Aula Prática – Modelos Fuzzy TSK

Exercício 1:

Utilizando o Método do gradiente mostrados nos slides e vídeos do tópico 3, aproximar a saída para $f(x) = x^2$

Professor:

Rogério Martins Gomes

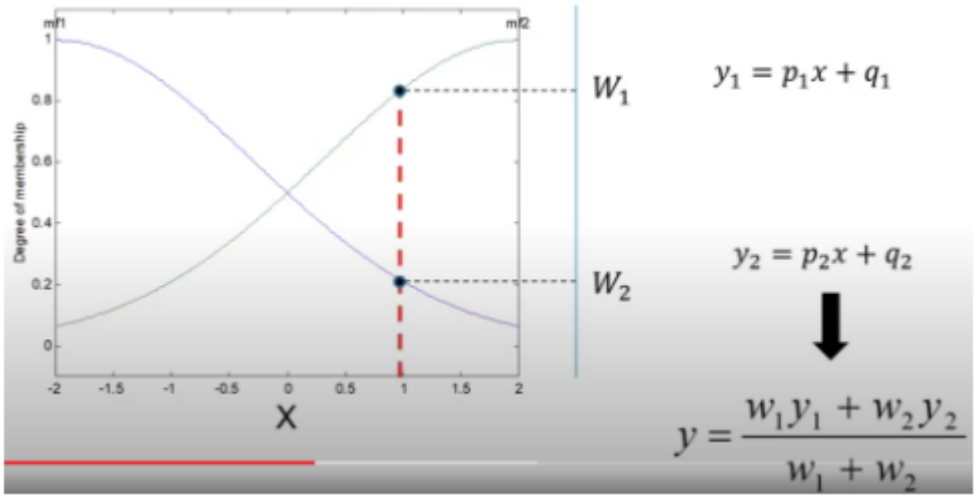
Alunos:

Aritana Noara Costa Santos e Victor Augusto Januário da Cruz

A solução exige uso de gaussianas.

$$w_1 = \exp\left[-\frac{1}{2}\left(\frac{x-\bar{x}_1}{\sigma_1}\right)^2\right]$$

$$w_2 = \exp\left[-\frac{1}{2}\left(\frac{x-\bar{x}_2}{\sigma_2}\right)^2\right]$$



Derivada do erro em relação aos parâmetros

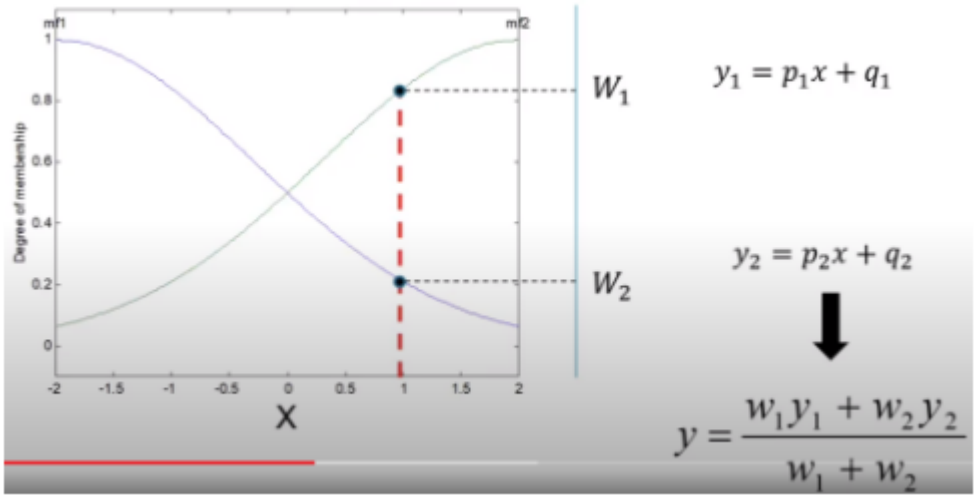
In [1]:
```
def Dp1(y, yd, w1, w2,x):
    return (y-yd) * x * w1 / (w1 + w2)


def Dp2(y, yd, w1, w2,x):
    return (y-yd) * x * w2 / (w1 + w2)


def Dpq1(y, yd, w1, w2,x):
    return (y-yd) * w1 / (w1 + w2)

def Dpq2(y, yd, w1, w2,x):
    return (y-yd) * w2 / (w1 + w2)

def Dpx1_bar(y, yd, w1, w2, x, y1, y2, x1_bar, sd1):
    return (y-yd) * w2 * ((y1 - y2)/((w1 + w2)**2)) * w1 *((x - x1_bar)/sd1**2

def Dpx2_bar(y, yd, w1, w2, x, y1, y2, x2_bar, sd2):
    return (y-yd) * w1 * ((y2 - y1)/(w1 + w2)**2) * w2 *((x - x2_bar)/sd2**2)

def Dpsd1(y, yd, w1, w2, x, y1, y2, x1_bar, sd1):
    return (y-yd) * w2 * ((y1 - y2)/(w1 + w2)**2) * w1 *((x - x1_bar)**2/sd1**

def Dpsd2(y, yd, w1, w2, x, y1, y2, x2_bar, sd2):
    return (y-yd) * w1 * ((y2 - y1)/(w1 + w2)**2) * w2 *((x - x2_bar)**2/sd2**
```

Testes das funções que calculam a derivada

In [2]:
```
dp1 = Dp1(y=1,yd=2,w1=3,w2=4,x=5)

if(round(dp1,11)!=-2.14285714286):
    print("Erro em dp1")
    print(round(dp1,11))


dp2 = Dp2(y=1,yd=2,w1=3,w2=4,x=5)

if(round(dp2,11)!=-2.85714285714):
```

```
    print("Erro em dp2")
    print(round(dp2,11))

dpq1 = Dpq1(y=1,yd=2,w1=3,w2=4,x=5)

if(round(dpq1,11)!=-0.42857142857):
    print("Erro em Dpq1")
    print(round(dpq1,11))


dpq2 = Dpq2(y=1,yd=2,w1=3,w2=4,x=5)

if(round(dpq2,11)!=-0.57142857143):
    print("Erro em Dpq2")
    print(round(dpq2,11))


dpx1_bar = Dpx1_bar(y=1,yd=2,w1=3,w2=4,x=5,y1=6, y2=7, x1_bar=8, sd1=9)

if(round(dpx1_bar,11)!=-0.00907029478):
    print("Erro em dpx1_bar")
    print(round(dpx1_bar,11))


dpx1_bar = Dpx1_bar(y=1,yd=2,w1=3,w2=4,x=5,y1=6, y2=7, x1_bar=8, sd1=9)

if(round(dpx1_bar,11)!=-0.00907029478):
    print("Erro em dpx1_bar")
    print(round(dpx1_bar,11))

dpx2_bar = Dpx2_bar(y=1,yd=2,w1=3,w2=4,x=5,y1=6, y2=7, x2_bar=8, sd2=9)

if(round(dpx2_bar,11)!=0.00907029478):
    print("Erro em dpx2_bar")
    print(round(dpx2_bar,11))


dpsd1 = Dpsd1(y=1,yd=2,w1=3,w2=4,x=5,y1=6, y2=7, x1_bar=8, sd1=9)

if(round(dpsd1,11)!=0.00302343159):
    print("Erro em Dpsd1")
    print(round(dpsd1,11))

dpsd2 = Dpsd2(y=1,yd=2,w1=3,w2=4,x=5,y1=6, y2=7, x2_bar=8, sd2=9)

if(round(dpsd2,11)!=-0.00302343159):
    print("Erro em Dpsd2")
    print(round(dpsd2,11))
```

Estimar Curva



In [3]:
```
#Equações das curvas
import math

def DoGaussian(x, mean, sd ):
  w = (x - mean)/ sd
  w =  w**2
  w =  -0.5*w
  w =  math.exp(w)
  return w

def DoY1(p1,x, q1):
    return p1 *x + q1

def DoY2(p2,x, q2):
    return p2 *x + q2

def DoY(w1,y1,w2,y2):
    return (w1*y1 + w2*y2)/(w1 + w2)

def Initial_sd(mean,numberOfPoints,points):
    serie = 0
    for i in range(numberOfPoints):
        serie = serie + (points[i] - mean)**2
    sd = math.sqrt(serie/numberOfPoints)
    return sd
```

Testes das equações das curvas

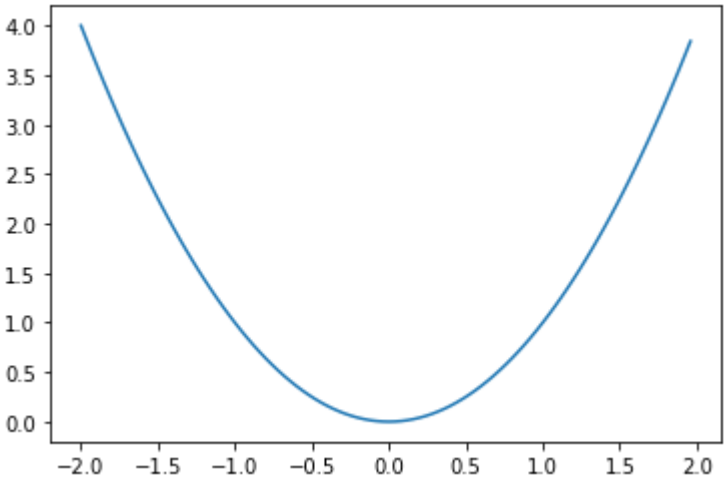
In [4]:
```
doGaussian = DoGaussian(x=92, mean=2, sd=2.311133055451371)
if(round(doGaussian,11)!=0.94595946891):
    print("Erro em DoGaussian")
    print(round(doGaussian,11))



doY1 = DoY1(p1=1,x=2, q1=3)
if(round(doY1,11)!=5):
    print("Erro em DoY1")
    print(round(doY1,11))
```

```
doY2 = DoY2(p2=1,x=2, q2=3)
if(round(doY2,11)!=5):
    print("Erro em DoY2")
    print(round(doY2,11))


doY = DoY(w1=1,y1=2,w2=3,y2=4)
if(round(doY,11)!=3.5):
    print("Erro em DoY")
    print(round(doY,11))

initial_sd = Initial_sd(mean=1.68, numberOfPoints=3,points=[1.55,1.70,1.80])
if(round(initial_sd,11)!=0.10279429297):
    print("Erro em Initial_sd")
    print(round(initial_sd,11))
```

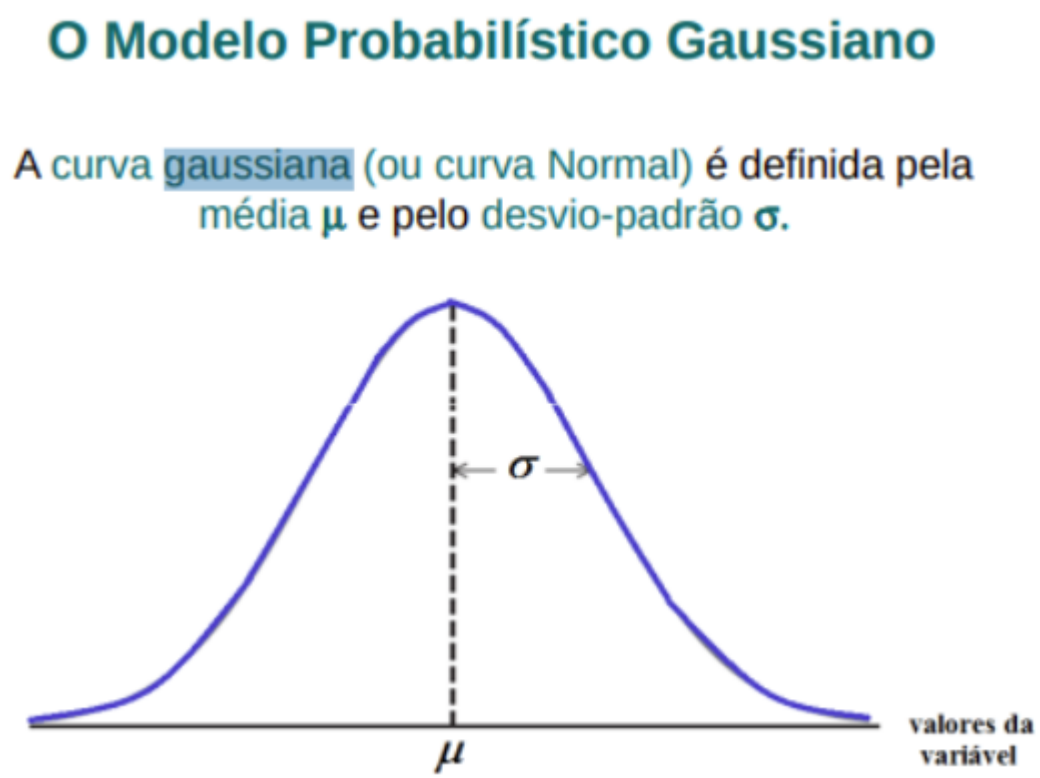
```
Erro em DoGaussian
0.0
```

In [5]:

```
##Criar array com 1000 posições Para Plotar Função Quadrática

import numpy as np
import matplotlib.pyplot as plt

x = -2
y = x**2
arrayX = [x]
arrayY = [y]
RANGE = 100
STEPS = 0.04

#Array com 1000 pontos entre -2 e 2
for i in range(1,RANGE):
    x = x + STEPS
    arrayX.append(x)


#calculando a funcao para cada ponto

for i in range(1,RANGE):
    y = (arrayX[i])**2
    arrayY.append(y)



xpoints = np.array(arrayX)
ypoints = np.array(arrayY)


plt.plot(xpoints, ypoints)
plt.show()
```

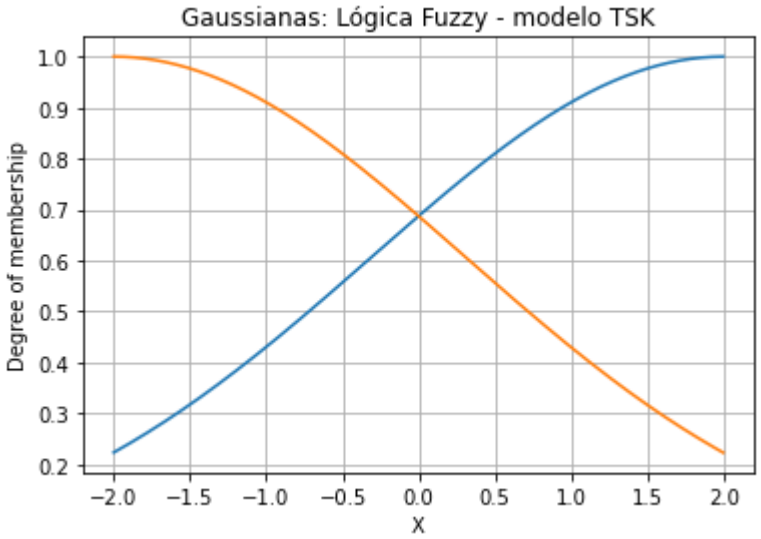


Plotar Gaussianas 1 e 2



In [6]:
```
import math
import numpy as np
import matplotlib.pyplot as plt

RANGE = 4
FROM=-2
TO = 2
NUMBER_OF_POINTS = 1000
STEPS = RANGE/NUMBER_OF_POINTS

# Plot between -2 and 2 with .004 steps.
x_axis = np.arange(FROM, TO, STEPS)
y_axis_1 = []
y_axis_2 = []
```

```
# Calculating mean and standard deviation
mean_1 = 2
mean_2 = -2

sd_1 = Initial_sd(mean_1, NUMBER_OF_POINTS,x_axis)
sd_2 = Initial_sd(mean_2, NUMBER_OF_POINTS,x_axis)

for i in range(len(x_axis)):
    y = DoGaussian(x_axis[i], mean_1, sd_1)
    y_axis_1.append(y)

for i in range(len(x_axis)):
    y = DoGaussian(x_axis[i], mean_2, sd_2)
    y_axis_2.append(y)

plt.plot(x_axis,y_axis_1)
plt.plot(x_axis,y_axis_2)

plt.title("Gaussianas: Lógica Fuzzy - modelo TSK")
plt.xlabel("X")
plt.ylabel("Degree of membership")

plt.grid()
plt.show()
```



Algoritmo

0) Começar com os parâmetros de forma aleatória

1) Encontrar valor de y e o valor real yd (parábola)

$$y_1 = p_1 x + q_1$$

$$y_2 = p_2 x + q_2$$

$$\downarrow$$

$$y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2}$$

2) Encontrar o valor do erro.

$$e = \frac{1}{2}(y - yd)^2 \Rightarrow f(\bar{x}_1, \sigma_1, \bar{x}_2, \sigma_2, p_1, p_2, q_1, q_2)$$

3) Definir um valor alfa para corrigir cada parâmetro em relação ao passo do gradiente.

$$p_1(k+1) = p_1(k) - \alpha * \frac{\partial e}{\partial p_1} \qquad p_2(k+1) = p_2(k) - \alpha * \frac{\partial e}{\partial p_2}$$

$$q_1(k+1) = q_1(k) - \alpha * \frac{\partial e}{\partial q_1} \qquad q_2(k+1) = q_2(k) - \alpha * \frac{\partial e}{\partial q_2}$$

$$\bar{x}_1(k+1) = \bar{x}_1(k) - \alpha * \frac{\partial e}{\partial \bar{x}_1} \qquad \bar{x}_2(k+1) = \bar{x}_2(k) - \alpha * \frac{\partial e}{\partial \bar{x}_2}$$

$$\sigma_1(k+1) = \sigma_1(k) - \alpha * \frac{\partial e}{\partial \sigma_1} \qquad \sigma_2(k+1) = \sigma_2(k) - \alpha * \frac{\partial e}{\partial \sigma_2}$$

4) Corrigir cada parâmetro

5) O ajuste deve ser aleatório incrementalmente para os o vetor contendo mil pontos, de forma que todos os mil pontos sejam corrigidos, mas de forma incremental e aletória, para permitir um melhor ajuste da curva.

6) Imprimir valor do erro.

7) Plotar y

In [10]:
```
#0) Definir eixo x, valores iniciais dos parâmetros

import math
import numpy as np
```

```
import matplotlib.pyplot as plt

p1 = 0.5
p2 = 0.6

q1 = 0.3
q2 = 0.4

mean_1 = 2 #x1 barrado
mean_2 = -2 #x2 barrado

sd_1 = Initial_sd(mean_1, NUMBER_OF_POINTS,x_axis)
sd_2 = Initial_sd(mean_2, NUMBER_OF_POINTS,x_axis)

#definir vetor da abscissa dos pontos que foram atualizados
PontosAtualizados=[]
#vetorParametros
P1x =[]
P2x =[]
Q1x =[]
Q2x =[]
Mean_1 =[]
Mean_2 =[]
Sd_1 =[]
Sd_2 =[]
Y=[]


#definir eixo x, vetor com 1000 pontos no intervalo (-2, 2), com passo de 0.00
RANGE = 4
FROM=-2
TO = 2
NUMBER_OF_POINTS = 1000
STEPS = RANGE/NUMBER_OF_POINTS

x_axis = np.arange(FROM, TO, STEPS)
```

In [72]:

```
##Funcao para atualizar parametros
import random



#testado
def InicializarVetorParametros(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2):

    PontosAtualizados.clear()
    P1x.clear()
    P2x.clear()
    Q1x.clear()
    Q2x.clear()
    Mean_1.clear()
    Mean_2.clear()
    Sd_1.clear()
    Sd_2.clear()
    Y.clear()

    for i in range(len(x_axis)):
        P1x.append(p1)
        P2x.append(p2)
```

```
        Q1x.append(q1)
        Q2x.append(q2)
        Mean_1.append(mean_1)
        Mean_2.append(mean_2)
        Sd_1.append(sd_1)
        Sd_2.append(sd_2)
        PontosAtualizados.append(0) #0: nao utilizado, 1:utilizado
        Y.append(1 )#inicializar Y com numeros 1's que serao substituidos por


def AtualizarVetorDePontosAtulizados():
    PontosAtualizados.clear()
    for i in range(len(x_axis)):
        PontosAtualizados.append(0)


#testado
def RecuperarParametrosDoPonto(x):
    p1 = P1x[x]
    p2 = P2x[x]
    q1 = Q1x[x]
    q2 = Q2x[x]
    mean_1 = Mean_1[x]
    mean_2 = Mean_2[x]
    sd_1 = Sd_1[x]
    sd_2 = Sd_2[x]
    return(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2)


#testado
def AtualizarParametro(parametro,alfa, derivada):
    return parametro - alfa * derivada

#testado
def PegarNumeroAleatorio(minimo, maximo):
    randNumber = random.randint(minimo, maximo)#aleatorio entre 0 e maximo-1
    return randNumber

#testado
def VerificaSePontoJaFoiSelecinado(x):
    if(PontosAtualizados[x]==1):
        return 1
    else:
        return 0

#testado
def VerificaSeHaPontosParaSelecionar():
    posicoesVazias=0
    for i in range(len(PontosAtualizados)):
        if(PontosAtualizados[i]==0):
            posicoesVazias = posicoesVazias+1
    return posicoesVazias


#testado, retorna -1 ou um numero que indica a posicao vazia escolhida aleator
def PegarUmPontoNaoSelecionadoNoVetorPontosAtualizados(intervaloMinimo,interva
    quantidadePosicoesVazias = VerificaSeHaPontosParaSelecionar()
    if(quantidadePosicoesVazias != 0):
        randNumber = PegarNumeroAleatorio(intervaloMinimo,intervaloMaximo)
        selecionado = VerificaSePontoJaFoiSelecinado(randNumber)
```

```
        while(selecionado == 1):
            randNumber = PegarNumeroAleatorio(intervaloMinimo,intervaloMaximo)
            selecionado = VerificaSePontoJaFoiSelecinado(randNumber)
            #print("tentando")
        return randNumber
    else:
        return -1
    #print("----------------------TERMINADO ---------------------------------


#Testado
def PlotaGrafico(x_axis,Y,erro):
    plt.xlabel("x")
    plt.ylabel("y")
    plt.plot(x_axis,Y)
    plt.plot(x_axis,x_axis**2)
    plt.title("Y: Lógica Fuzzy - modelo TSK - Erro:%f" %erro)
    plt.grid()
    plt.show()

def RecalcularY():
    Y.clear()
    for i in range(len(x_axis)):
        x = x_axis [i]
        p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2 = RecuperarParametrosDoPonto(i)
        y1 = DoY1(p1,x, q1)
        #print("y1 %2f" %(y1))

        y2 = DoY2(p2,x, q2)
        #print("y2 %2f" %(y2))

        w1 = DoGaussian(x, mean_1, sd_1)
        #print("w1 %2f" %(w1))

        w2 = DoGaussian(x, mean_2, sd_2)
        #print("w2 %2f" %(w2))

        y = DoY(w1,y1,w2,y2)
        #print("y %2f" %(y))
        Y.append(y)
        #print("SALVOU Y----------------------------------------")
    return Y

def AtualizarVetorParametros(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2,posicao):

    P1x.pop(posicao)
    P1x.insert(posicao,p1)

    P2x.pop(posicao)
    P2x.insert(posicao,p2)

    Q1x.pop(posicao)
    Q1x.insert(posicao,q1)

    Q2x.pop(posicao)
    Q2x.insert(posicao,q2)

    Mean_1.pop(posicao)
    Mean_1.insert(posicao,mean_1)
```

```
    Mean_2.pop(posicao)
    Mean_2.insert(posicao,mean_2)

    Sd_1.pop(posicao)
    Sd_1.insert(posicao,sd_1)

    Sd_2.pop(posicao)
    Sd_2.insert(posicao,sd_2)


#testado
def ImprimirVetores():
    print("Vetores")
    print("--------------------x_axis:-----------------Tamanho %2d" %(len(x_ax
    print(x_axis)
    print("--------------------P1x:-----------------Tamanho %2d" %(len(P1x)))
    print(P1x)
    print("--------------------P2x-----------------Tamanho %2d" %(len(P2x)))
    print(P2x)
    print("--------------------Q1x:-----------------Tamanho %2d" %(len(Q1x)))
    print(Q1x)
    print("--------------------Q2x:-----------------Tamanho %2d" %(len(Q2x)))
    print(Q2x)
    print("--------------------Mean_1:-------------Tamanho %2d" %(len(Mean_1)
    print(Mean_1)
    print("--------------------Mean_2:-------------Tamanho %2d" %(len(Mean_2)
    print(Mean_2)
    print("--------------------Sd_1:---------------Tamanho %2d" %(len(Sd_1)))
    print(Sd_1)
    print("--------------------Sd_2:---------------Tamanho %2d" %(len(Sd_2)))
    print(Sd_2)
    print("--------------------PontosAtualizados:---Tamanho %2d" %(len(PontosA
    print(PontosAtualizados)
    print("--------------------Y:---Tamanho %2d" %(len(Y)))
    print(Y)

 #testado
def ImprimirTamanhoVetores():
    print("Vetores")
    print("--------------------x_axis:-----------------Tamanho %2d" %(len(x_ax
    print("--------------------P1x:-----------------Tamanho %2d" %(len(P1x)))
    print("--------------------P2x-----------------Tamanho %2d" %(len(P2x)))
    print("--------------------Q1x:-----------------Tamanho %2d" %(len(Q1x)))
    print("--------------------Q2x:-----------------Tamanho %2d" %(len(Q2x)))
    print("--------------------Mean_1:-------------Tamanho %2d" %(len(Mean_1)
    print("--------------------Mean_2:-------------Tamanho %2d" %(len(Mean_2)
    print("--------------------Sd_1:---------------Tamanho %2d" %(len(Sd_1)))
    print("--------------------Sd_2:---------------Tamanho %2d" %(len(Sd_2)))
    print("--------------------PontosAtualizados:---Tamanho %2d" %(len(PontosA
    print("--------------------Y:---Tamanho %2d" %(len(Y)))
```

