, -0.5425056443943457, 0.6109524697596177]
1 [0.019127380371093752, -0.08307641983032227, 0.1706580504179001]
                                                                                                                                       2 [0.4646382507492088, -0.5460624917297169, 0.6098394350001001]
                                                                   2 [0.32517911414710754, -0.4061139457267948, 0.6432050584752658]
2 [0.03246237218379975, -0.11493340066820384, 0.23879662534478124] 3 [0.33590570043410467, -0.4164299619717866, 0.6425019212332127]
                                                                                                                                      3 [0.46779148251723834, -0.5492767147577869, 0.60884245486652]
3 [0.047117081973468894, -0.14205280747766666, 0.2975703969430065]
                                                                   4 [0.3458848931952885, -0.42608201887588076, 0.641371314745167]
                                                                                                                                       4 [0.47066625259890155, -0.5521876001233217, 0.6079464467800136]
4 [0.06248239742781653, -0.1653861063973472, 0.3482682815002136]
                                                                   5 [0.35517904843569403, -0.4351343785670303, 0.6399575873984358]
                                                                                                                                       5 [0.4732816704089977, -0.5548264904939804, 0.6071390457536979]
5 \quad \begin{bmatrix} 0.07811454522409075 \,,\, -0.18568394791555562 \,,\, 0.39199462294032583 \end{bmatrix} \quad \begin{bmatrix} 0.3638469823080887 \,,\, -0.44363724389563713 \,,\, 0.6383658628053495 \end{bmatrix}
                                                                                                                                      6 [0.47565849445705555, -0.5572199725853755, 0.6064100311661823]
6 [0.09369559718816275, -0.20353883469561623, 0.4296972064885471]
                                                                   7 [0.3719422199614985, -0.45163173273917445, 0.6366724138078137] 7 [0.4778171763252966, -0.5593913732855198, 0.6057508282333465]
7 [0.10900291329235644, -0.21941871577705552, 0.46219072080944573]
                                                                   8 [0.3795126148597191, -0.4591528889775898, 0.6349322547947609]
                                                                                                                                       8 [0.4797770589398628, -0.56136151598341, 0.6051541255223969]
8 [0.12388555373491084, -0.2336934369422763, 0.4901764608038189]
                                                                   9 [0.3866006278280874, -0.4662315185262862, 0.6331847211881778]
                                                                                                                                      9 [0.4815560882077483, -0.5631491480303358, 0.6046135958236541]
9 [0.13824611923138538, -0.2466555666023148, 0.5142589167470192]
                                                                    10 [0.39324388059474424, -0.4728953214319681, 0.6314575833016354] 10 [0.48317075301337464, -0.5647712171955039, 0.6041236957777053]
10 [0.15202681369642923, -0.25853679341272134, 0.5349597773594943]
                                                                    11 [0.39947578321455224, -0.4791696014703524, 0.6297700873843295] 11 [0.4846361254695946, -0.5662430745563536, 0.6036795216214077]
11 [0.1651987863668182, -0.2695208374155362, 0.552729779020322]
                                                                   12 [0.40532613768660525, -0.48507772145748224, 0.6281352079471545 12 [0.4859659431374711, -0.5675786379531278, 0.6032767037052741]
12 [0.17775401657867243, -0.2797536161984596, 0.56795875613658]
                                                                   13 [0.41082167603823183, -0.490641404853583, 0.6265613181570169] 13 [0.48717270638523663, -0.5687905314191832, 0.6029113274952479]
13 [0.18969916602494397, -0.28935124985922256, 0.5809841850614068]
                                                                    14 [0.4159865204387207, -0.4958809438376338, 0.6250534296279106] 14 [0.4882677784524182, -0.5698902078740526, 0.6025798726963375]
14 [0.2010509499461281, -0.2984063644011003, 0.5920984630411966]
                                                                   15 [0.42084256729859687, -0.5008153498949679, 0.6236141128524975] 15 [0.4892614824973387, -0.5708880588083778, 0.6022791649219256]
15 [0.21183267786546522, -0.30699305543081273, 0.6015551221788341]
                                                                    16 [0.4254098035792541, -0.505462468539055, 0.6222441803556822]
                                                                                                                                       16 [0.49016319310069295, -0.5717935131033756, 0.602006336222762]
16 [0.22207169207892077, -0.3151707970662714, 0.6095741444981588]
                                                                   17 [0.4297065655355134, -0.5098390711699698, 0.6209431933186386]
                                                                                                                                      17 [0.4909814212482575, -0.5726151263947983, 0.6017587920397139]
17 [0.23179749279401862, -0.32298752036685924, 0.6163465164244246]
                                                                    18 [0.4337497499960044, -0.5139609319193132, 0.6197098367440219] 18 [0.491723892578698, -0.5733606620307845, 0.6015341829594352]
18 [0.24104038622915935, -0.33048203789172226, 0.6220381381872553]
