Adjunto los códigos de Python por separado pues en el archivo generado por OverLeaf no pude hacer que aparecieran completos :c

Ejercicio 1

import numpy as np

from math import exp

import matplotlib.pyplot as plt

from matplotlib import cm

x = np.arange(0, 100, 0.1)

y = np.arange(0, 100, 0.1)

z = np.arange(0, 100, 0.1)

X, Y = np.meshgrid(x, y)

A1 = np.zeros(len(x))

A2 = np.zeros(len(x))

B1 = np.zeros(len(y))

B2 = np.zeros(len(y))

C1 = np.zeros(len(z))

C2 = np.zeros(len(z))

CZ = np.zeros((len(x),len(y)))

def difusificacion(): #Obtención de los grados de pertenencia a los conjuntos de los universos de entrada

  valx = int(input("Ingrese el valor de x: "))

  memA = [A1[valx\*10],A2[valx\*10]]

  valy = int(input("Ingrese el valor de y: "))

  memB = [B1[valy\*10],B2[valy\*10]]

  composicion(memA,memB)

def composicion(memA,memB): #Composición max - min (Evaluación de las reglas "si-entonces")

  tabla = [[np.minimum(memA[0],memB[0]),np.minimum(memA[1],memB[0])],

           [np.minimum(memA[0],memB[1]),np.minimum(memA[1],memB[1])]] #Tabla de inferencia (intersección - min)

  print("Tabla de inferencia: ",str(tabla))

  agregacionC1 = max(tabla[0][0],tabla[1][0]) #Agregación - max

  agregacionC2 = max(tabla[0][1],tabla[1][1])

  agregacion(agregacionC1,agregacionC2)

def agregacion(agregacionC1,agregacionC2):

  aux1 = agregacionC1\*np.ones(len(C1))

  aux2 = agregacionC2\*np.ones(len(C2))

  C1\_r = np.minimum(C1,aux1)

  C2\_r = np.minimum(C2,aux2)

  plt.figure(4)

  plt.plot(z,C1,z,C2,z,C1\_r,z,C2\_r)

  plt.xlabel("Universo Z")

  plt.ylabel("Membresía")

  dedifusificacion(C1\_r,C2\_r)

def dedifusificacion(C1\_r,C2\_r):

  z\_d = 0

  aux1 = 0

  aux2 = 0

  CT = np.maximum(C1\_r,C2\_r)

  plt.figure(5)

  plt.plot(z,CT)

  plt.xlabel("Universo Z")

  plt.ylabel("Membresía")

  for i in range(len(z)):

    aux1 += CT[i]\*i/10

    aux2 += CT[i]

  z\_d = aux1/aux2

  print("z\* = ", str(z\_d))

#Generación de los conjuntos difusos

for i in range(len(x)):

  A1[i] = 1 / ( 1 + exp(0.3\*(x[i]-50)))

  A2[i] = 1 / ( 1 + exp(-0.3\*(x[i]-50)))

  B1[i] = exp(-1/2\*((y[i]-25)/20)\*\*2)

  B2[i] = exp(-1/2\*((y[i]-75)/20)\*\*2)

  C1[i] = 1 / ( 1 + exp(0.3\*(z[i]-50)))

  C2[i] = exp(-1/2\*((z[i]-75)/20)\*\*2)

plt.figure(1)

plt.plot(x,A1,x,A2)

plt.xlabel("Universo X")

plt.ylabel("Membresía")

plt.figure(2)

plt.plot(y,B1,y,B2)

plt.xlabel("Universo Y")

plt.ylabel("Membresía")

plt.figure(3)

plt.plot(z,C1,z,C2)

plt.xlabel("Universo Z")

plt.ylabel("Membresía")

difusificacion()

Ejercicio 2

def difusificacion(valx,valy):

  memA = [A1[valx],A2[valx]]

  memB = [B1[valy],B2[valy]]

  return composicion(memA,memB)

def composicion(memA,memB):

  tabla = [[np.minimum(memA[0],memB[0]),np.minimum(memA[1],memB[0])],

           [np.minimum(memA[0],memB[1]),np.minimum(memA[1],memB[1])]]

  agregacionC1 = max(tabla[0][0],tabla[1][0])

  agregacionC2 = max(tabla[0][1],tabla[1][1])

  return agregacion(agregacionC1,agregacionC2)

def agregacion(agregacionC1,agregacionC2):

  aux1 = agregacionC1\*np.ones(len(C1))

  aux2 = agregacionC2\*np.ones(len(C2))

  C1\_r = np.minimum(C1,aux1)

  C2\_r = np.minimum(C2,aux2)

  return dedifusificacion(C1\_r,C2\_r)

def dedifusificacion(C1\_r,C2\_r):

  z\_d = 0

  aux1 = 0

  aux2 = 0

  CT = np.maximum(C1\_r,C2\_r)

  for i in range(len(z)):

    aux1 += CT[i]\*i/10

    aux2 += CT[i]

  z\_d = aux1/aux2

  return z\_d

def malla():

  for i in range(len(x)):

    for j in range(len(y)):

      CM[i,j] = difusificacion(i,j)

  fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})

  ax.set\_xlabel('Y')

  ax.set\_ylabel('X')

  ax.set\_zlabel('Z')

  surf = ax.plot\_surface(X, Y, CM, cmap=cm.coolwarm,

                       linewidth=0, antialiased=False)

malla()