Teste das funções

In [73]:

```
#### para testar, passar de markdown para code
#payload
p1 = 0.5
p2 = 0.6
```

```
q1 = 0.3
q2 = 0.4
mean_1 = 2 #x1 barrado
mean_2 = -2 #x2 barrado
sd_1 = Initial_sd(mean_1, NUMBER_OF_POINTS,x_axis)
sd_2 = Initial_sd(mean_2, NUMBER_OF_POINTS,x_axis)

##T1
InicializarVetorParametros(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2)
#ImprimirVetores()

##T2
P1x.pop(0)
P1x.insert(0,-1)

PontosAtualizados.pop(0)
PontosAtualizados.insert(0,1)

P2x.pop(0)
P2x.insert(0,-1)

Q1x.pop(0)
Q1x.insert(0,-1)

Q2x.pop(0)
Q2x.insert(0,-1)

Mean_1.pop(0)
Mean_1.insert(0,-1)

Mean_2.pop(0)
Mean_2.insert(0,-1)

Sd_1.pop(0)
Sd_1.insert(0,-1)

Sd_2.pop(0)
Sd_2.insert(0,-1)

p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2 = RecuperarParametrosDoPonto(0)

#print(p1)
#print(p2)
#print(q1)
#print(q2)
#print(mean_1)
#print(mean_2)
#print(sd_1)
#print(sd_2)

##T3
parametro = 2
alfa=0.1
derivada=0.5

pk1 = AtualizarParametro(parametro,alfa, derivada)
#print(pk1)


##T4
```

```
randNumber = PegarNumeroAleatorio(minimo=0, maximo=3)
#print(randNumber)

##T5
PontosAtualizados.pop(0)
PontosAtualizados.insert(0,1)

selecionado=VerificaSePontoJaFoiSelecinado(0)
#print(selecionado)
selecionado=VerificaSePontoJaFoiSelecinado(1)
#print(selecionado)

##T6
PontosAtualizados.clear()
for i in range(len(x_axis)):
    PontosAtualizados.append(1) #0: nao utilizado, 1:utilizado

PontosAtualizados.pop(500)
PontosAtualizados.insert(500,0)
PontosAtualizados.pop(502)
PontosAtualizados.insert(502,0)

#quantidadePosicoesVazias = VerificaSeHaPontosParaSelecionar()
#print(quantidadePosicoesVazias)

#print(PontosAtualizados)

ponto = PegarUmPontoNaoSelecionadoNoVetorPontosAtualizados(intervaloMinimo=0,i
#print("Ponto")
#print(ponto)
PontosAtualizados.pop(500)
PontosAtualizados.insert(500,1)

#print("Ponto")
ponto = PegarUmPontoNaoSelecionadoNoVetorPontosAtualizados(intervaloMinimo=0,i
#print(ponto)

PontosAtualizados.pop(502)
PontosAtualizados.insert(502,1)

#print("Ponto")#nao ha mais
ponto = PegarUmPontoNaoSelecionadoNoVetorPontosAtualizados(intervaloMinimo=0,i
#print(ponto)

##T7

#payload
p1 = 0.5
p2 = 0.6
q1 = 0.3
q2 = 0.4
mean_1 = 2 #x1 barrado
mean_2 = -2 #x2 barrado
sd_1 = Initial_sd(mean_1, NUMBER_OF_POINTS,x_axis)
sd_2 = Initial_sd(mean_2, NUMBER_OF_POINTS,x_axis)

InicializarVetorParametros(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2)
```

```
Y = RecalcularY()
#ImprimirVetores()
#ImprimirTamanhoVetores()


##T8


p1 = -5
p2 = -5
q1 = -5
q2 = -5
mean_1 = -5
mean_2 = -5
sd_1 = -5
sd_2 = -5

AtualizarVetorParametros(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2,0)
#ImprimirTamanhoVetores()
#ImprimirVetores()

##T8
PontosAtualizados.pop(0)
PontosAtualizados.insert(0,50)


AtualizarVetorDePontosAtulizados()
#ImprimirTamanhoVetores()
#ImprimirVetores()
```

Codigo Final

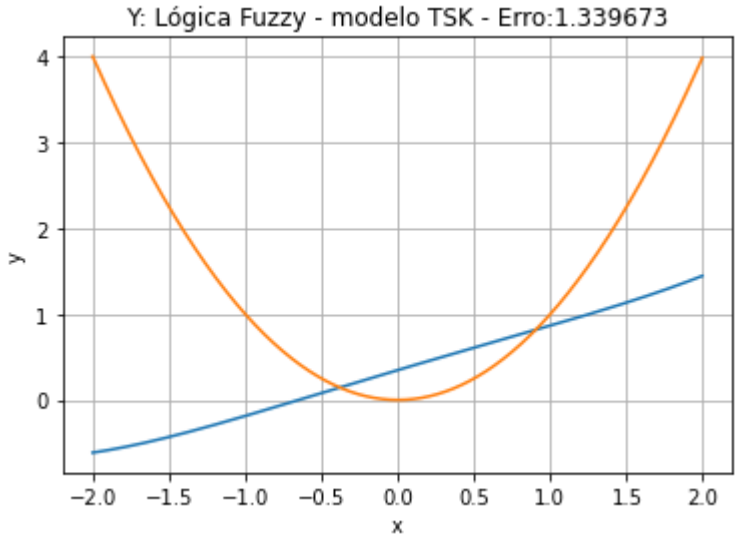
In [81]:

```
import random

#payload
p1 = 0.5
p2 = 0.6
q1 = 0.3
q2 = 0.4
mean_1 = 2 #x1 barrado
mean_2 = -2 #x2 barrado
sd_1 = Initial_sd(mean_1, NUMBER_OF_POINTS,x_axis)
sd_2 = Initial_sd(mean_2, NUMBER_OF_POINTS,x_axis)

InicializarVetorParametros(p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2)
#ImprimirTamanhoVetores()
#ImprimirVetores()

#talves cada epoca vai atualizar mil pontos
#atualizar todos os 1000 pontos, de dez em dez
for epoca in range(369):
    AtualizarVetorDePontosAtulizados()
    for subEpoca in range(100): # 0 a 100
        for grupo in range(10):#0 a 10
            #pegar uma posicao aleatoria que nao foi atualizada na epoca
            posicaoDisponivel = PegarUmPontoNaoSelecionadoNoVetorPontosAtualiz
            if(posicaoDisponivel == -1):
                print("Não há posições para atualizar, verifique se todos os p
```

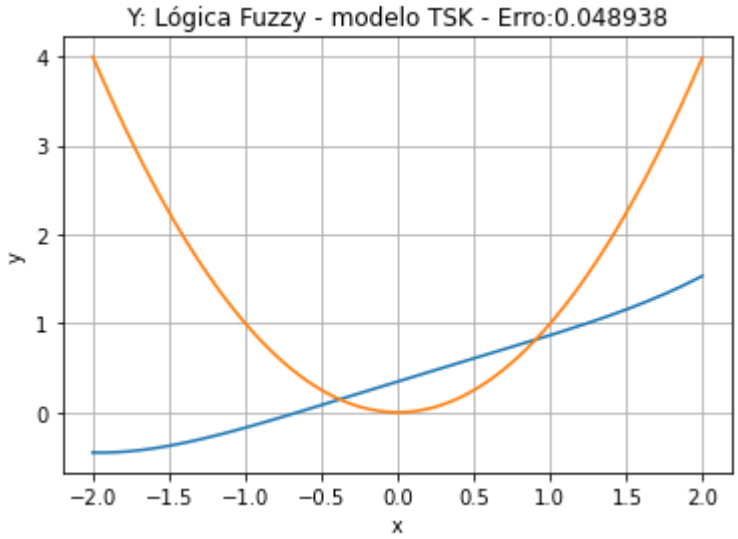
```
            else:
                p1,p2,q1,q2,mean_1,mean_2,sd_1,sd_2 = RecuperarParametrosDoPon

                #correcao de nome de variaveis
                x1_bar = mean_1
                x2_bar = mean_2
                sd1 = sd_1
                sd2 = sd_2

                x = x_axis [posicaoDisponivel]
                #calcular w1,w2,y1,y2, yd e y
                w1 = DoGaussian(x, mean_1, sd_1)
                w2 = DoGaussian(x, mean_2, sd_2)
                y1 = DoY1(p1,x, q1)
                y2 = DoY2(p2,x, q2)
                y = DoY(w1,y1,w2,y2)

                #calcula o erro
                yd = x**2 #valor esperado
                erro = 0.5*(y - yd)**2

                #calcula as derivadas
                dp1 = Dp1(y, yd, w1, w2,x)
                dp2 = Dp2(y, yd, w1, w2,x)
                dpq1 = Dpq1(y, yd, w1, w2,x)
                dpq2 = Dpq2(y, yd, w1, w2,x)
                dmean_1 = Dpx1_bar(y, yd, w1, w2, x, y1, y2, x1_bar, sd1)
                dmean_2 = Dpx2_bar(y, yd, w1, w2, x, y1, y2, x2_bar, sd2)
                dpsd1 = Dpsd1(y, yd, w1, w2, x, y1, y2, x1_bar, sd1)
                dpsd2 = Dpsd2(y, yd, w1, w2, x, y1, y2, x2_bar, sd2)

                #atualiza os parametros em relacao ao passo do gradiente pi-al

                p1_atualizado =  AtualizarParametro(parametro = p1, alfa = 0.0
                p2_atualizado =  AtualizarParametro(parametro = p2, alfa = 0.0
                q1_atualizado =   AtualizarParametro(parametro = q1, alfa = 0.
                q2_atualizado =   AtualizarParametro(parametro = q2, alfa = 0.
                mean_1_atualizado =   AtualizarParametro(parametro = mean_1 ,
                mean_2_atualizado =  AtualizarParametro(parametro = mean_2,alf
                sd_1_atualizado =   AtualizarParametro(parametro = sd_1 , alfa
                sd_2_atualizado =   AtualizarParametro(parametro = sd_2, alfa

                #atualizar posicao no vetor de parametros
                AtualizarVetorParametros(p1_atualizado,p2_atualizado,q1_atuali

                #recalcula Y em relacao aos novos parametros
                w1 = DoGaussian(x, mean_1_atualizado, sd_1_atualizado)
                w2 = DoGaussian(x, mean_2_atualizado, sd_2_atualizado)
                y1 = DoY1(p1_atualizado,x, q1_atualizado)
                y2 = DoY2(p2_atualizado,x, q2_atualizado)
                y = DoY(w1,y1,w2,y2)


                #atualizar Y[]
                Y.pop(posicaoDisponivel)
                Y.insert(posicaoDisponivel,y)

                #MarcarPosicao como não disponivel no vetor PontosAtualizados
                PontosAtualizados.pop(posicaoDisponivel)
                PontosAtualizados.insert(posicaoDisponivel,1)
```

```
    #ao atualizar os dez pontos, plotar Y
    print("epoca:", epoca)
    PlotaGrafico(x_axis,Y,erro)
```