                                                                   19 [0.4375549872318745, -0.51784289424896, 0.6185421966674565]
                                                                                                                                       19 [0.4923976190578444, -0.5740371644768121, 0.6013303801799459]
19 [0.2498305290028634, -0.33768595343092367, 0.626793184869132]
                                                                   20 [0.4411367830955859, -0.5214989302254264, 0.6174379643615032] 20 [0.49300896442025616, -0.5746510258993347, 0.6011454539344036]
20 [0.25819727101165457, -0.34462516638224044, 0.6307370003110455]
                                                                   21 [0.4445086367389569, -0.5249421942850303, 0.616394586128313]
                                                                                                                                       21 [0.493563703796848, -0.5752080465735974, 0.6009776543427274]
21 [0.26616872148353277, -0.35132105696382143, 0.6339785922420454]
                                                                   22 [0.4476831389826609, -0.5281850726400021, 0.6154093725562346] 22 [0.49406707797185423, -0.5757134896934517, 0.6008253943054436]
22 [0.27377148038844945, -0.357791420122197, 0.6366127863329711]
23 [0.28103049097634925, -0.36405120156304976, 0.6387220879925044] 23 [0.45067205535699606, -0.5312392290746737, 0.6144795776000089] 23 [0.4945238427086922, -0.5761721311040917, 0.6006872341498937]
                                                                   24 [0.45348639696924836, -0.5341156476377388, 0.613602455222648] 24 [0.4949383135659584, -0.5765883044289306, 0.6005618678034713]
24 [0.2879689797508415, -0.3701130779695442, 0.6403782933068931]
```







d) Evalue la matriz de transición del métdo ${\bf SOR}$ y determine varias soluciones aproximadas, para 10 valores de ω . Utilice una tolerancia de 10^{-16}

```
99
```

```
W = 0.625
W = 0.5
                                                                                                                                                                               W = 0.75
                                                                                                                                                                              0 [0.498999244419779, -0.580665963036485, 0.5993335310521185]
                                                                                         0 [0.4988988428063806, -0.5805691723670325, 0.5993617096257995]
0 [0.4954397709438282, -0.577138832842967, 0.6003885797963648]
                                                                                        1 [0.4989163655994392, -0.5805864688833431, 0.5993565053466525] 1 [0.49899941531296754, -0.5806661248003933, 0.5993334843629559]
1 [0.49589694778802673, -0.5776071993674479, 0.6002433899772515]
                                                                                        2 [0.49893104438885677, -0.5806006489281158, 0.5993523381099767] 2
                                                                                                                                                                                  [0.4989995490284631, -0.5806662492105545, 0.59933344936376]
2 [0.4963011736568063, -0.5780111160578698, 0.6001203054813921]
                                                                                        3 [0.4989431958308756, -0.5806123135640825, 0.5993489527968534]
                                                                                                                                                                                  [0.4989996524380527, -0.5806663450433384, 0.5993334226453141]
3 [0.49665475535010434, -0.57836105310005, 0.6000150211031898]
                                                                                        4 [0.498953220419742, -0.5806219157837766, 0.599346182957605]
                                                                                                                                                                               4 [0.49899973219639104, -0.5806664188753081, 0.5993334021222083]
4 [0.49696277258757093, -0.5786646886324653, 0.5999244244725368]
                                                                                         5 [0.49896148068104856, -0.580629821401032, 0.5993439090151906] 5
                                                                                                                                                                                  [0.4989997936664586, -0.580666475758296, 0.5993333863258716]
5 [0.49723064453096, -0.5789282829562756, 0.599846176866734]
                                                                                        6 [0.49896828403712695, -0.5806363303841667, 0.5993420392581704] 6 [0.49899984103065614, -0.5806665195832172, 0.5993333741596147]
6 [0.49746342837408336, -0.5791571550100774, 0.5997784440571073]
                                                                                        7 [0.49897388638150075, -0.5806416895013019, 0.5993405007372355] 7 [0.4989998775232237, -0.5806665533476899, 0.5993333647872119]
7 [0.4976656473158207, -0.579355889741333, 0.599719735810887]
                                                                                        8 [0.49897849934679805, -0.5806461018916068, 0.5993392343558998] 8 [0.4989999056388815, -0.5806665793611911, 0.5993333575665796]
8 [0.4978412823109102, -0.5795284587608969, 0.5996688105603314]
                                                                                        9 [0.49898229751673995, -0.580649734802215, 0.5993381918206163] 9 [0.4989999273003904, -0.5806665994030337, 0.5993333520035761]