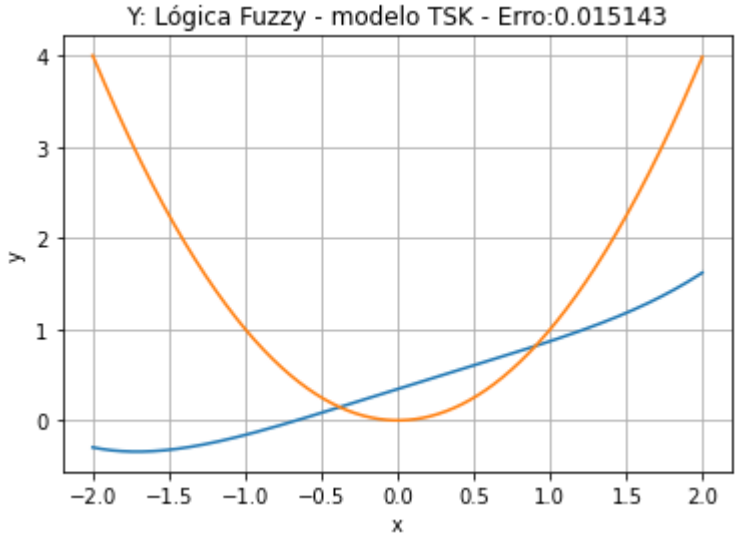
epoca: 0



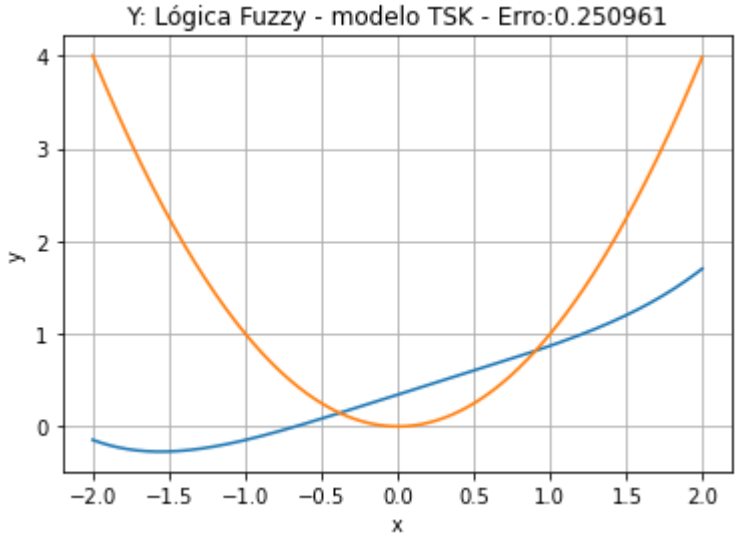
epoca: 1



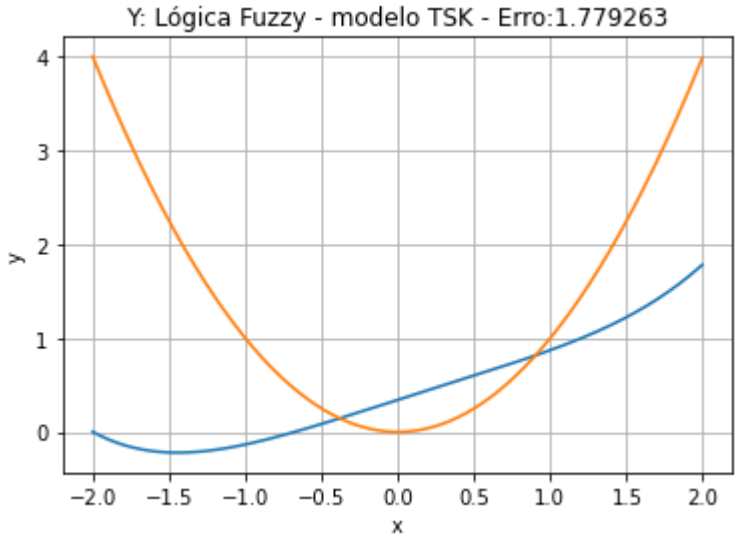
epoca: 2



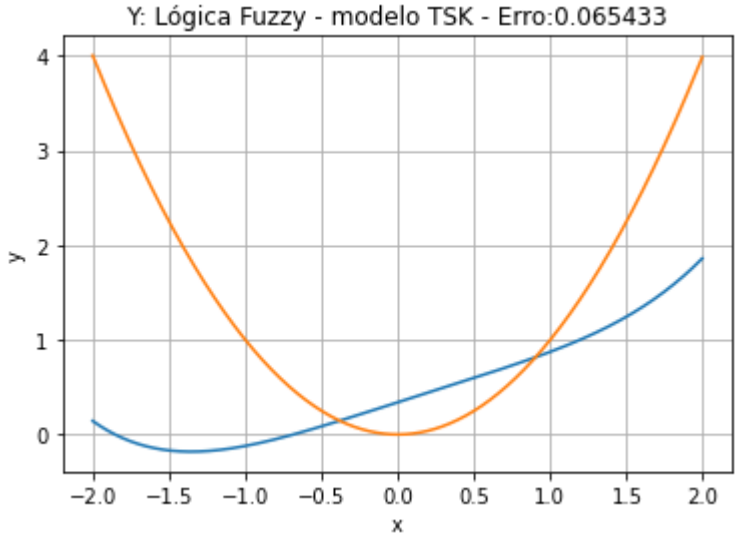
epoca: 3



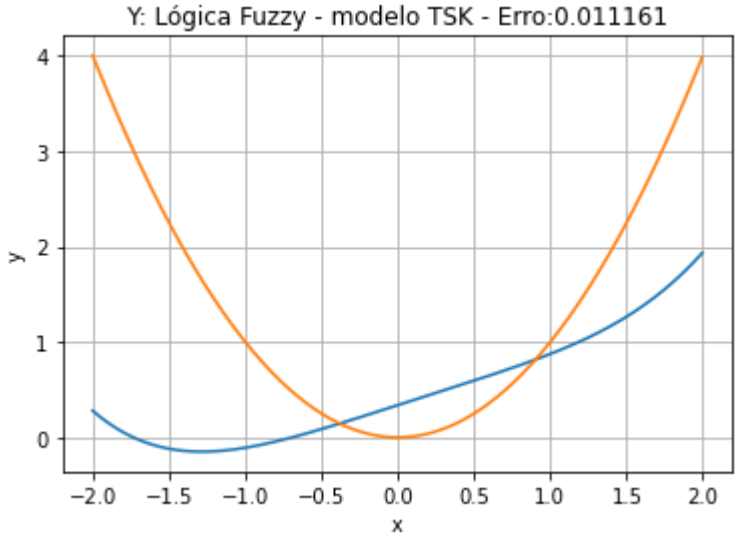
epoca: 4



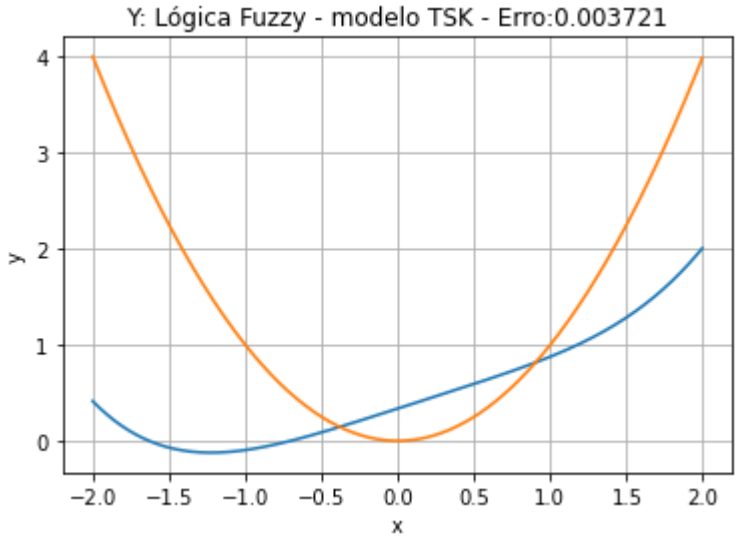
epoca: 5



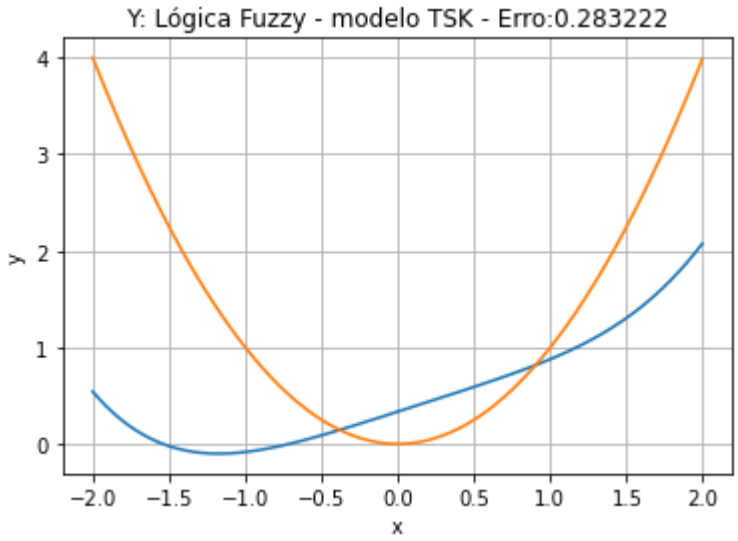
epoca: 6



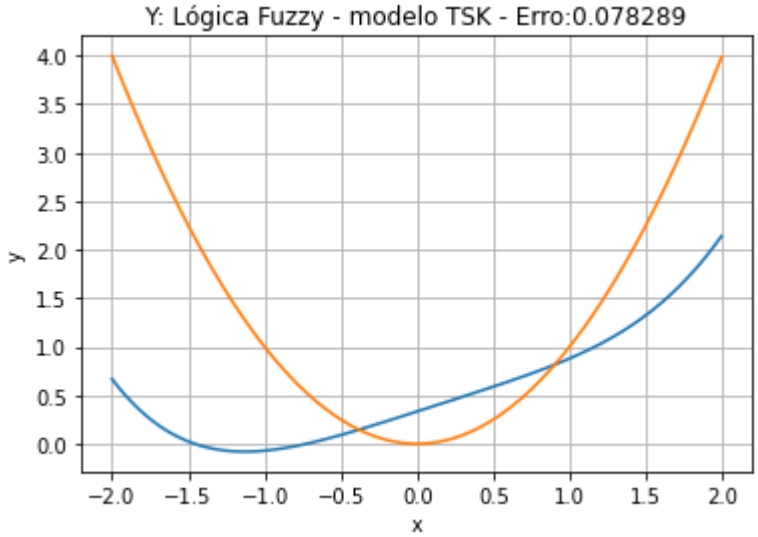
epoca: 7



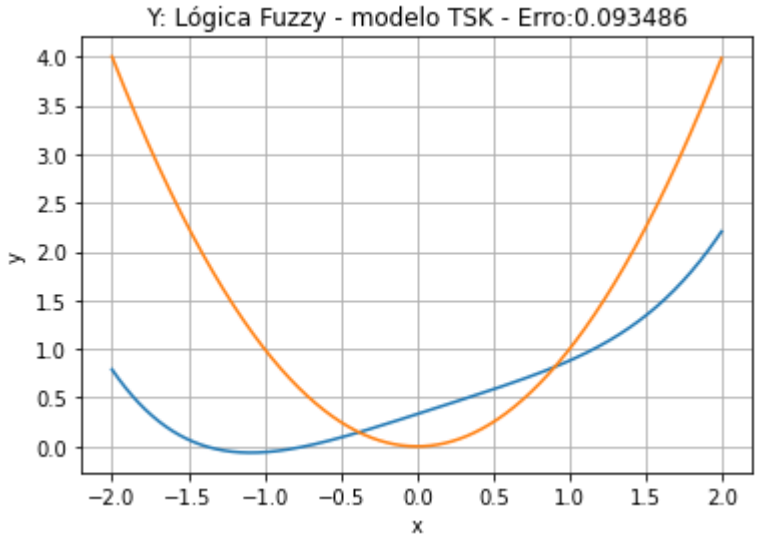
epoca: 8



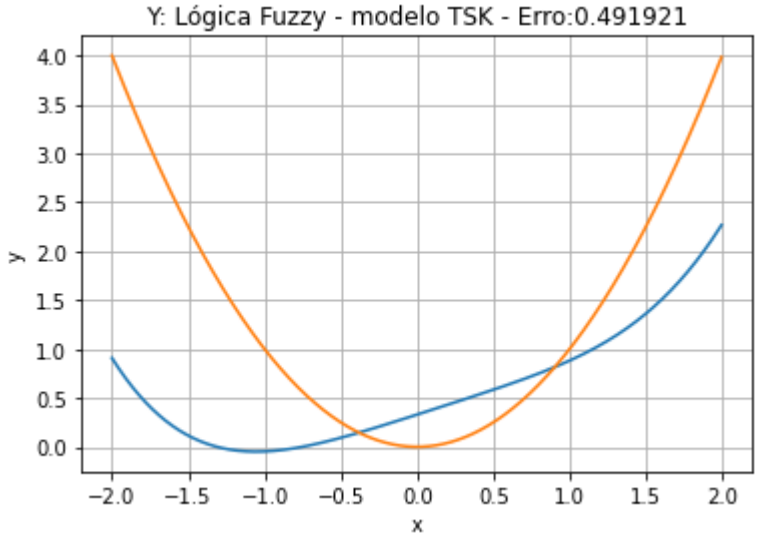
epoca: 9



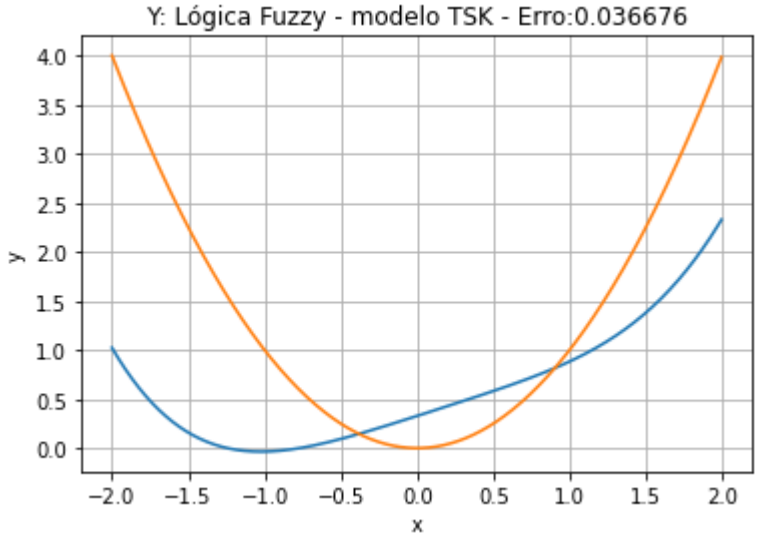
epoca: 10



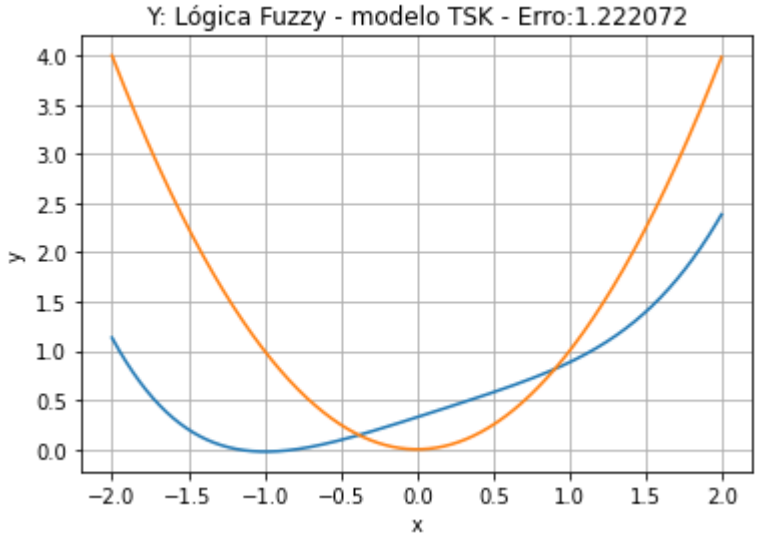
epoca: 11



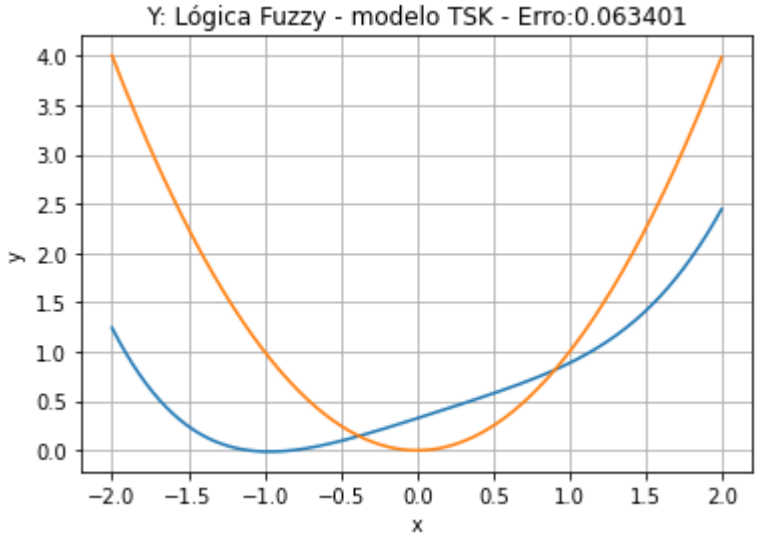
epoca: 12



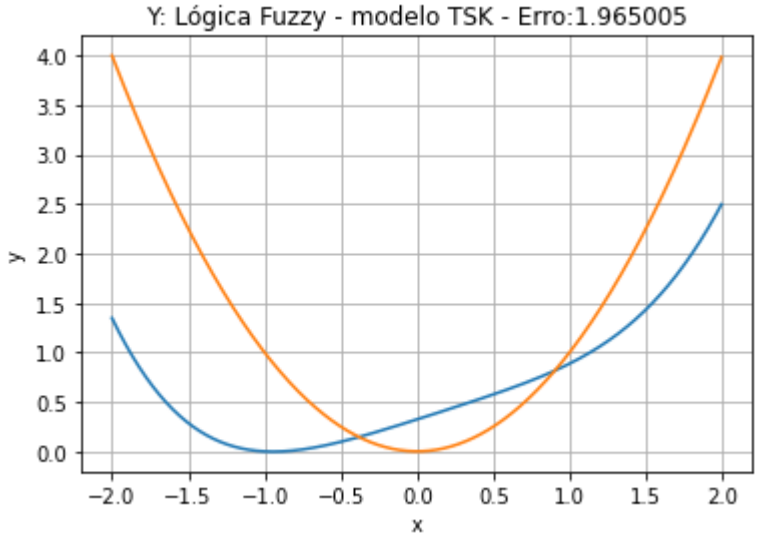
epoca: 13



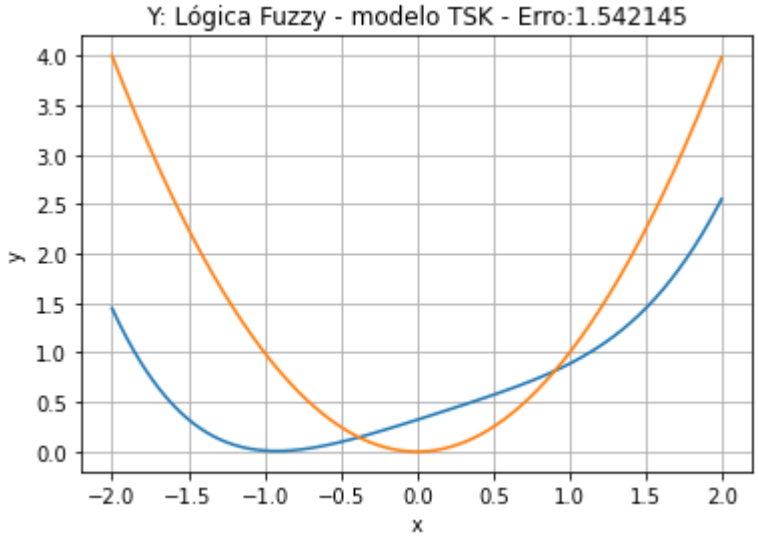
epoca: 14



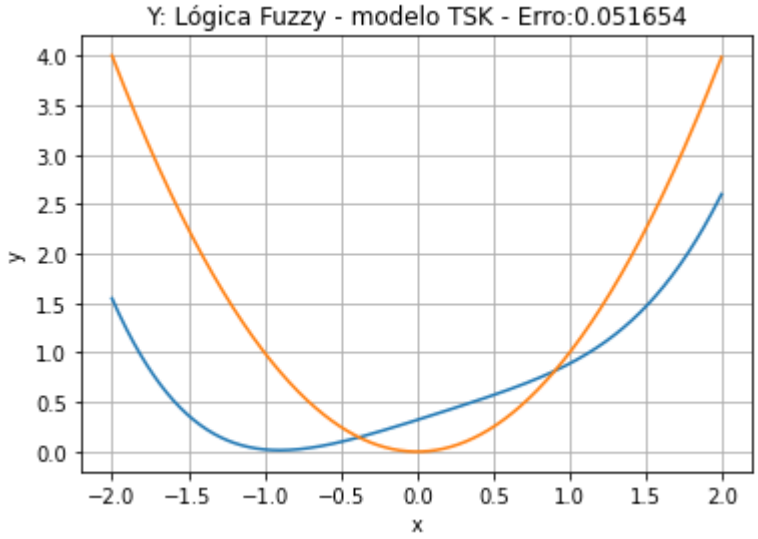
epoca: 15



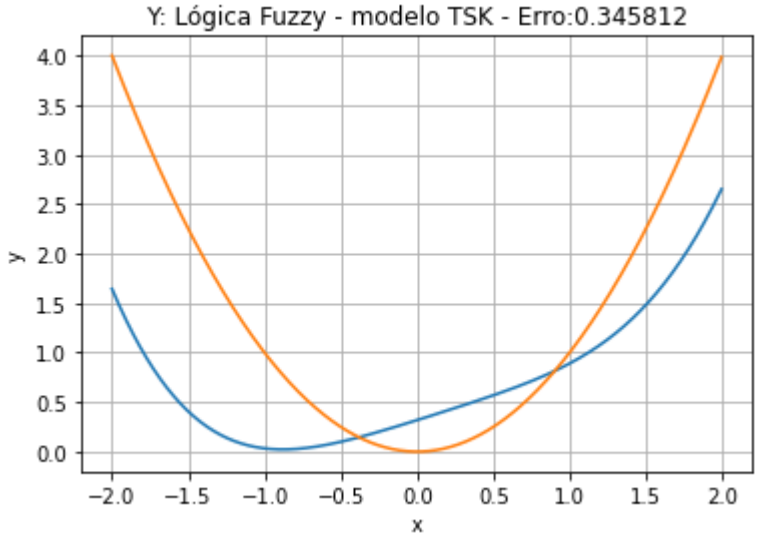
epoca: 16



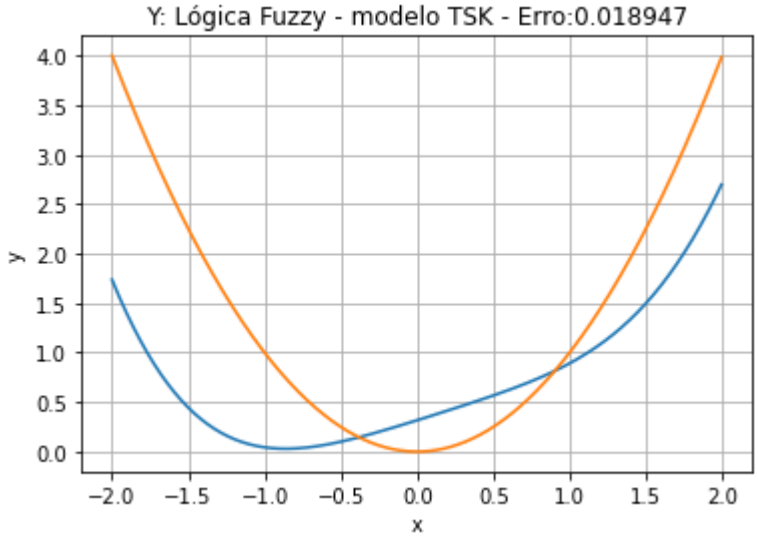
epoca: 17



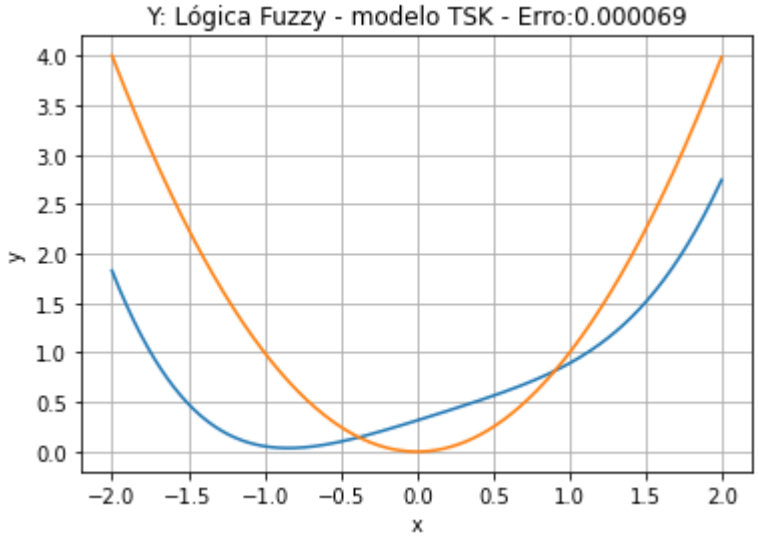
epoca: 18



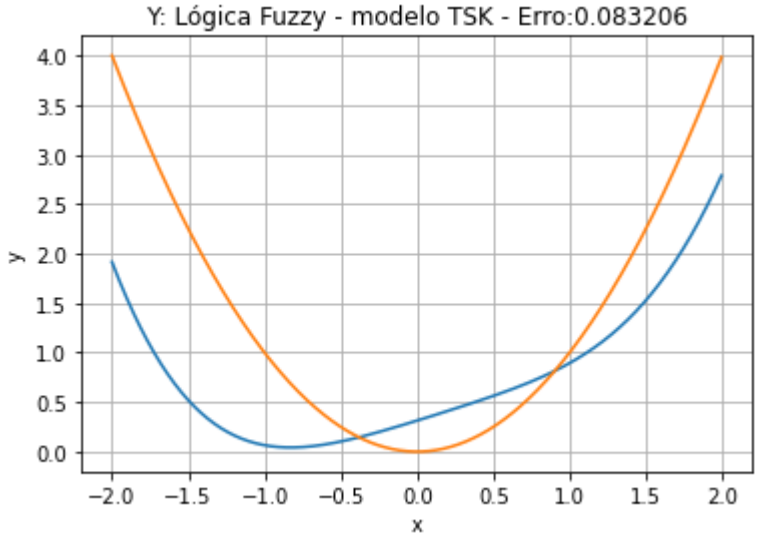
epoca: 19



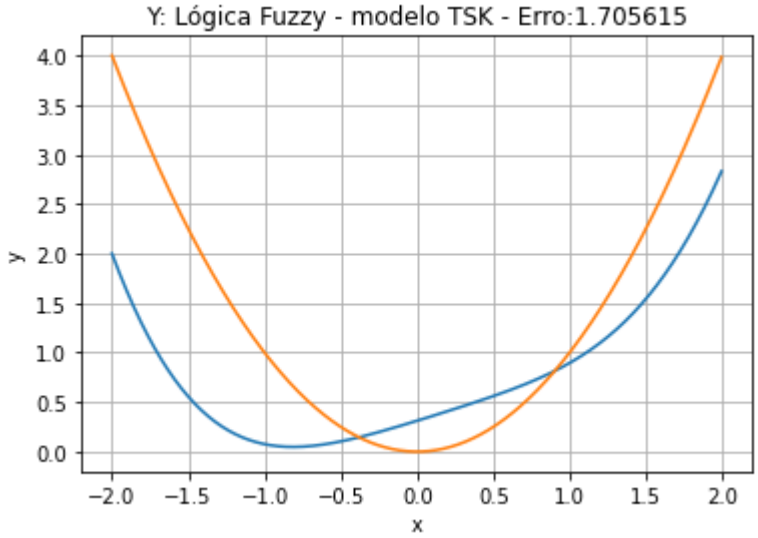
epoca: 20



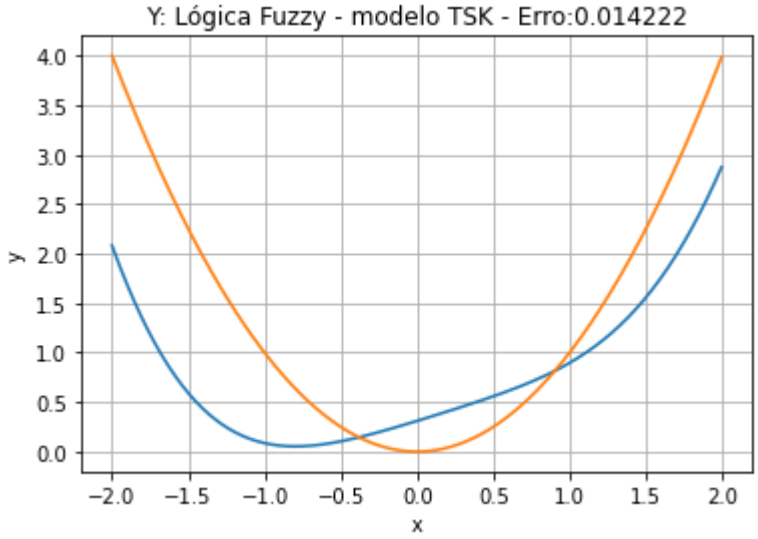
epoca: 21



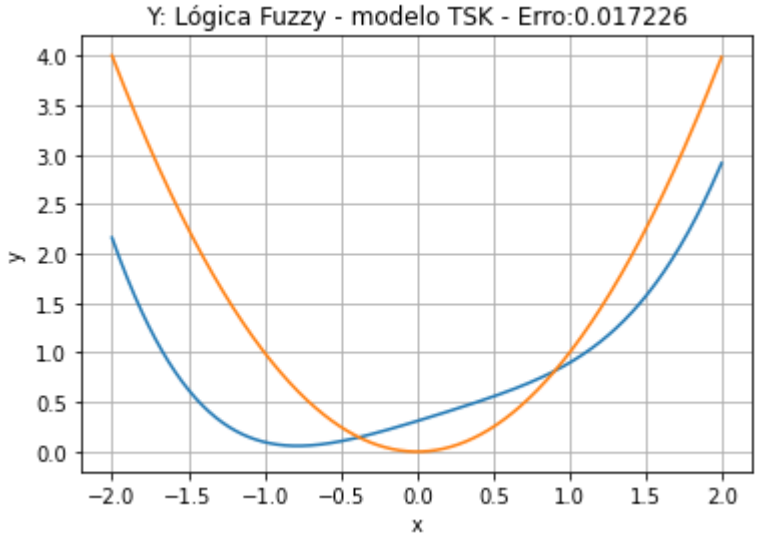
epoca: 22



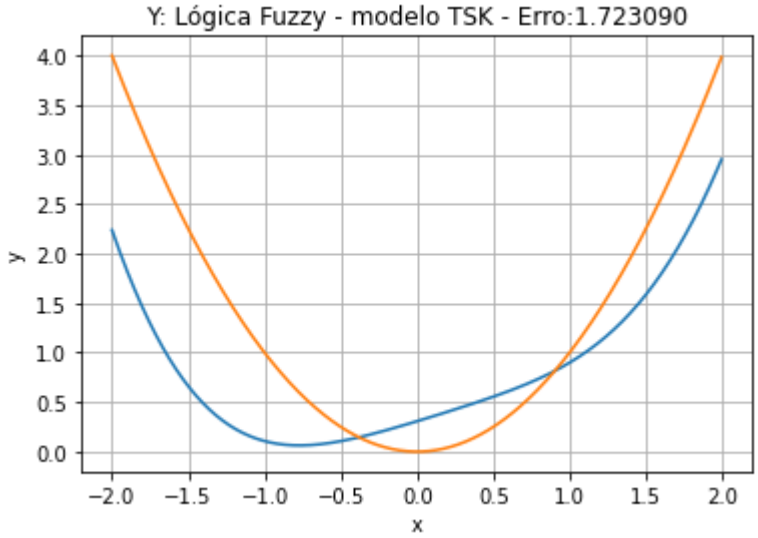
epoca: 23



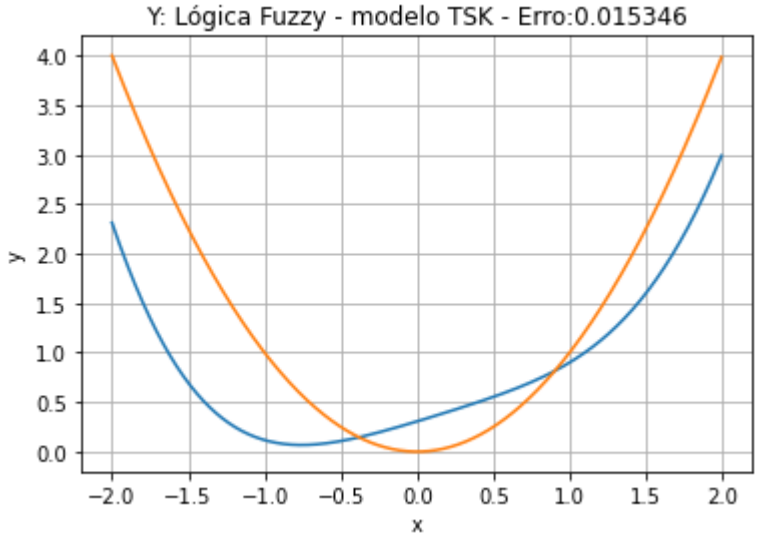
epoca: 24



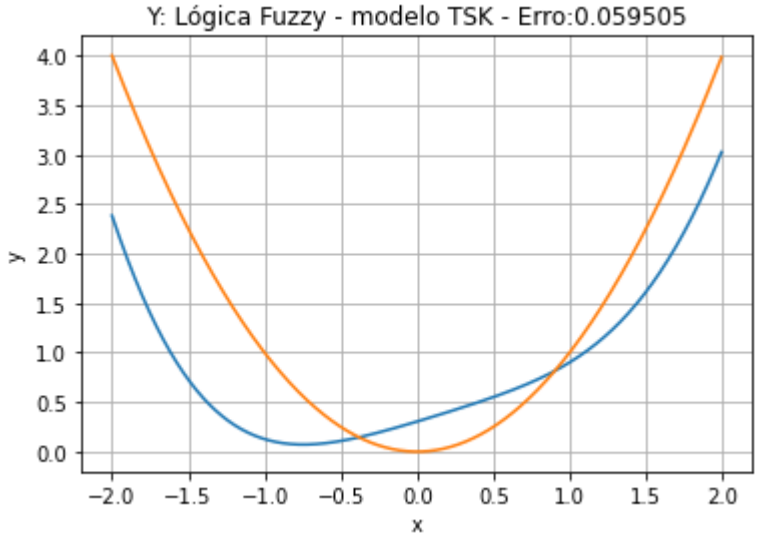
epoca: 25



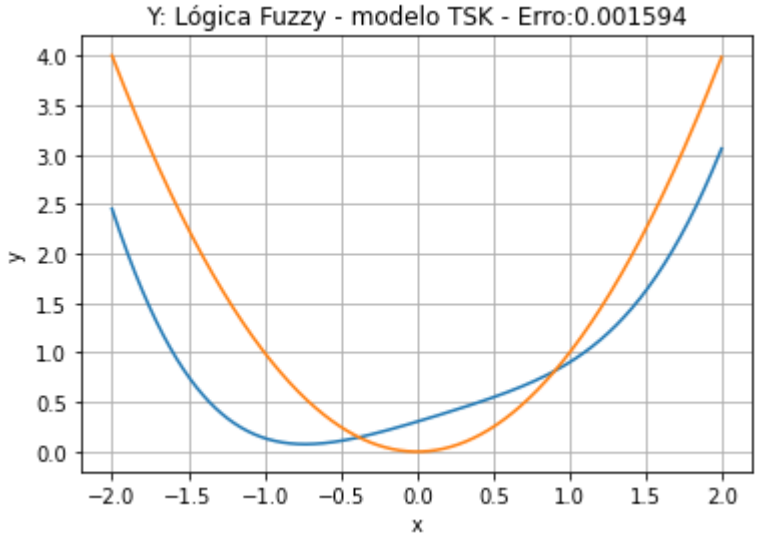
epoca: 26



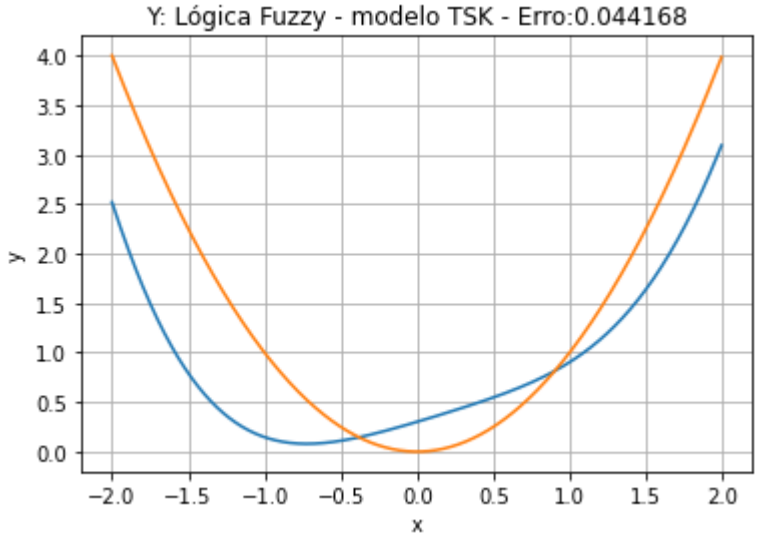
epoca: 27



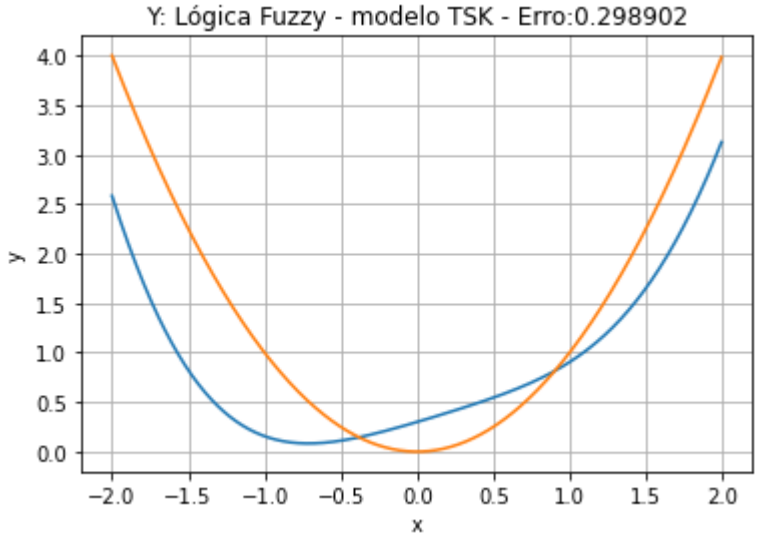
epoca: 28



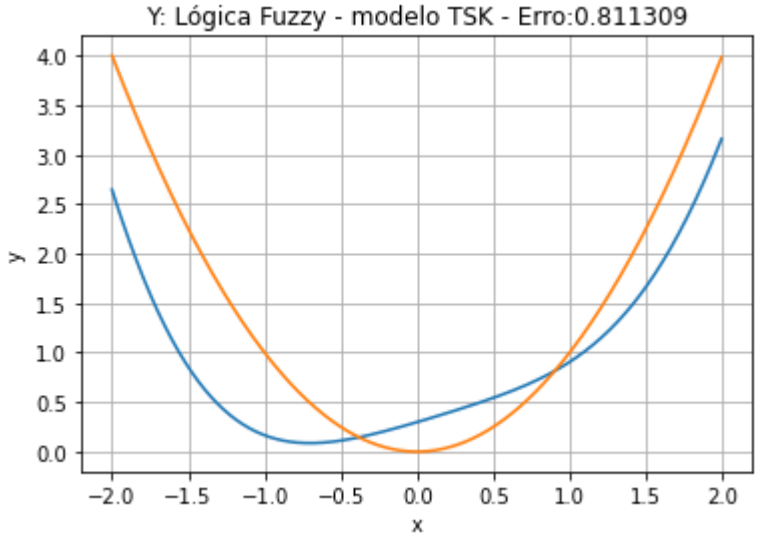
epoca: 29



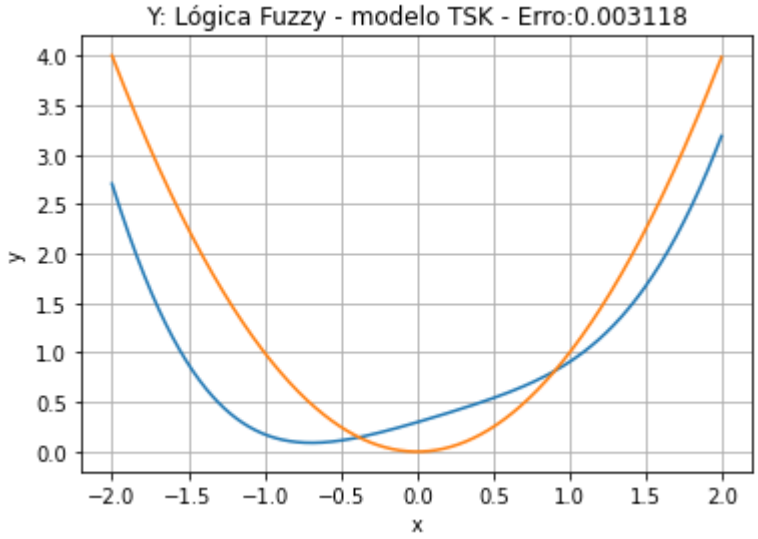
epoca: 30



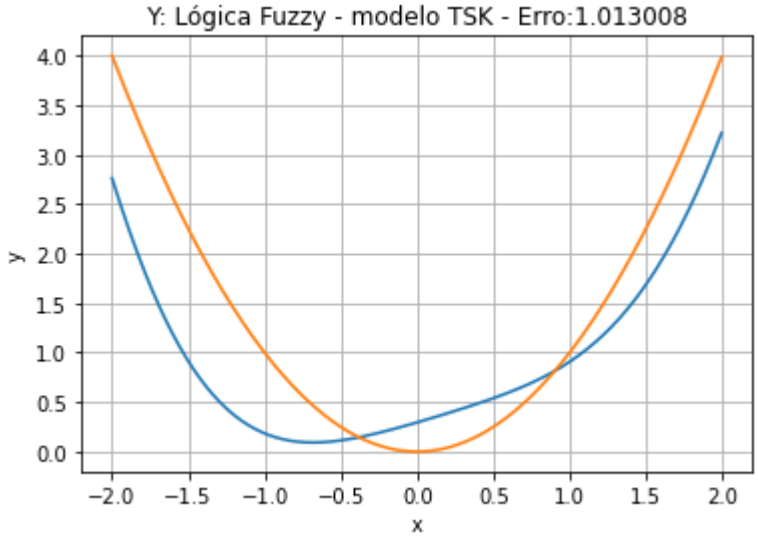
epoca: 31



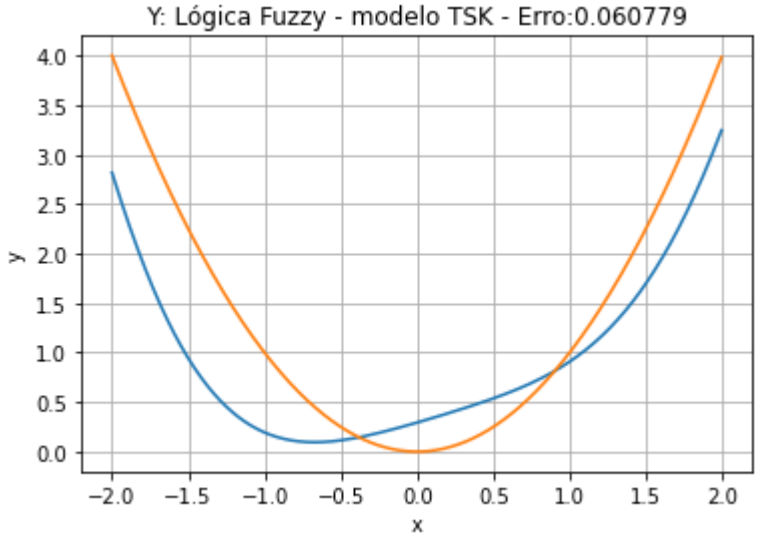
epoca: 32



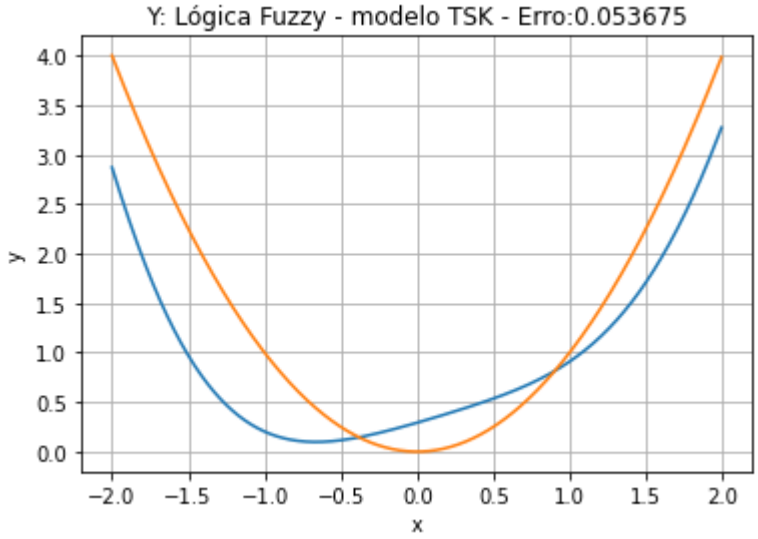
epoca: 33



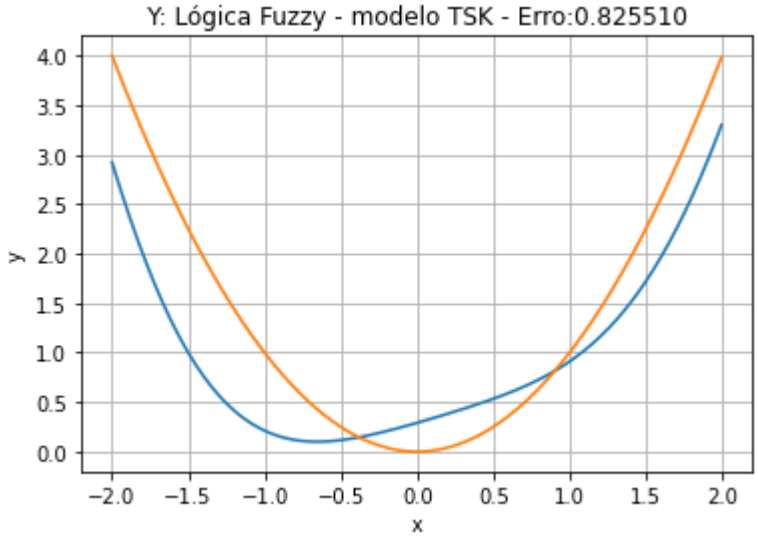