9 [0.4979938131907914, -0.5796783080069539, 0.5996246167792965]
                                                                                        10 [0.4989854247573158, -0.5806527259338511, 0.599337333505567]
                                                                                                                                                                              10 [0.49899994398930403, -0.5806666148440714, 0.5993333477176306]
10 [0.4981262720980034, -0.5798084289428161, 0.5995862547717963]
                                                                                        11 [0.4989879995654861, -0.580655188661271, 0.599336626836264]
                                                                                                                                                                               11 [0.4989999568471162, -0.580666626740465, 0.5993333444155704]
11 [0.49824129690255775, -0.5799214189633928, 0.599552950015474]
                                                                                       12 [0.49834118056255117, -0.5800195334407188, 0.5995240333276473]
                                                                                        13 [0.4989918649758047, -0.5806588857983317, 0.5993355659734798] 13 [0.49899997438540417, -0.5806666429673584, 0.5993333399115034]
13 [0.4984279153215451, -0.5801047307999829, 0.5994989253138258]
                                                                                       14 [0.49899330208389475, -0.5806602603428438, 0.599335171561485614 [0.49899998026549014, -0.5806666484077709, 0.5993333384014188]
14 [0.49850323171076616, -0.5801787116273005, 0.5994771237035004]
                                                                                        15 \quad [0.49899448531716856, \quad -0.5806613920645071, \quad 0.5993348468254779 \\ 15 \quad [0.4989999847957437, \quad -0.5806666525992825, \quad 0.5993333372379892]
15 [0.4985686327156208, -0.5802429526190704, 0.5994581927743664]
                                                                                        16 \ [0.4989954595241759, -0.5806623238596667, \ 0.5993345794564813] \\ 16 \ [0.4989999882860323, -0.5806666558285909, \ 0.5993333363416365] \\ 16 \ [0.4989999882860323, -0.5806666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.5806666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.5806666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.58066666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.58066666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.58066666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.58066666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.58066666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.58066666558285909, \ 0.59933333363416365] \\ 16 \ [0.4989999882860323, -0.5806666655828590] \\ 16 \ [0.4989999882860323, -0.580666665582859] \\ 16 \ [0.4989999882860323, -0.5806666655828] \\ 16 \ [0.4989999882860323, -0.5806666665] \\ 16 \ [0.4989999882860323, -0.5806666665] \\ 16 \ [0.4989999882860323, -0.5806666666] \\ 16 \ [0.4989999882860323, -0.580666666] \\ 16 \ [0.4989999882860323, -0.5806666666] \\ 16 \ [0.49899998828603] \\ 16 \ [0.49899998828603] \\ 16 \ [0.49899998828603] \\ 16 \ [0.49899998828603] \\ 16 \ [0.49899998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.4989998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.498998828603] \\ 16 \ [0.49898828603] \\ 16 \ [0.4989888828] \\ 16 \ [0.4989888288] \\ 16 \ [0.4989888288] \\ 16 \ [0.498988828] \\ 16 \ [0.498988888] \\
16 [0.4986254235899618, -0.5802987360589751, 0.5994417543798114]
                                                                                        17 \ [0.4989962616307847, \ -0.5806630910467301, \ 0.5993343593201289] \ 17 \ [0.49899999997509046, \ -0.5806666583165793, \ 0.5993333356510505]
17 [0.4986747378170966, -0.5803471754134224, 0.5994274802632279]
                                                                                        18 [0.498996922039699, -0.5806637227048626, 0.5993341780724136]
                                                                                                                                                                               18 [0.49899999304684844, -0.5806666602334252, 0.5993333351189954]
18 [0.49871755968858167, -0.5803892375570259, 0.5994150854369857]
                                                                                       19 [0.4989974657827914, -0.5806642427761923, 0.5993340288433752] 19 [0.49899999464301376, -0.58066666171024, 0.5993333347090789]
19 [0.49875474392817554, -0.5804257620719555, 0.5994043224594985]
                                                                                        20 [0.49878703274107106, -0.5804574780064421, 0.599394976478944]
                                                                                        21 [0.498998282069944, -0.580665023526418, 0.5993338048152548]
21 [0.49881507062295133, -0.5804850184269599, 0.599386860936102]
                                                                                                                                                                               21 [0.49899999682021173, -0.5806666637246417, 0.5993333341499454]
                                                                                       22 [0.4989985855542375, -0.580665313798572, 0.5993337215246937]
22 [0.49883941722158565, -0.5805089330545619, 0.5993798138362307]
                                                                                                                                                                               22 [0.4989999975501639, -0.5806666644000129, 0.599333333962484]