epoca: 34



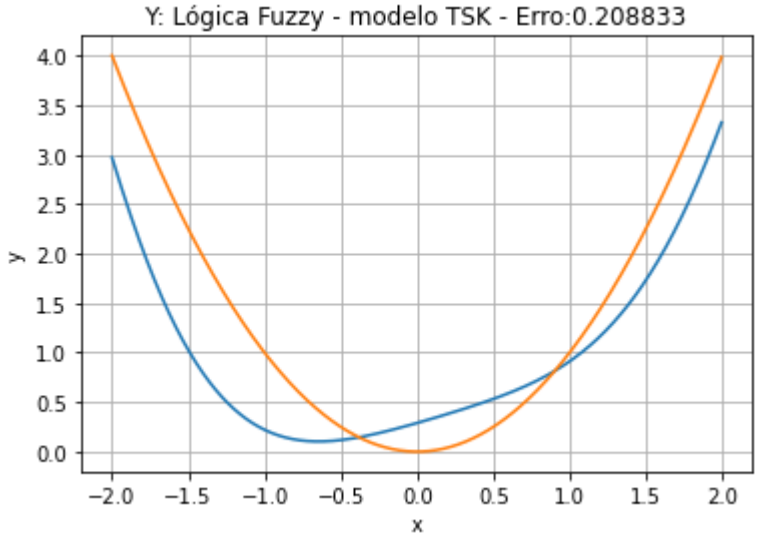
epoca: 35



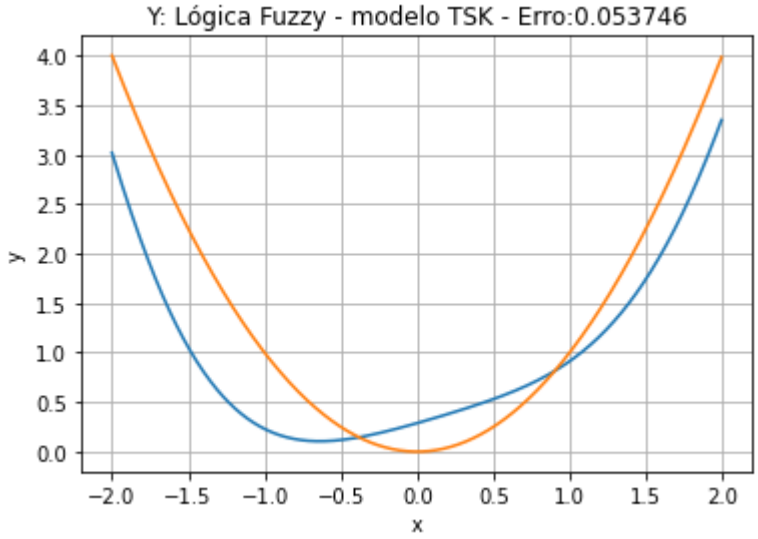
epoca: 36



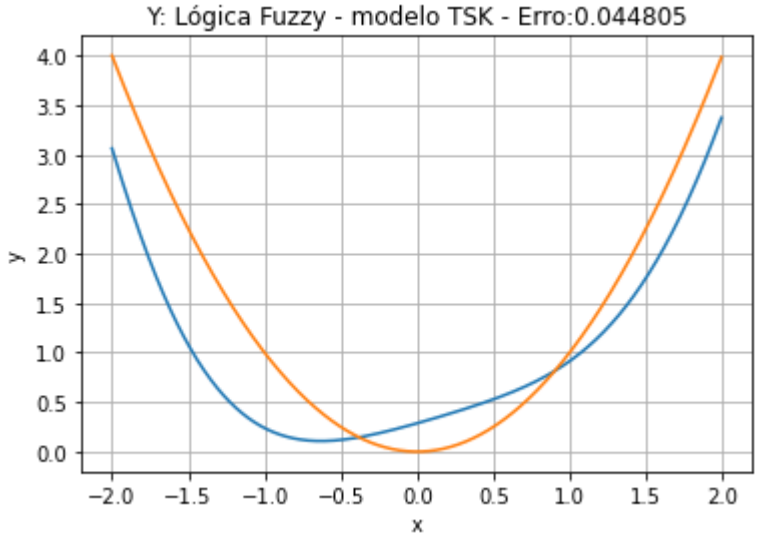
epoca: 37



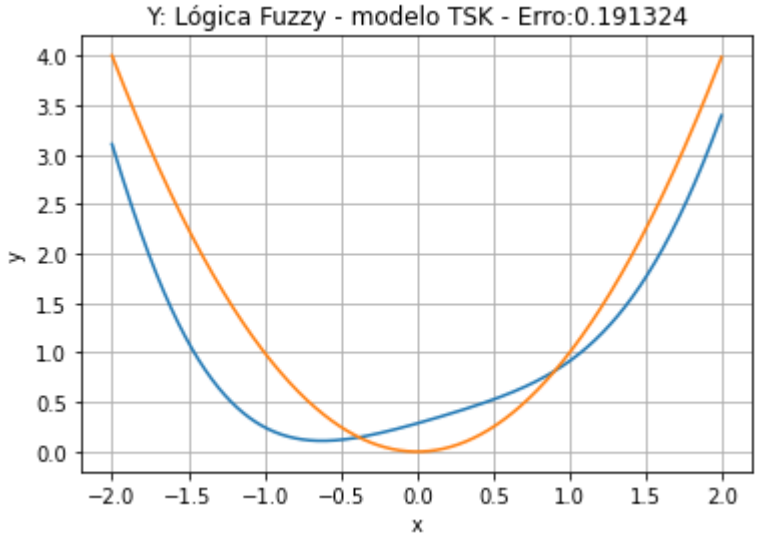
epoca: 38



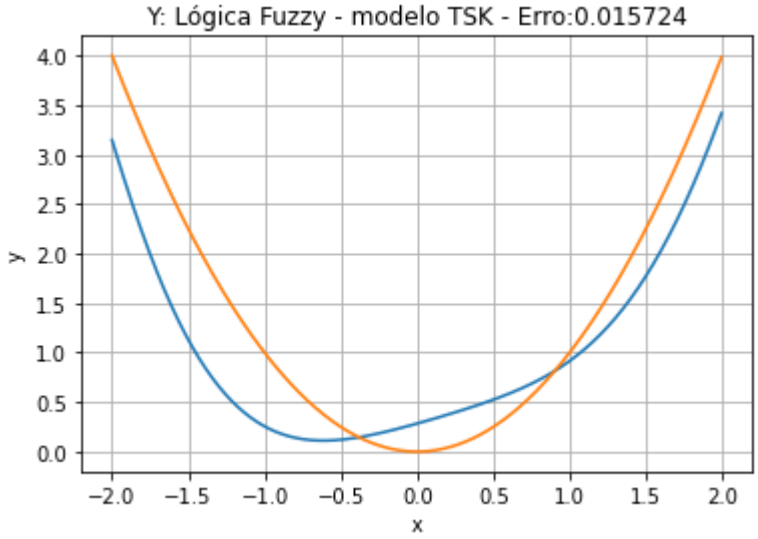
epoca: 39



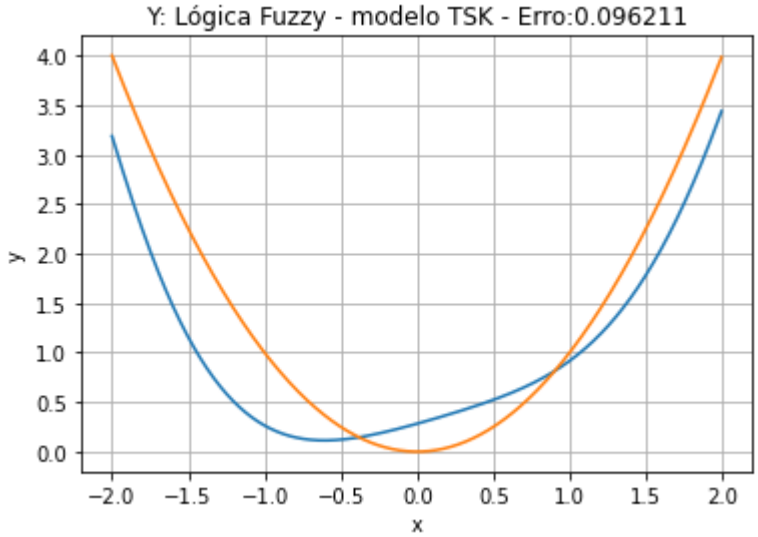
epoca: 40



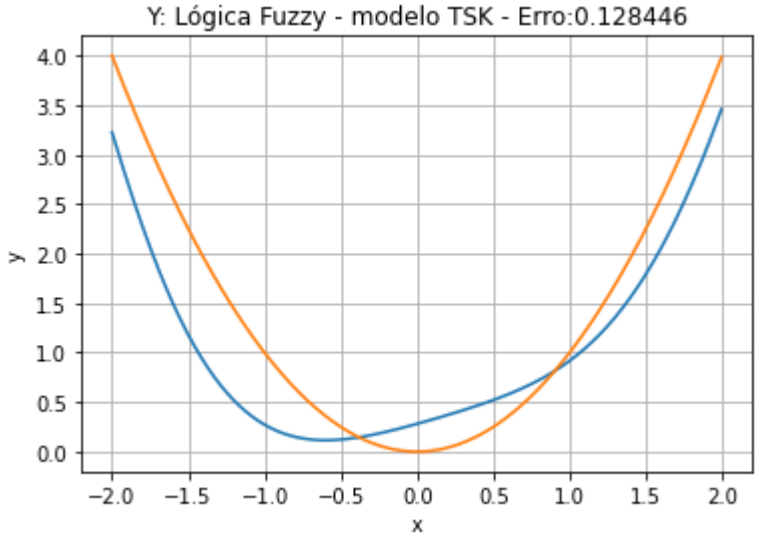
epoca: 41



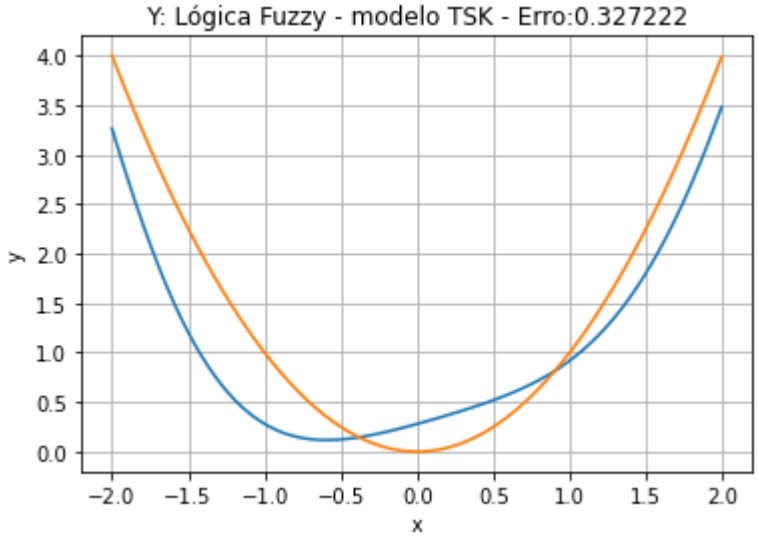
epoca: 42



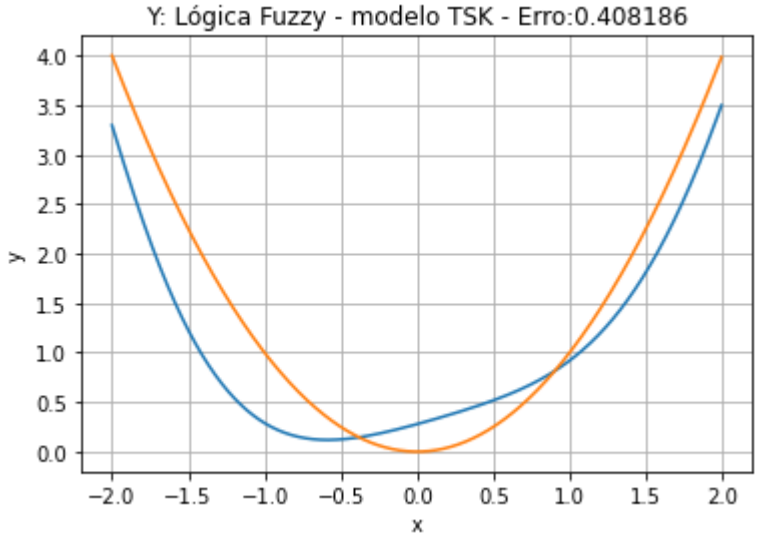
epoca: 43



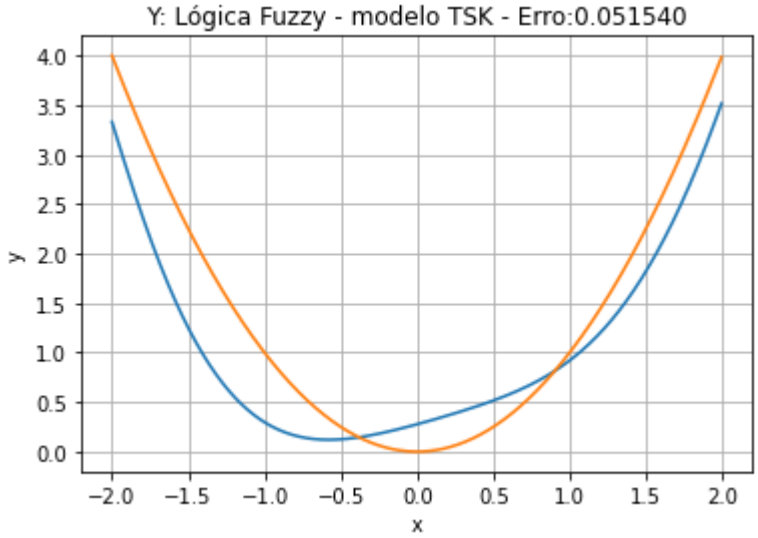
epoca: 44



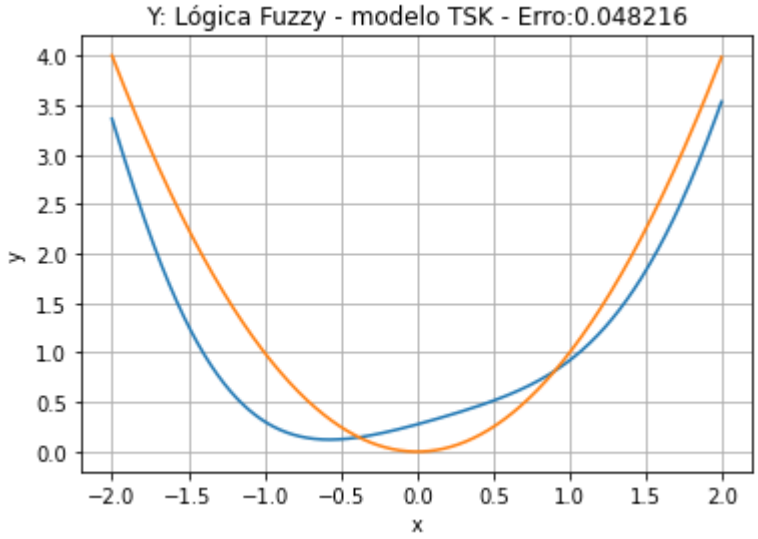
epoca: 45



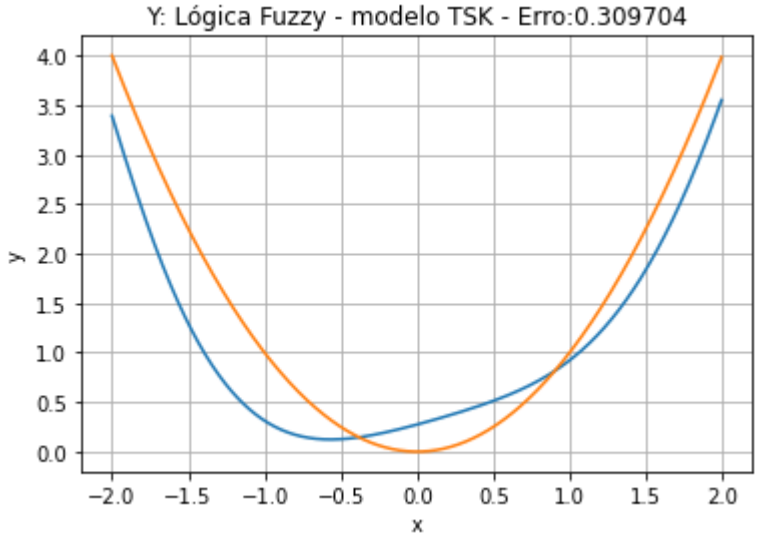
epoca: 46



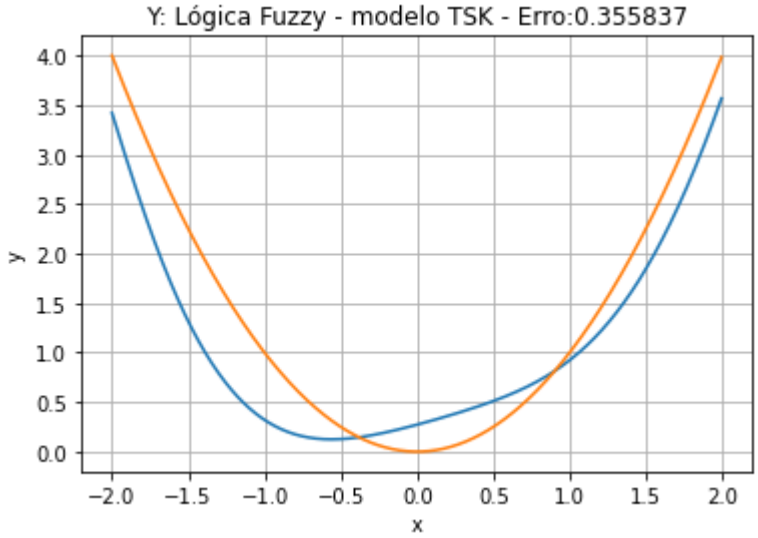
epoca: 47



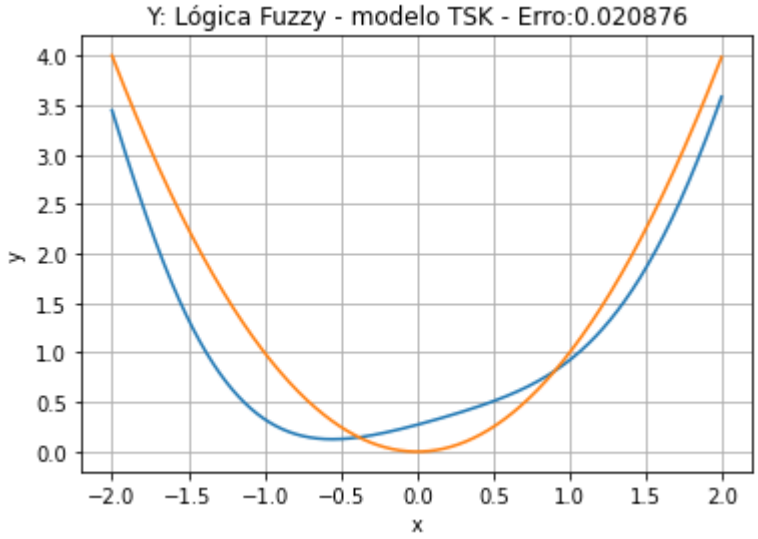
epoca: 48



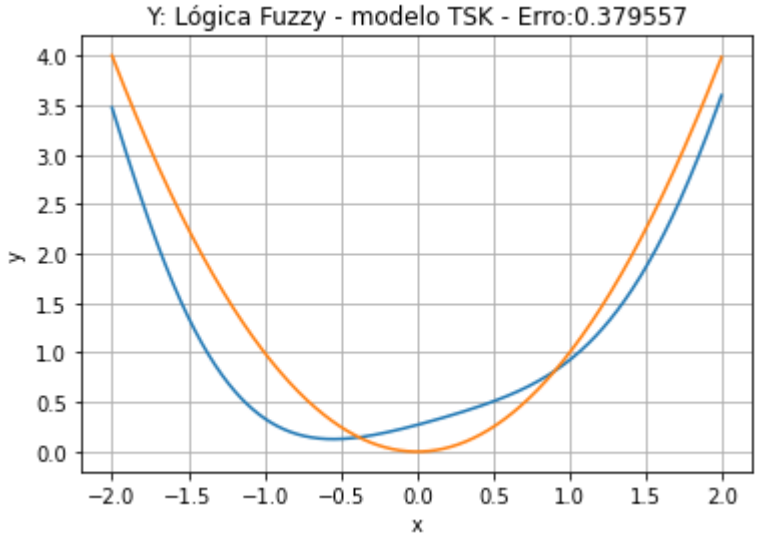
epoca: 49



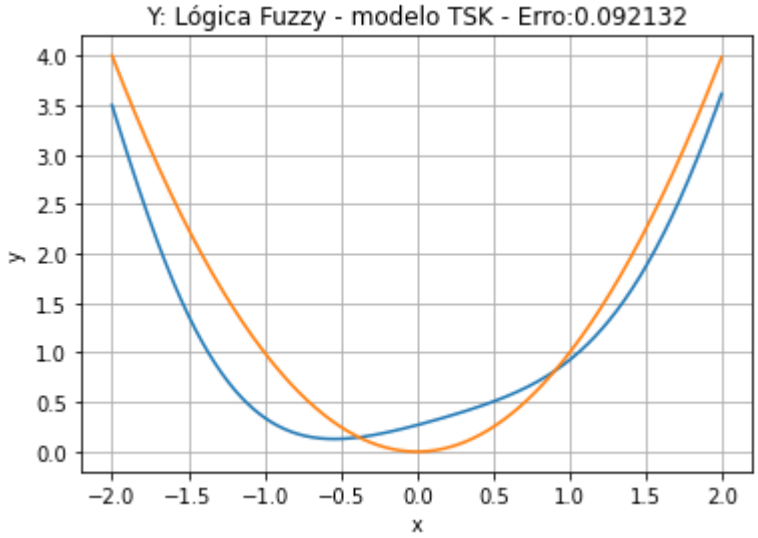
epoca: 50



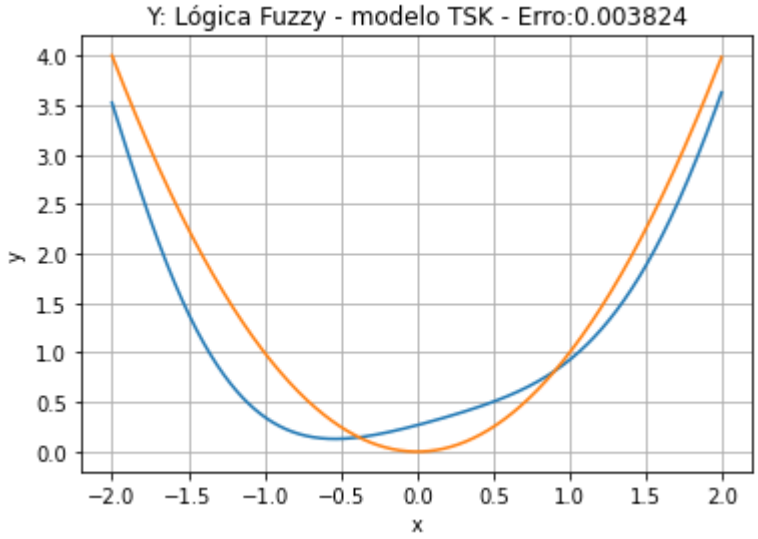
epoca: 51



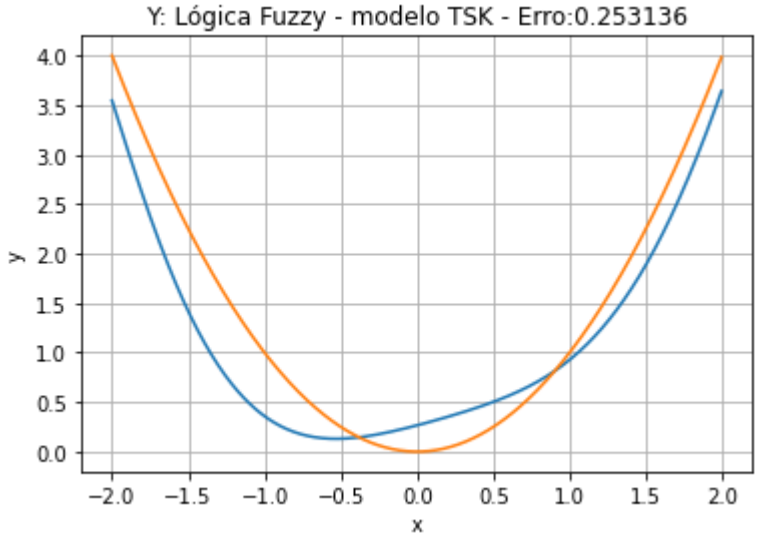
epoca: 52



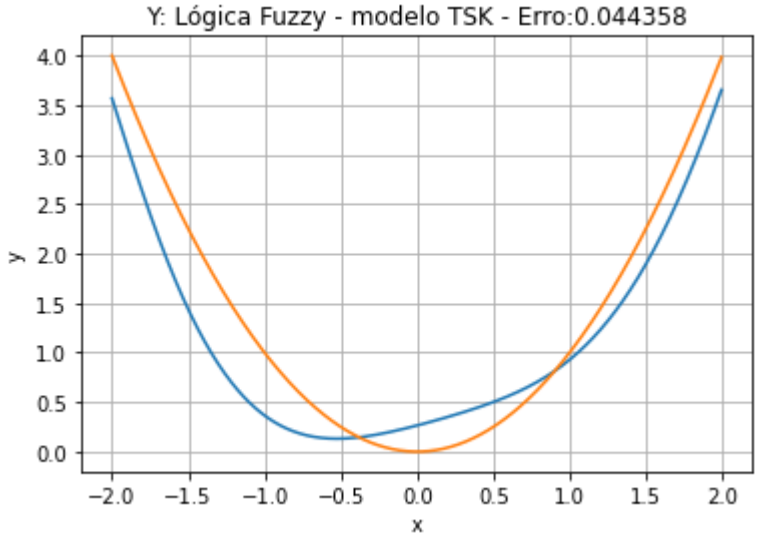
epoca: 53



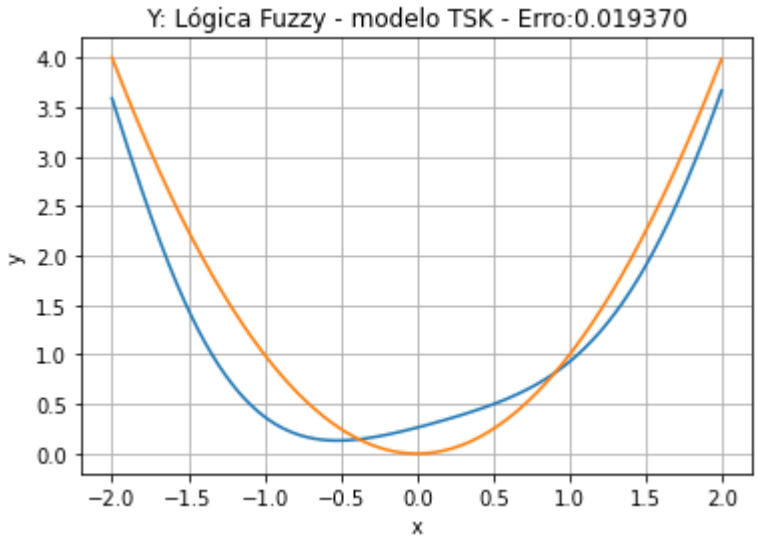
epoca: 54



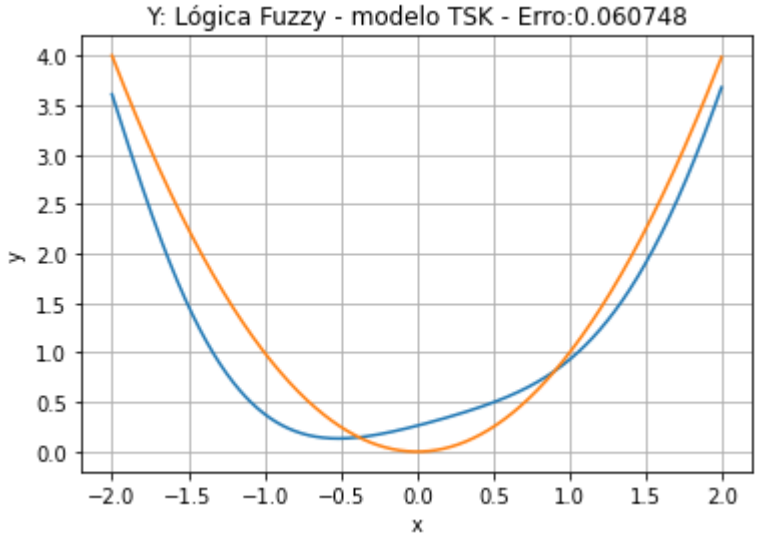
epoca: 55



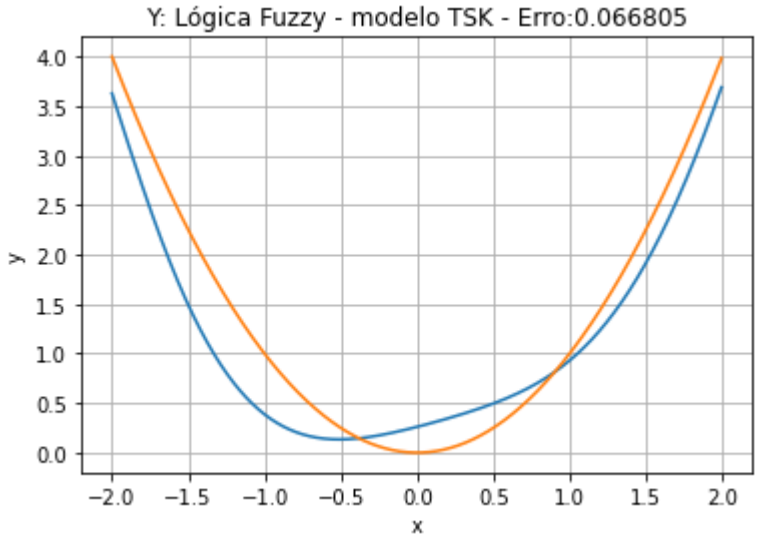
epoca: 56



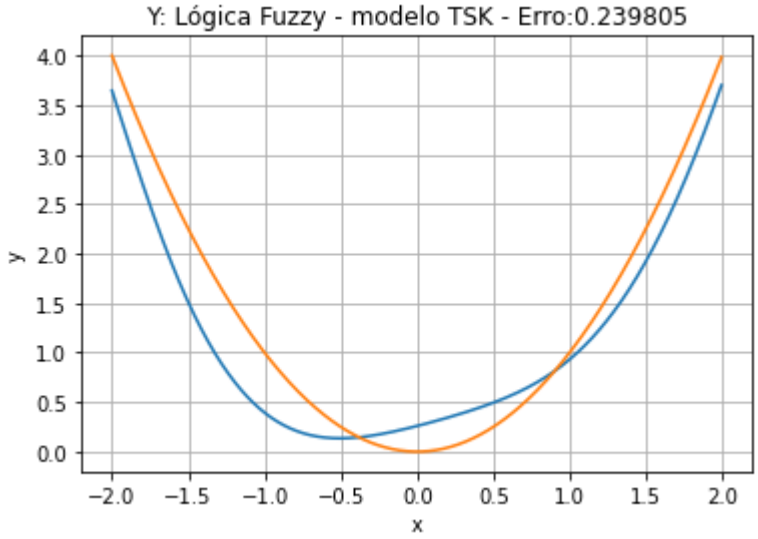
epoca: 57



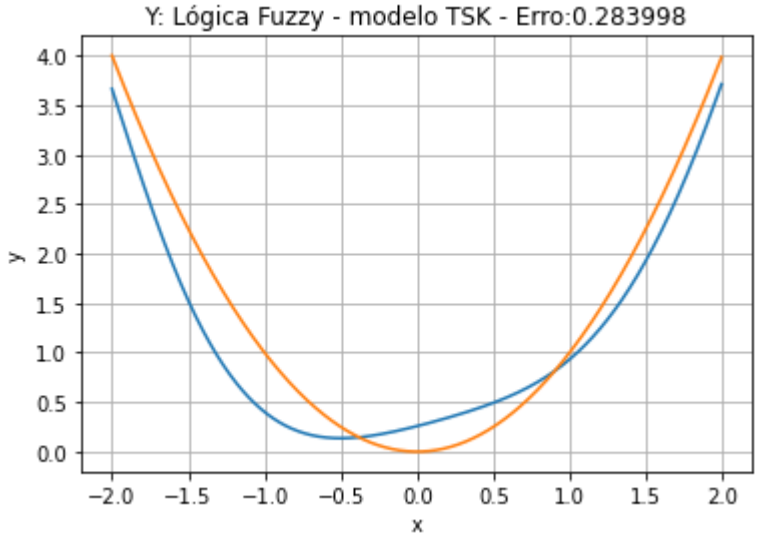
epoca: 58



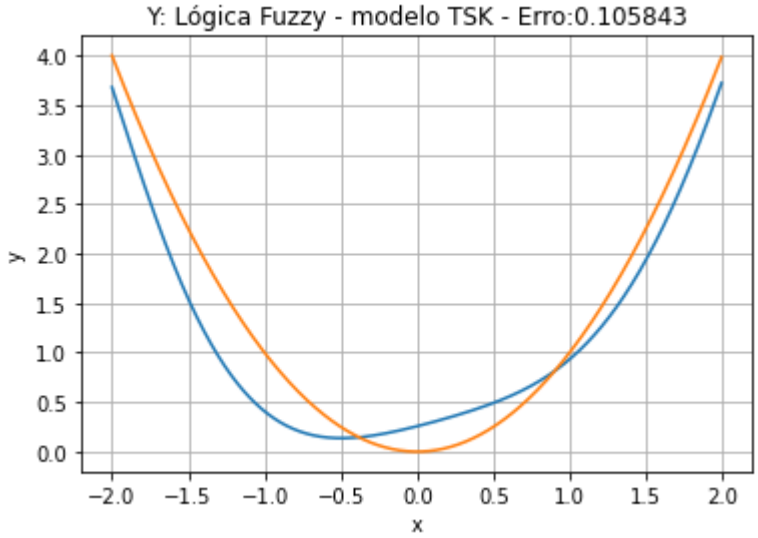
epoca: 59



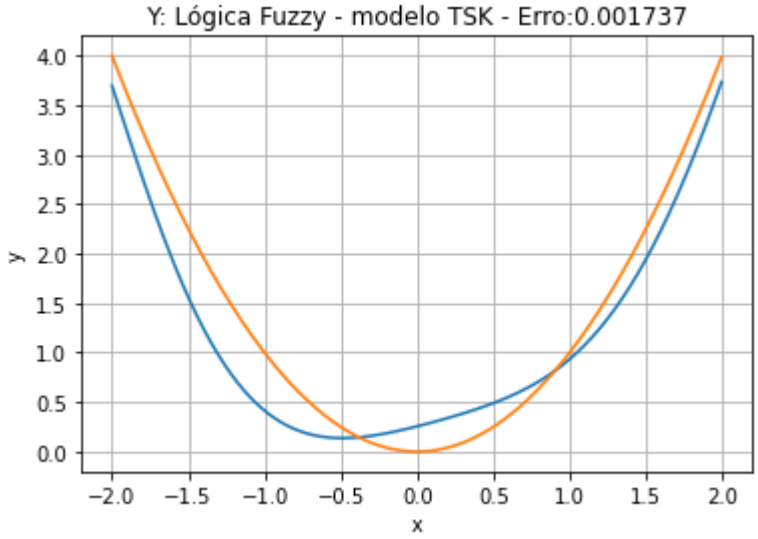
epoca: 60



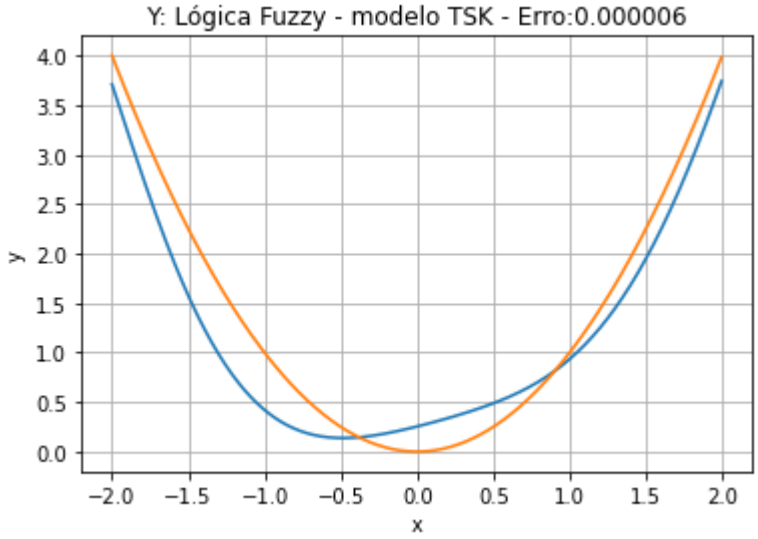
epoca: 61



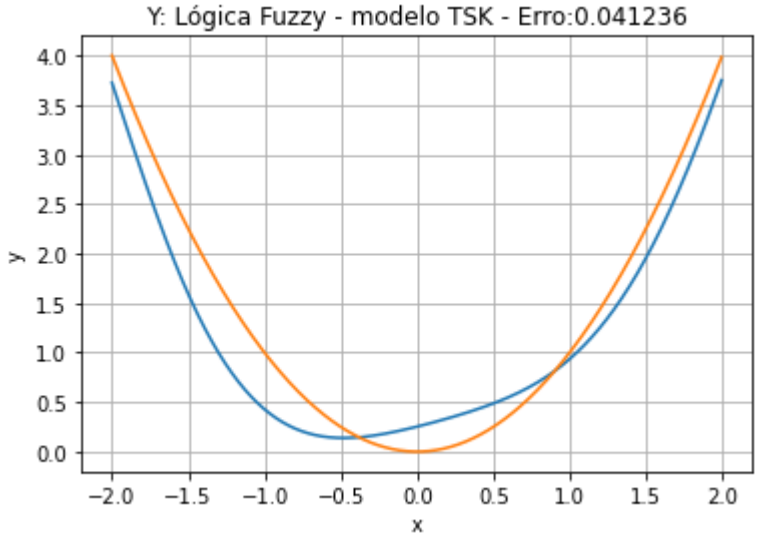
epoca: 62



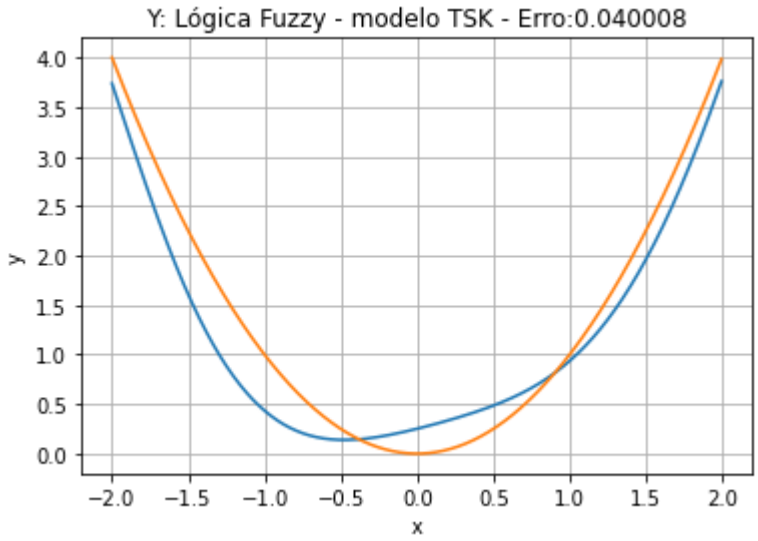
epoca: 63



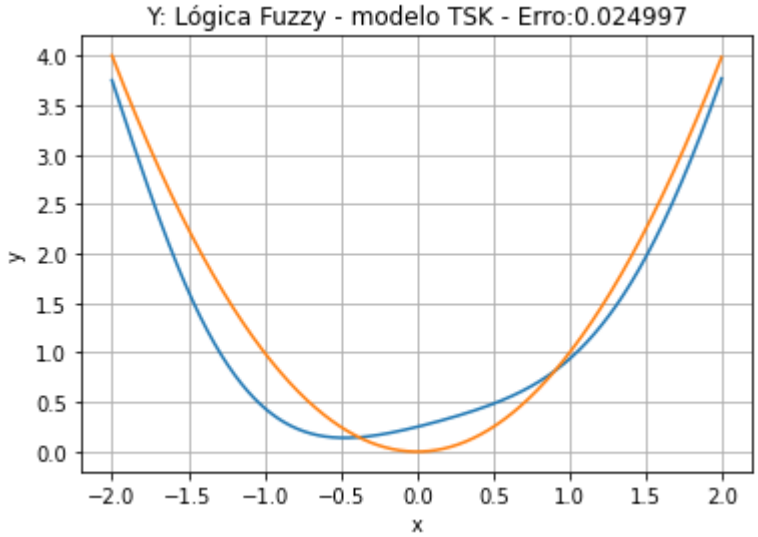
epoca: 64



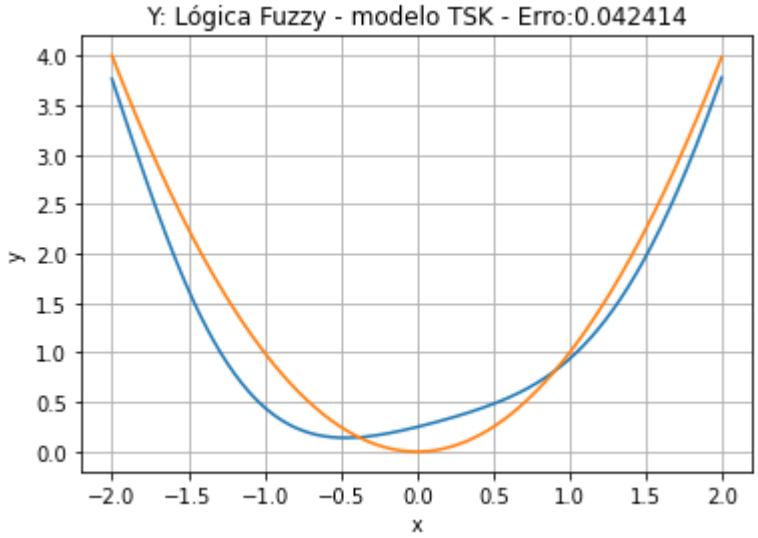
epoca: 65



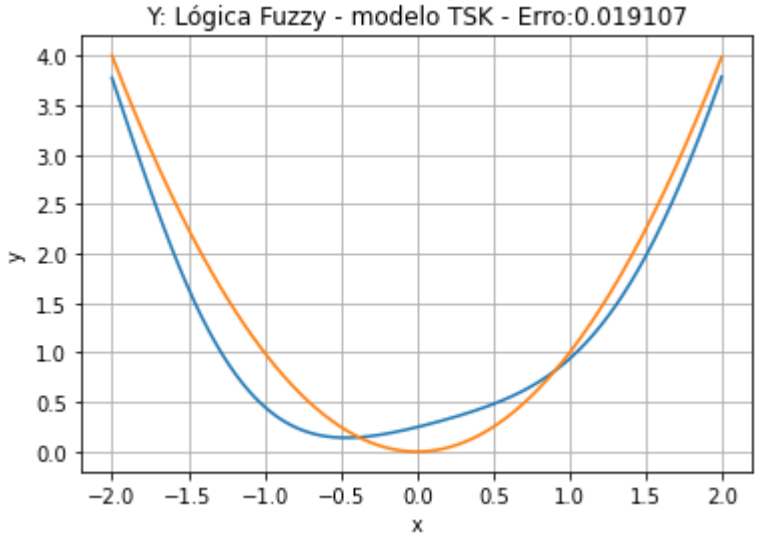
epoca: 66



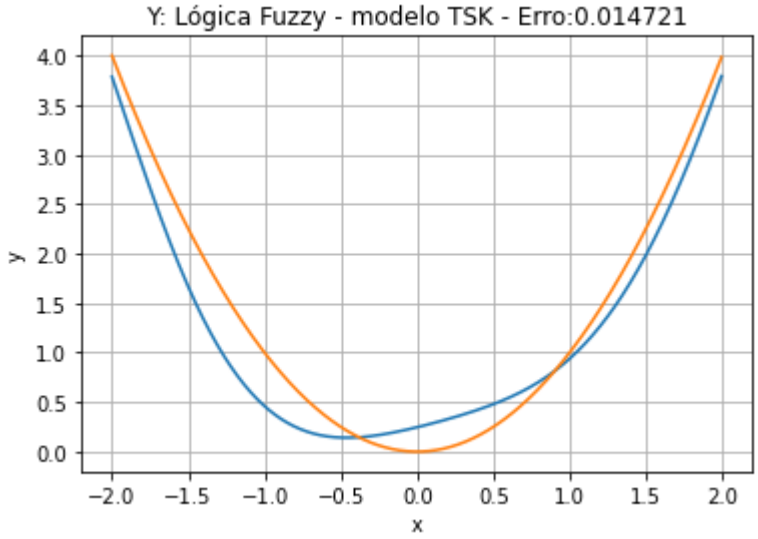
epoca: 67



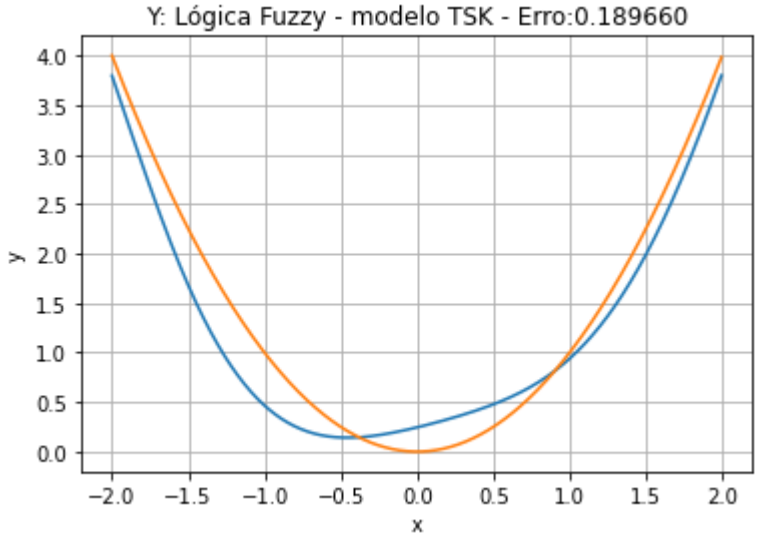
epoca: 68



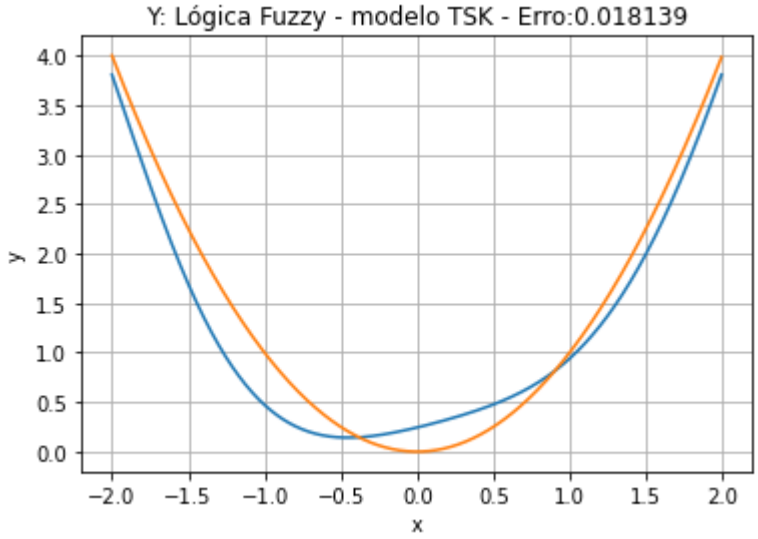