                                                                                        23 [0.49899883542591966, -0.580665552792131, 0.5993336529479897] 23 [0.49899999811254825, -0.5806666649203458, 0.5993333338180562]
23 [0.49886055850625355, -0.5805296992375972, 0.5993736945134157]
                                                                                        24 [0.4988789164672257, -0.5805477314798313, 0.5993683808217289]
```







d) Evalue la matriz de transición del métdo **SOR** y determine varias soluciones aproximadas, para 10 valores de ω . Utilice una tolerancia de 10^{-16}

```
99
```

```
W = 0.875
                                                                                                                           W = 1.125
                                                             W = 1.0
0 [0.49899999893528685, -0.580666665718077, 0.5993333335875186]
                                                                                                                           0 [0.4989999999983047, -0.580666666665246, 0.5993333333333689]
1 [0.4989999992443989, -0.5806666659966266, 0.5993333335116778]
                                                                                                                           1 [0.4989999999999983, -0.5806666666666666, 0.5993333333333333]
                                                             1 [0.4989999999998934, -0.5806666666665778, 0.599333333333556]
2 [0.4989999994658361, -0.5806666661933538, 0.5993333334591636]
                                                             2 [0.498999999999334, -0.58066666666666112, 0.59933333333333473]
                                                                                                                           2 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
3 [0.498999999622618, -0.5806666663323202, 0.5993333334222004]
                                                             3 [0.4989999999995837, -0.580666666666632, 0.59933333333333342]
                                                                                                                           3 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
4 [0.4989999973341236, -0.5806666664304856, 0.5993333333961064]
                                                             4 [0.49899999999974, -0.5806666666666645, 0.59933333333333388]
                                                                                                                           4 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
5 [0.4989999998116827, -0.5806666664998292, 0.5993333333776758]
                                                             5 [0.498999999999838, -0.580666666666531, 0.5993333333333368]
                                                                                                                           5 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
6 [0.498999998669733, -0.580666665488134, 0.5993333333646566]
                                                             6 [0.49899999999998984, -0.58066666666666582, 0.599333333333355]
                                                                                                                           6 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
7 [0.4989999990603044, -0.5806666665834156, 0.5993333335546]
                                                             7 [0.4989999999999367, -0.58066666666666613, 0.59933333333333347]
                                                                                                                           7 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
8 [0.498999999336203, -0.5806666666078584, 0.5993333333489635]
                                                             8 [0.49899999999996, -0.5806666666666633, 0.5993333333333342]
                                                                                                                           8 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
9 [0.498999999531096, -0.5806666666251248, 0.5993333333443744]
                                                             9 [0.498999999999975, -0.5806666666666647, 0.599333333333333]
                                                                                                                           9 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
10 [0.4989999999999983, -0.5806666666666666, 0.5993333333333333]
11 [0.4989999997660194, -0.5806666666459375, 0.5993333333388429] 11 [0.498999999999, -0.5806666666666658, 0.599333333333333
                                                                                                                           11 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
12 [0.498999999834717, -0.580666666520236, 0.5993333333372253]
                                                             12 [0.4989999999999933, -0.58066666666666661, 0.5993333333333333]
                                                                                                                           12 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
13 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
14 [0.498999999917524, -0.580666666593597, 0.599333333352755]
                                                             14 [0.4989999999999967, -0.5806666666666664, 0.5993333333333335]
                                                                                                                           14 [0.4989999999999983, -0.5806666666666666, 0.5993333333333335]
15 [0.498999999941739, -0.58066666666505, 0.5993333333347053]
                                                             15 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