epoca: 69



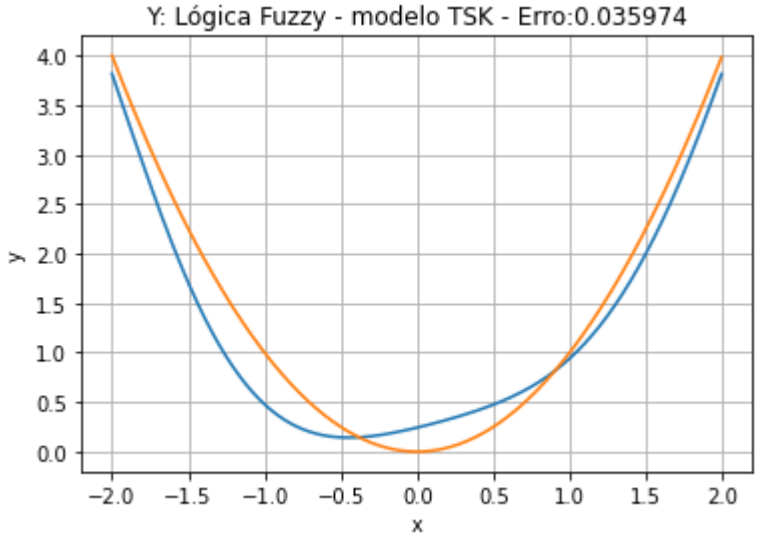
epoca: 70



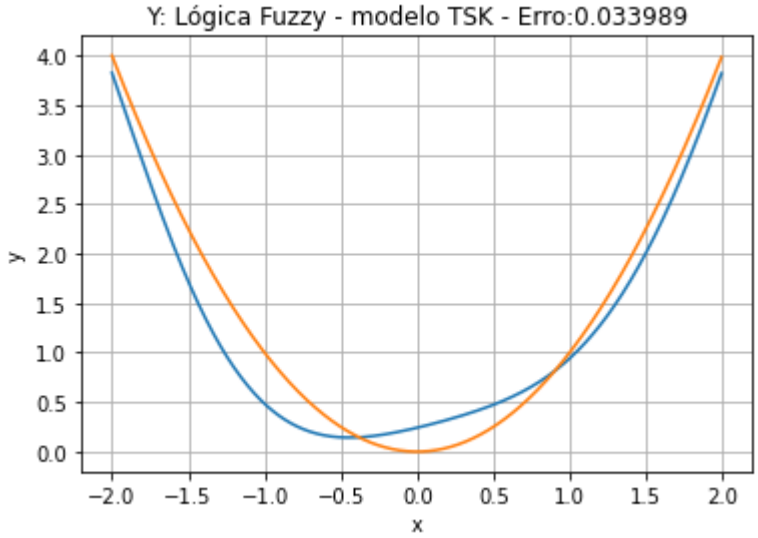
epoca: 71



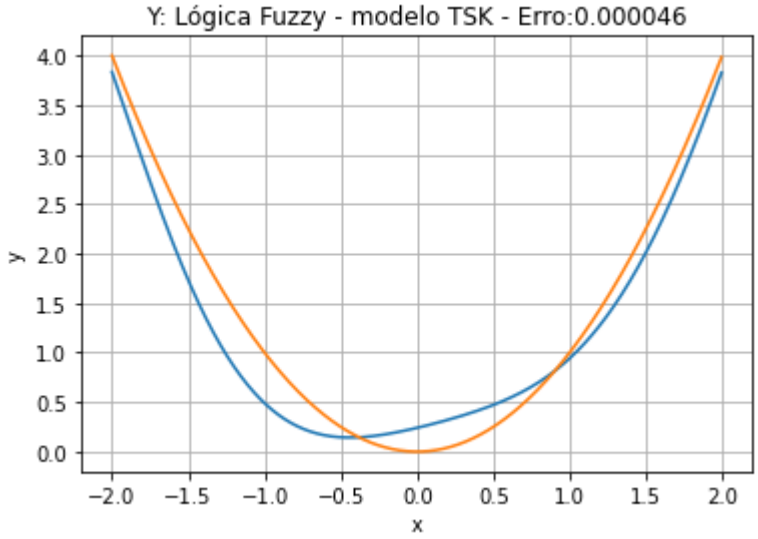
epoca: 72



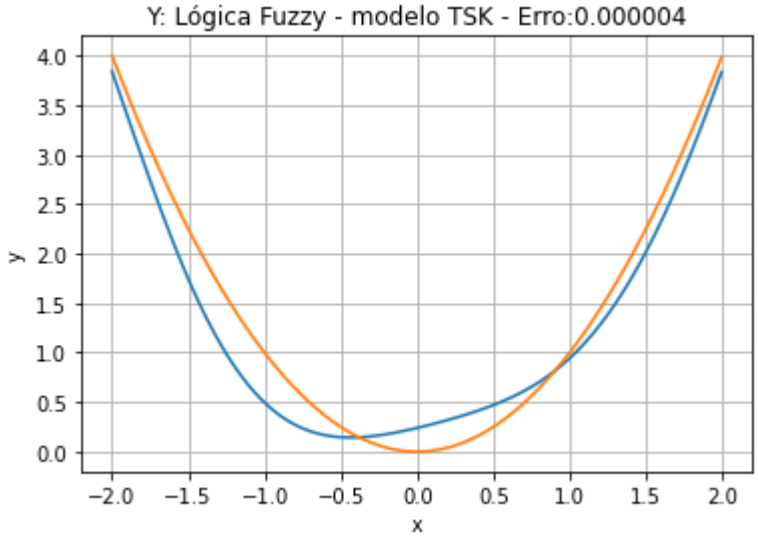
epoca: 73



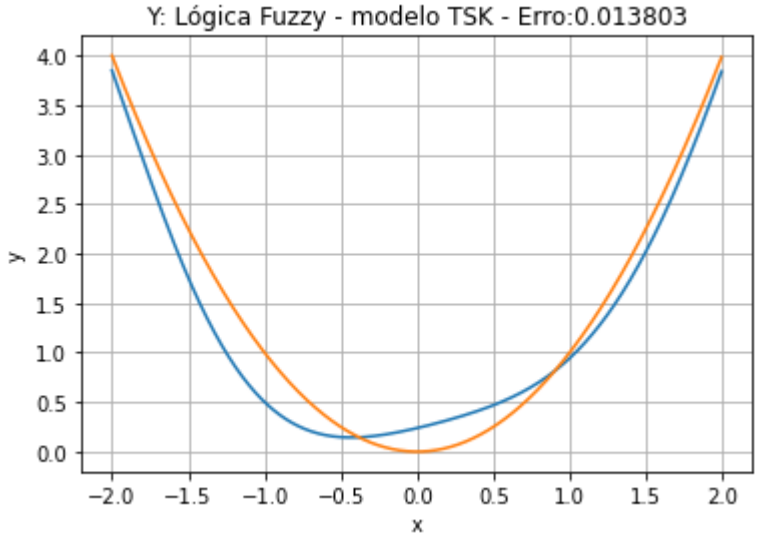
epoca: 74



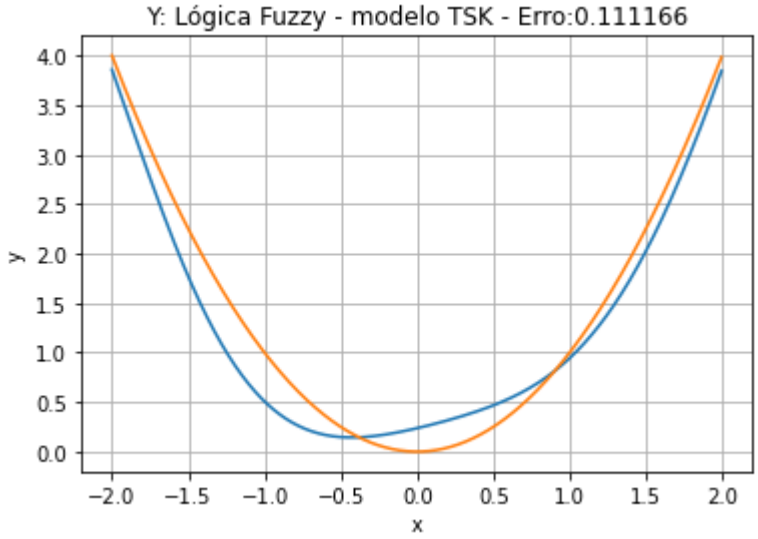
epoca: 75



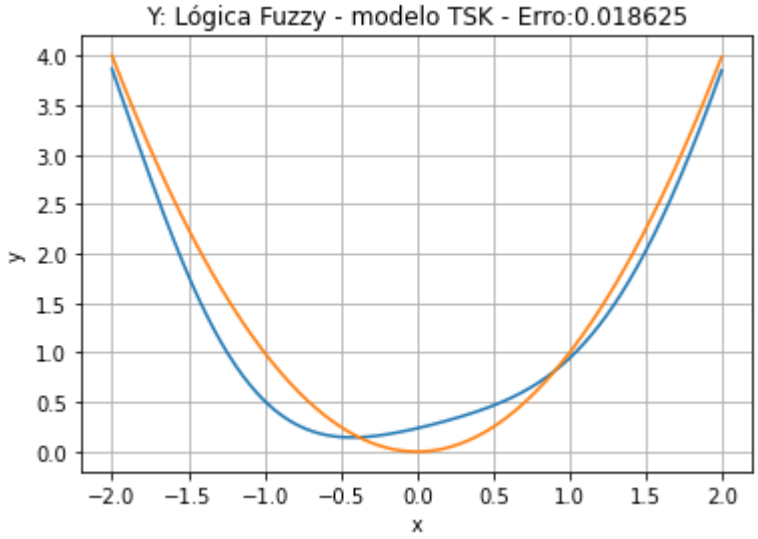
epoca: 76



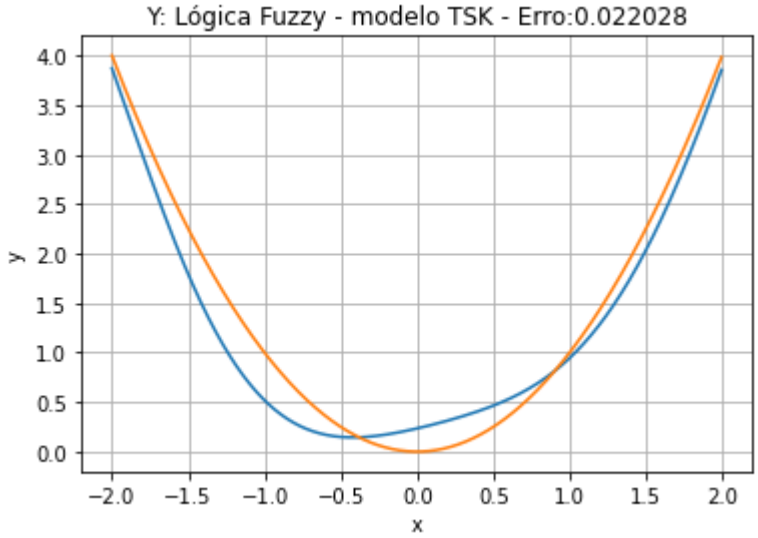
epoca: 77



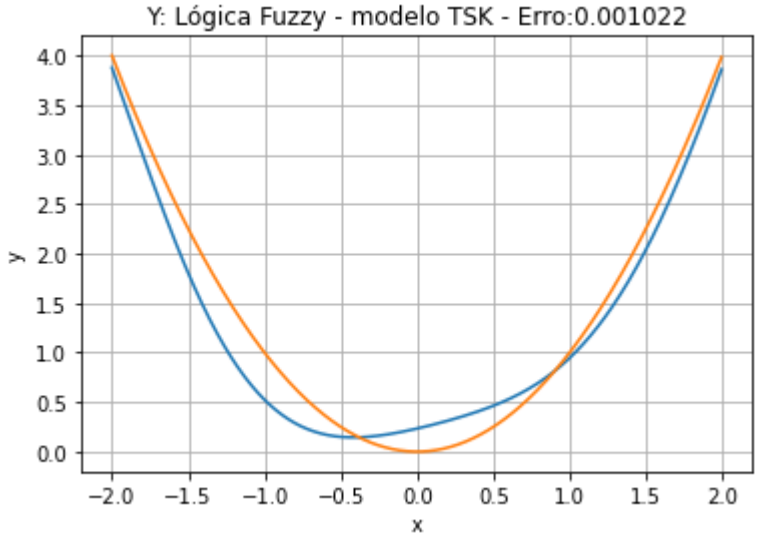
epoca: 78



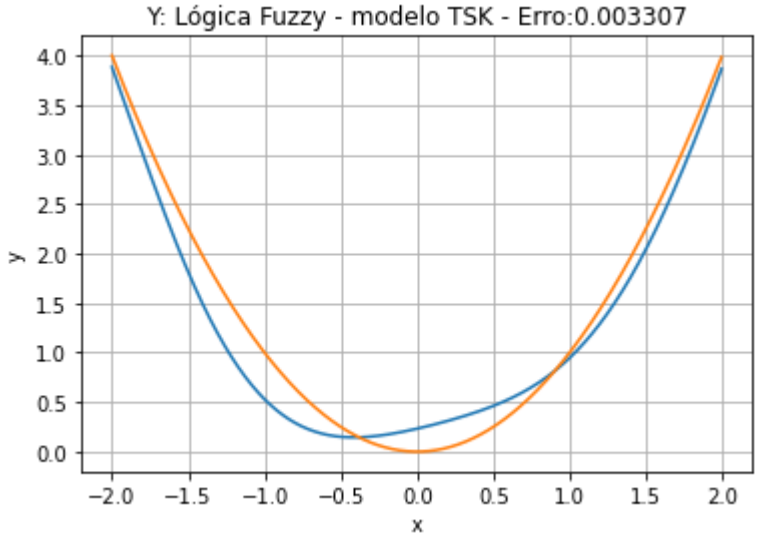
epoca: 79



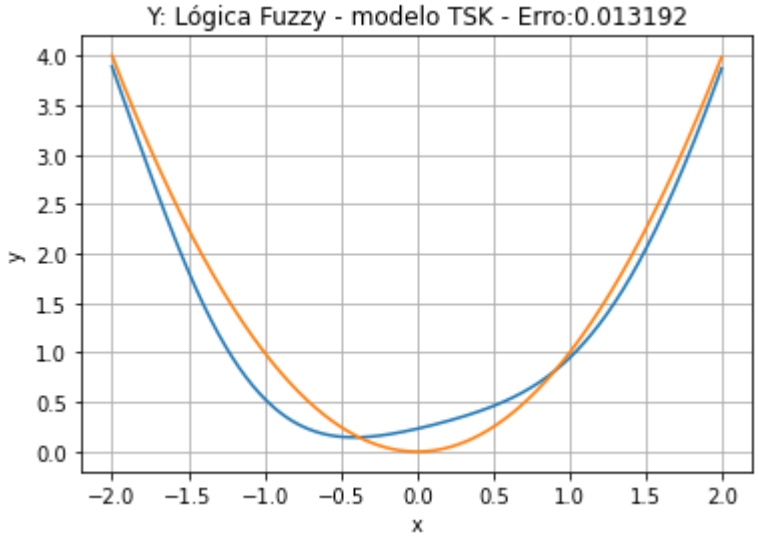
epoca: 80



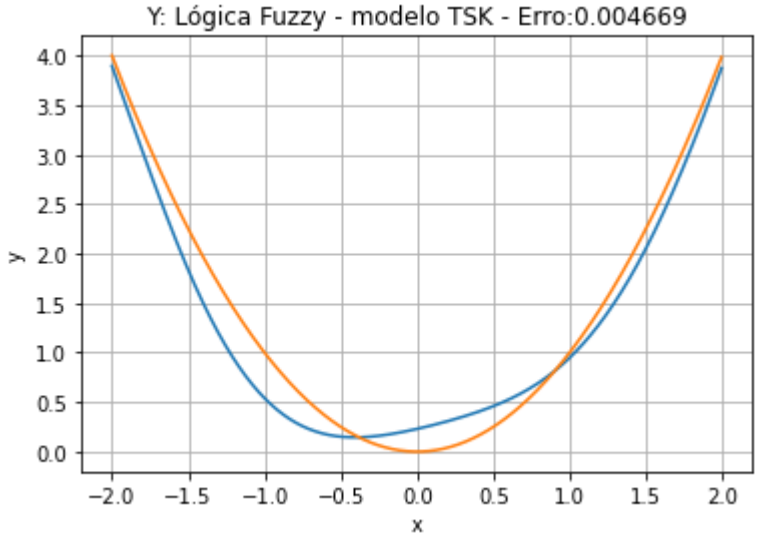
epoca: 81



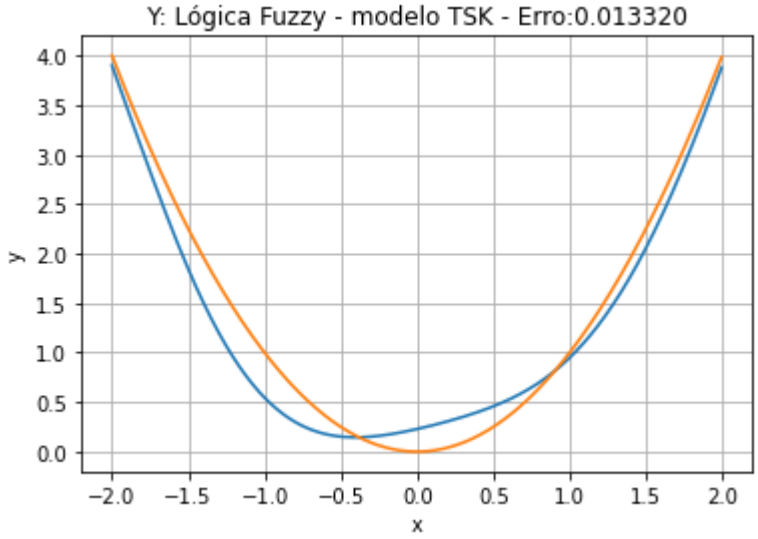
epoca: 82



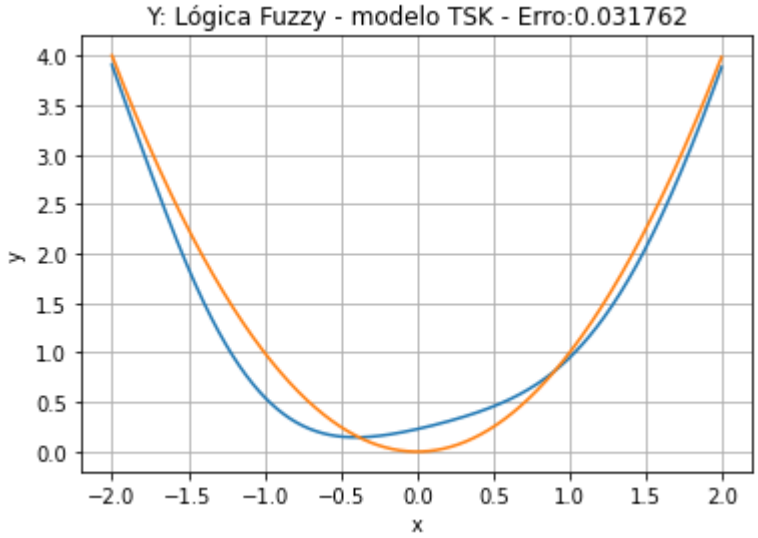
epoca: 83



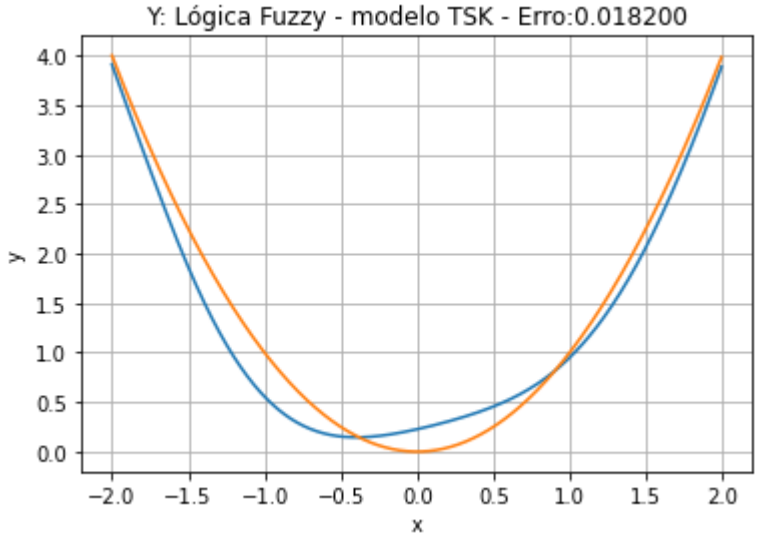
epoca: 84



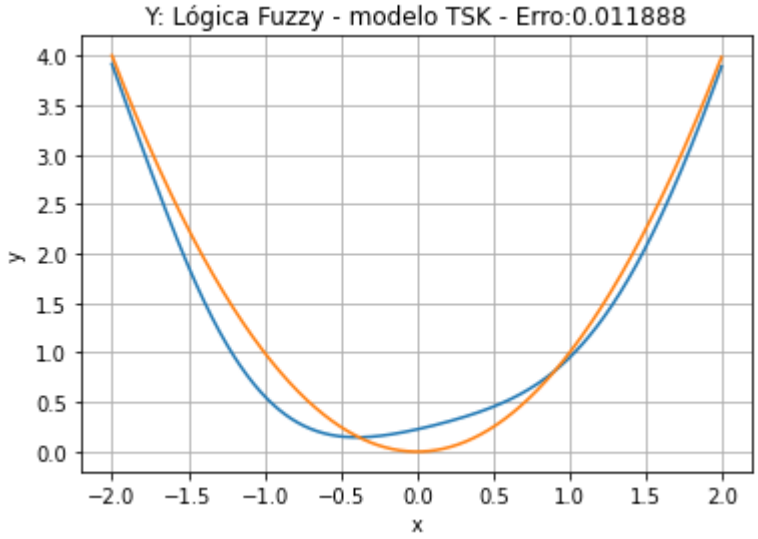
epoca: 85



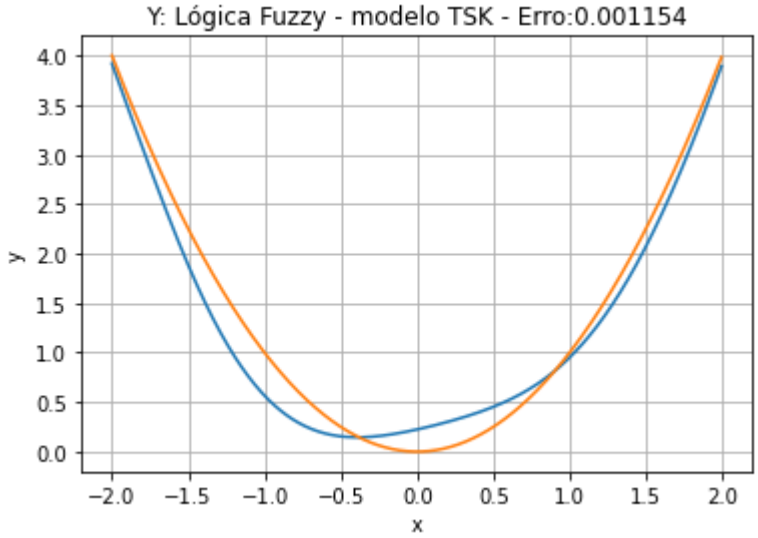