                                                                                                                           15 [0.4989999999999983, -0.5806666666666664, 0.59933333333333355]
16 [0.498999999958844, -0.5806666666630205, 0.5993333333343025]
                                                             16 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
                                                                                                                           16 [0.4989999999999983, -0.5806666666666666, 0.5993333333333333]
17 [0.4989999999970927, -0.5806666666664091, 0.599333333334018]
                                                             17 [0.4989999999999983, -0.580666666666664, 0.5993333333333333]
                                                                                                                           17 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
18 [0.49899999999794625, -0.58066666666648472, 0.5993333333333817]
                                                             18 [0.4989999999999983, -0.580666666666664, 0.5993333333333333]
                                                                                                                           18 [0.4989999999999983, -0.5806666666666666, 0.5993333333333335]
19 [0.4989999999854927, -0.580666666653815, 0.59933333333675]
                                                             19 [0.4989999999999983, -0.58066666666666664, 0.5993333333333333]
                                                                                                                           19 [0.4989999999999983, -0.58066666666666666, 0.5993333333333333]
20 [0.4989999999897526, -0.5806666666657587, 0.599333333335747] 20 [0.49899999999999983, -0.58066666666666664, 0.59933333333333333
                                                                                                                           20 [0.4989999999999983, -0.5806666666666666, 0.5993333333333335]
21 [0.49899999999761, -0.5806666666666253, 0.5993333333335038]
                                                             21 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
                                                                                                                           21 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
22 [0.498999999994886, -0.5806666666662137, 0.59933333333334537]
                                                             22 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
                                                                                                                           22 [0.4989999999999983, -0.5806666666666664, 0.59933333333333355]
23 [0.498999999996388, -0.58066666666663467, 0.5993333333334184]
                                                             23 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
                                                                                                                           23 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
24 [0.4989999999974487, -0.58066666666664406, 0.5993333333333333333]
                                                             24 [0.4989999999999983, -0.5806666666666666, 0.59933333333333355]
                                                                                                                           24 [0.4989999999999983, -0.58066666666666666, 0.5993333333333333]
```







d) Evalue la matriz de transición del métdo **SOR** y determine varias soluciones aproximadas, para 10 valores de ω . Utilice una tolerancia de 10^{-16}

```
W = 1.375
0 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
1 [0.4989999999999983, -0.58066666666666664, 0.599333333333333333
2 [0.4989999999999983, -0.58066666666666664, 0.599333333333333333
3 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
4 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333
5 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
6 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
7 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
8 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
9 [0.4989999999999983, -0.5806666666666664, 0.599333333333333]
10 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333
11 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333
12 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
13 [0.49899999999999983, -0.5806666666666664, 0.5993333333333333]
14 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333
15 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333
16 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
17 [0.49899999999999983, -0.5806666666666664, 0.5993333333333335]
18 [0.49899999999999983, -0.5806666666666664, 0.5993333333333335]
19 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
20 [0.4989999999999983, -0.5806666666666664, 0.5993333333333333]
22 [0.49899999999999983, -0.58066666666666664, 0.5993333333333333]
23 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
24 [0.4989999999999983, -0.5806666666666664, 0.5993333333333335]
```





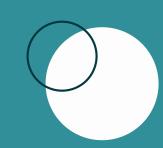


e) Construya una función $f(\omega)$ que determine el valor óptimo de ω para que el método ${\bf SOR}$ converja



No se pudo desarrollar





PUNTO 5

5. Sea I una imagen en blanco y negro, digamos con valores en una gama de 0 a 1 de 800×600 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, segun la caja

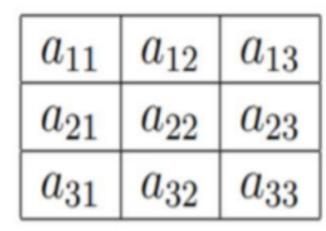
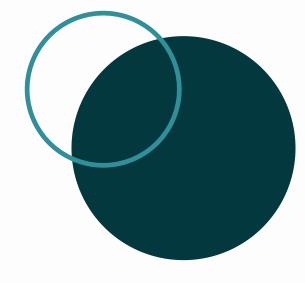


Figura 2: I

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

- a) 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?
- b) ¿Qué condiciones se han de dar para que la matriz de la transformación sea simétrica? ¿Y definida positiva?





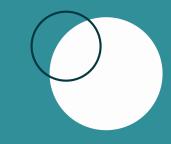
Main

```
\overset{\bullet}{t_0} main.py 	imes \overset{\bullet}{t_0} desenfoque.py 	imes
                                 🛵 gauss.py
      ⊝import numpy as np
        import desenfoque as des
        import jacobi as jac
        limport gauss as gau
        A = np.random.rand(800, 600)
        x = des.igual(A)
        d = des.desenfoque(A)
        x0 = np.zeros_like(x)
        j = jac.jacobi(d, x, x0, n=10)
        g = gau.gauss(d)
```

66

Desenfoque 99

```
🔀 desenfoque.py
                    ち gauss.py 🗡 🛮 ち jacobi
def igual(A):
    x = []
    for i in range(len(A)):
        for j in range(len(A[i])):
            x.append(A[i][j])
    return x
def desenfoque(A):
    d = []
    for i in range(len(A)):
        for j in range(len(A[i])):
            B = []
            if i-1 < 0:
               B.append(0.)
            else:
                B.append(A[i-1, j])
            if i+1 > len(A[i]) - 1:
                B.append(0.)
            else:
                B.append(A[i+1, j])
               B.append(0.)
            else:
                B.append(A[i, j-1])
            if j+1 > len(A[i]) - 1:
                B.append(0.)
            else:
                B.append(A[i, j+1])
            d.append(B)
    return d
```





Gauss

```
import numpy as np
def gauss(A):
    D = np.diag(np.diag(A))
    try:
        LU = A - D
    except:
        print("● La matriz no es simetrica")
    else:
        BJ = np.dot(np.linalg.inv(D), -LU)
        print(BJ)
```

Jacobi 99

```
import numpy as np
def jacobi(A, b, x0, eps=1e-16, n=500):
    D = np.diag(np.diag(A))
    try:
       LU = A - D
    except:
       print("● La matriz no es simetrica")
   else:
       x = x0
        for i in range(0, n):
            D_inv = np.linalg.inv(D)
            xTemp = x
            x = np.dot(D_inv, np.dot(-(LU), x) + b)
            print(f"Pasos: {i} - x: {x}")
            if np.linalg.norm(x - xTemp) < eps:</pre>
                return x
```





output

- C:\Users\juanf\PycharmProjects\Meto
- La matriz no es simetrica
- La matriz no es simetrica

Process finished with exit code 0









10. **fecha de entrega 21 septiembre**: Dado un sistema de ecuaciones no lineales, implemente el método de Newton Multivariado (es decir para varias variables) para resolver el problema:

Determinar numericamente la interseccion entre la circunferencia $x^2 + y^2 = 1$ y la recta y = x. Usamos una aproximacion inicial (1,1).

No se pudo desarrollar





Bibliografia

1. HTTPS://GITHUB.COM/SALVADOR04/SCIENTIFIC_COMPUTATION/BLOB/3BACB6A56D147BDBC6A1A5E6DE6811F1F

ADE8DF6/6-GAUSSSEI.PY



Muchas gracias por su atención