epoca: 86



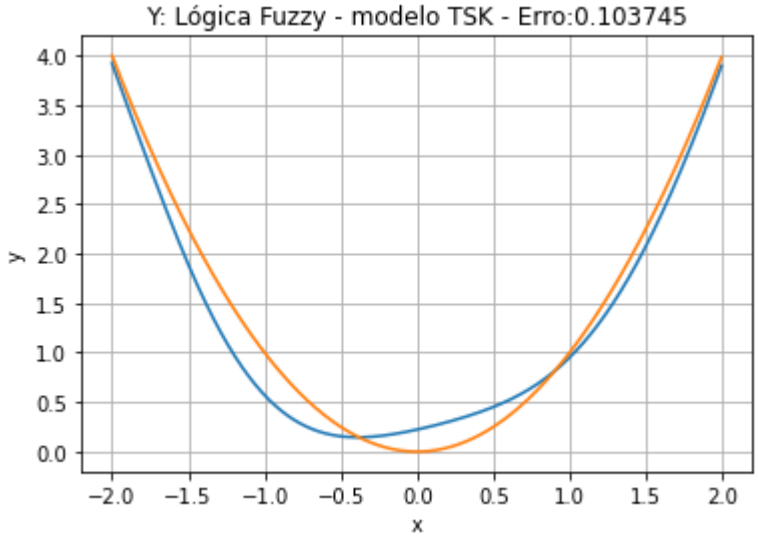
epoca: 87



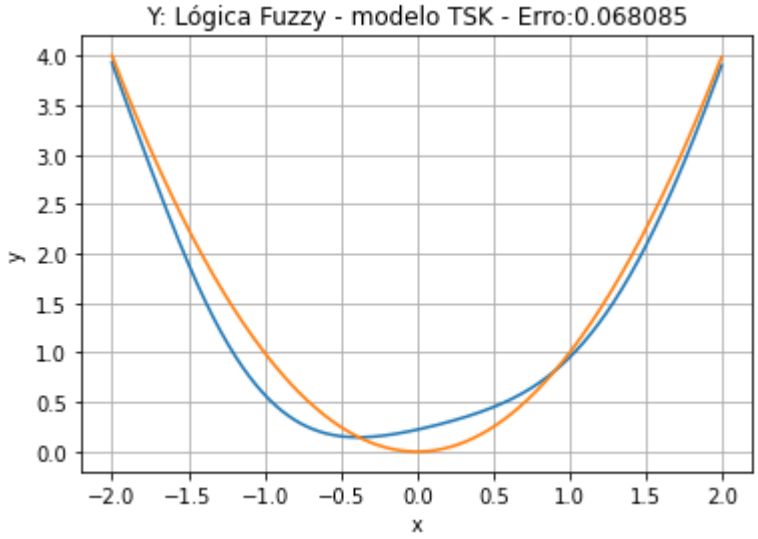
epoca: 88



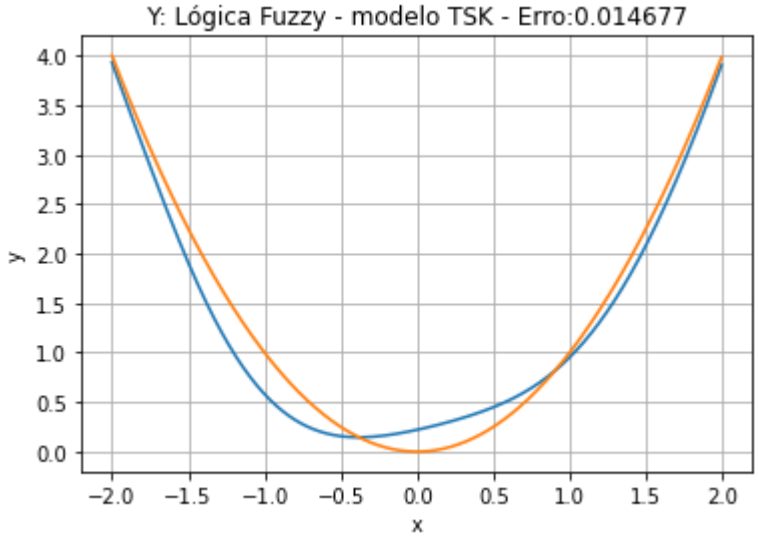
epoca: 89



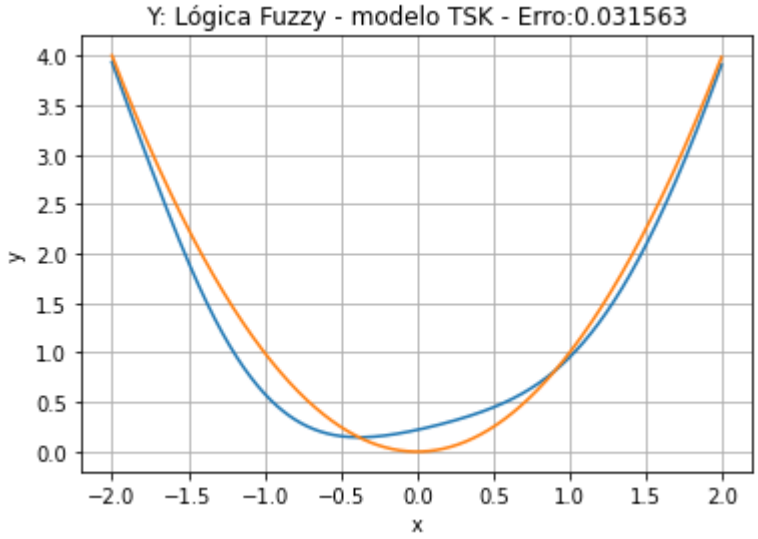
epoca: 90



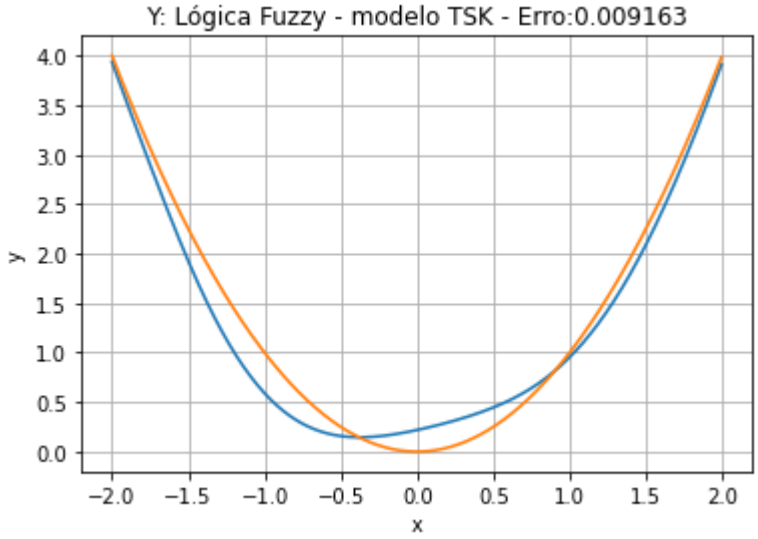
epoca: 91



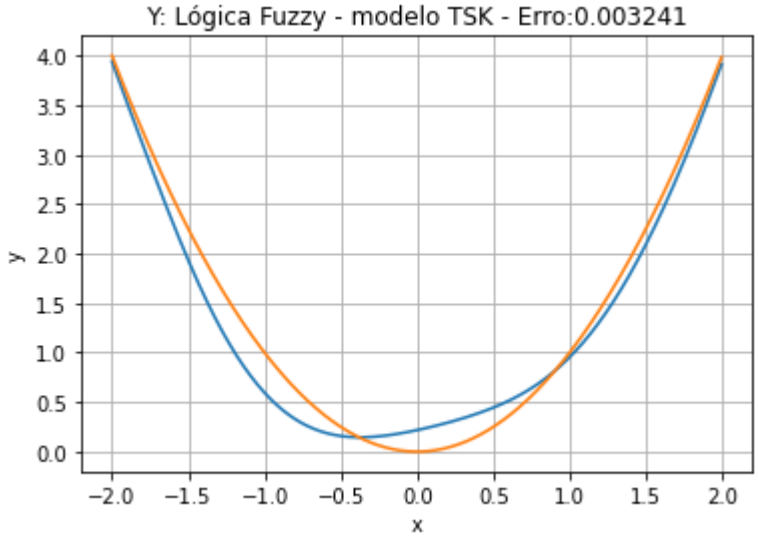
epoca: 92



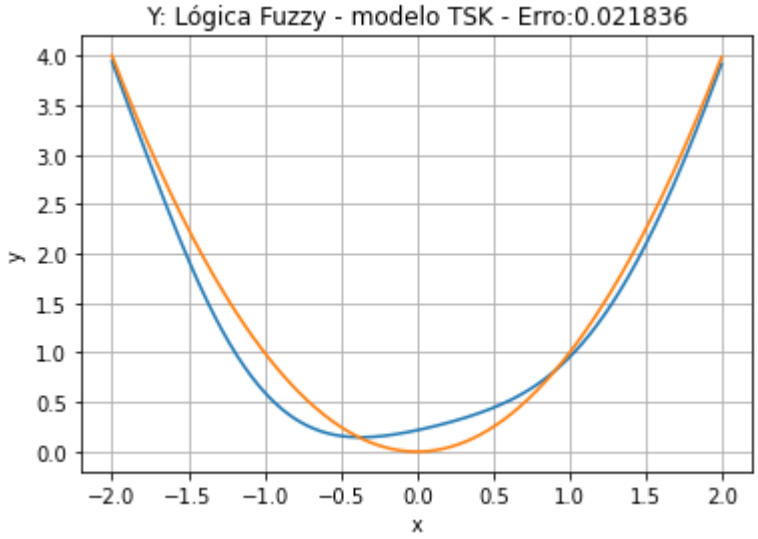
epoca: 93



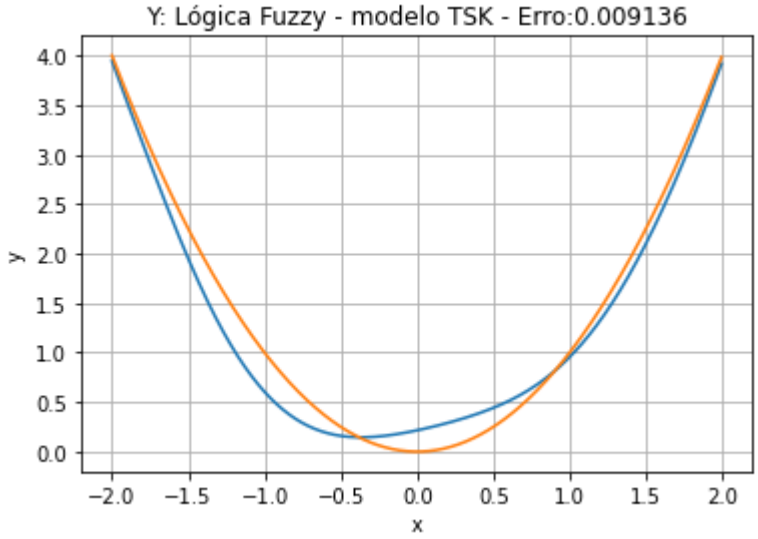
epoca: 94



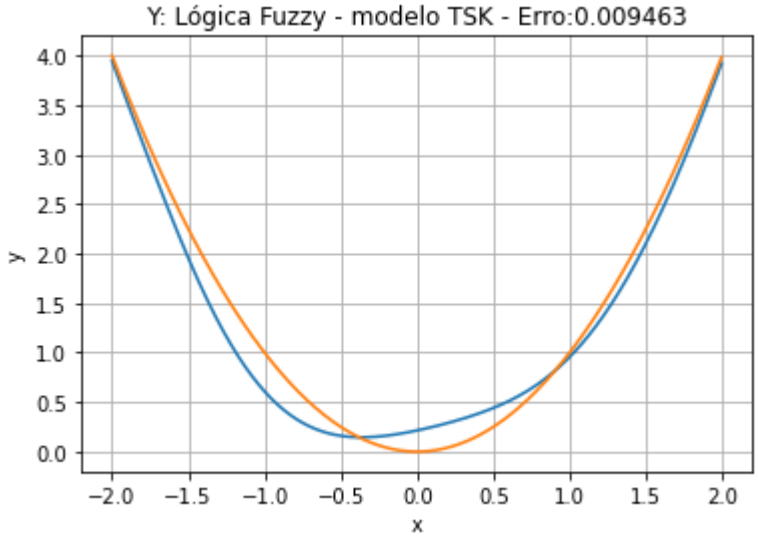
epoca: 95



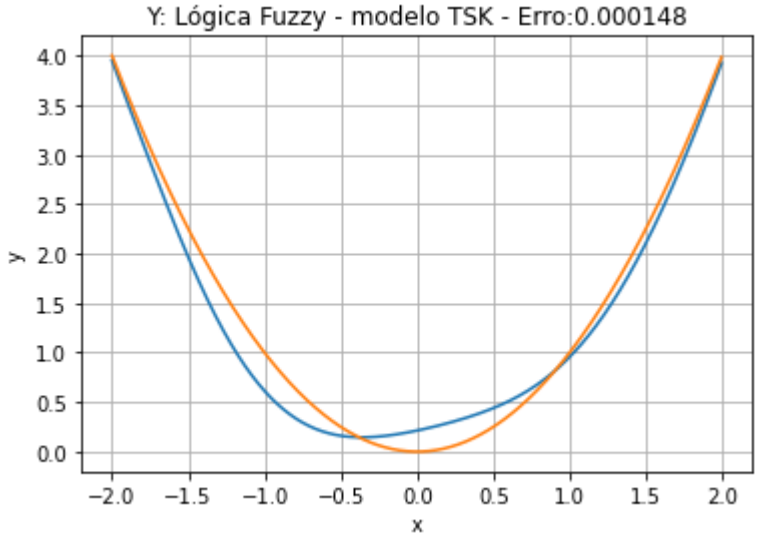
epoca: 96



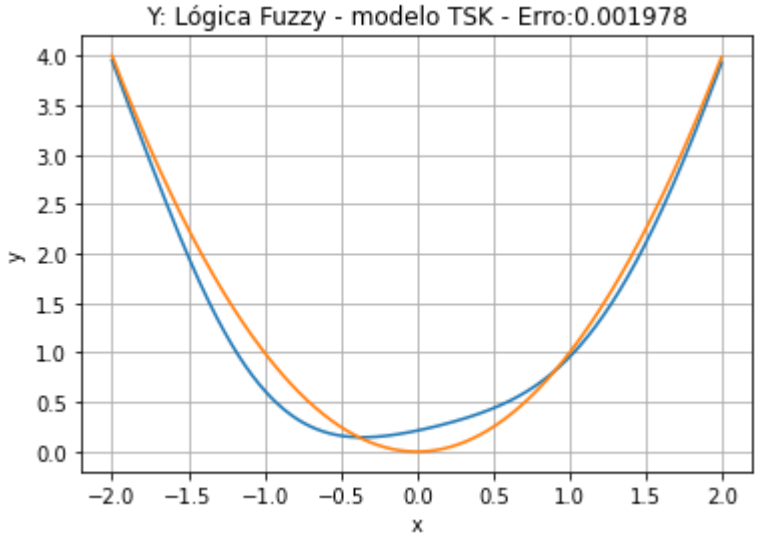
epoca: 97



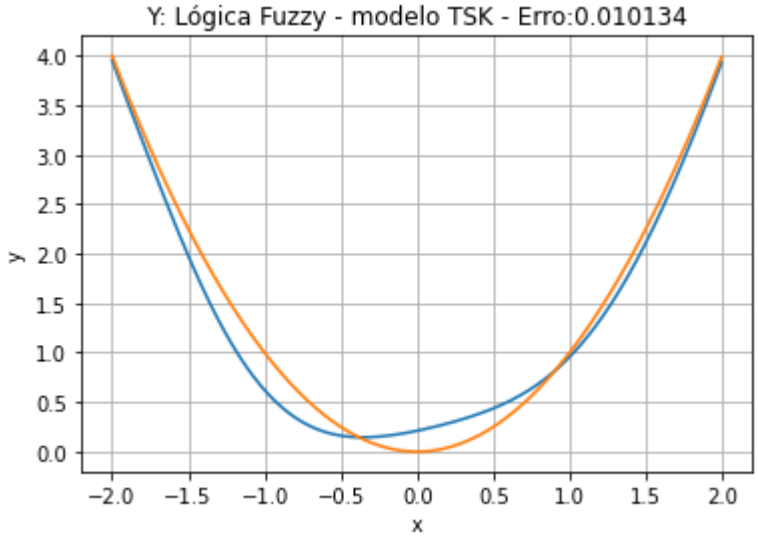
epoca: 98



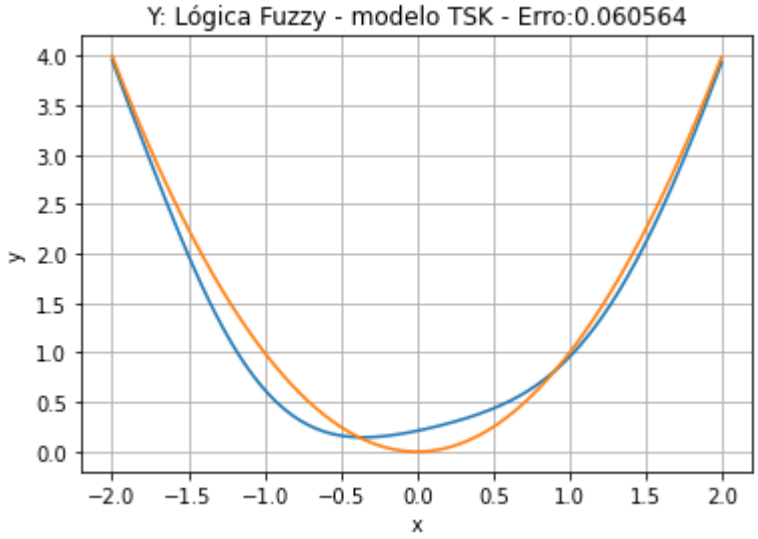
epoca: 99



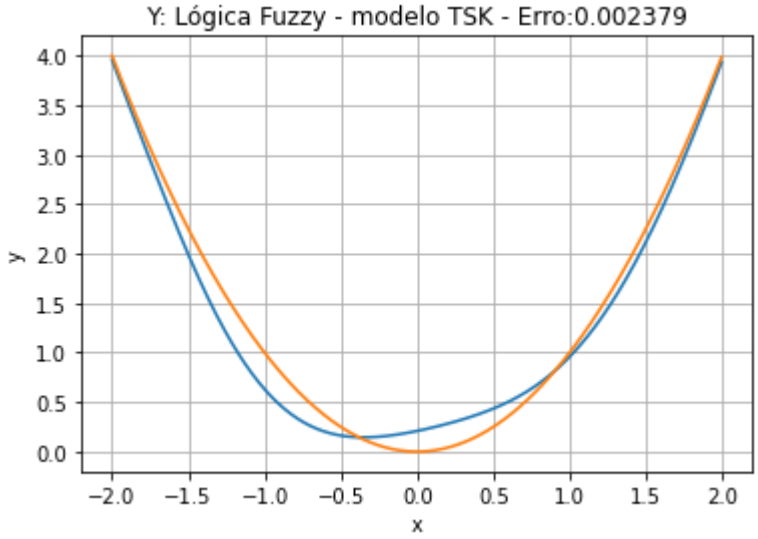
epoca: 100



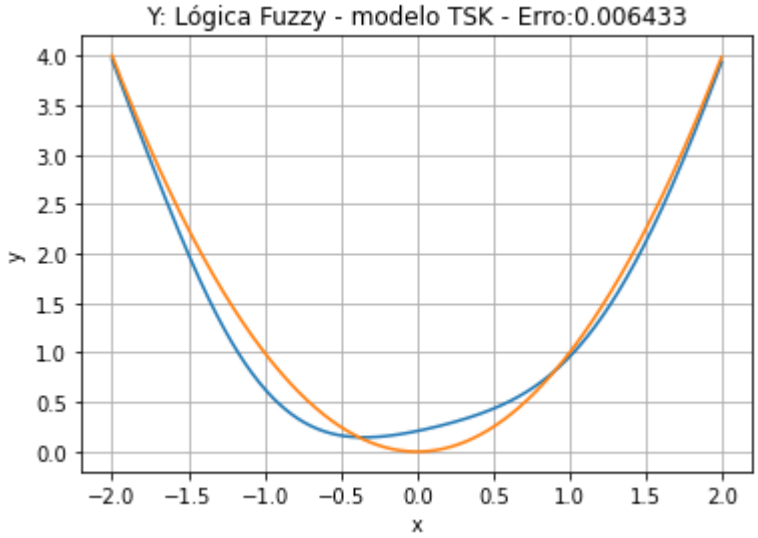
epoca: 101



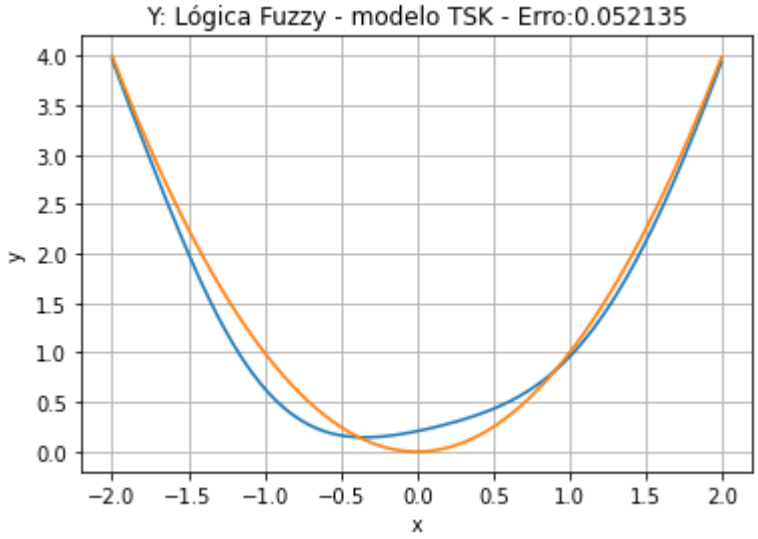
epoca: 102



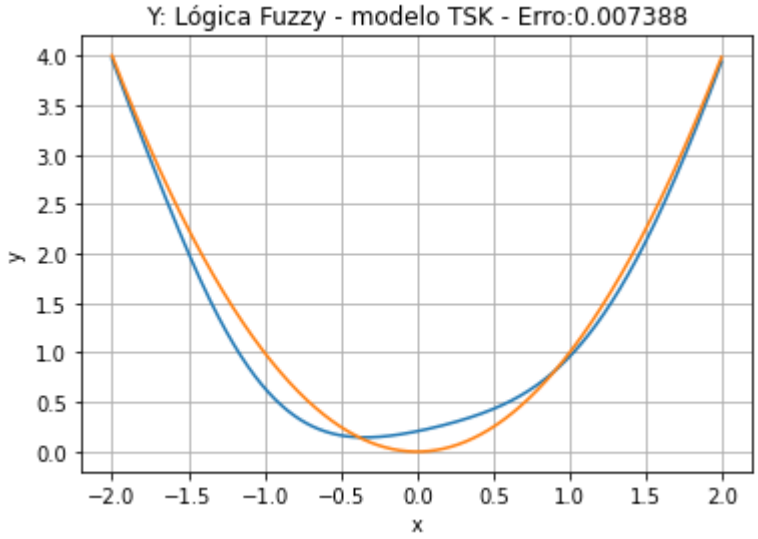
epoca: 103



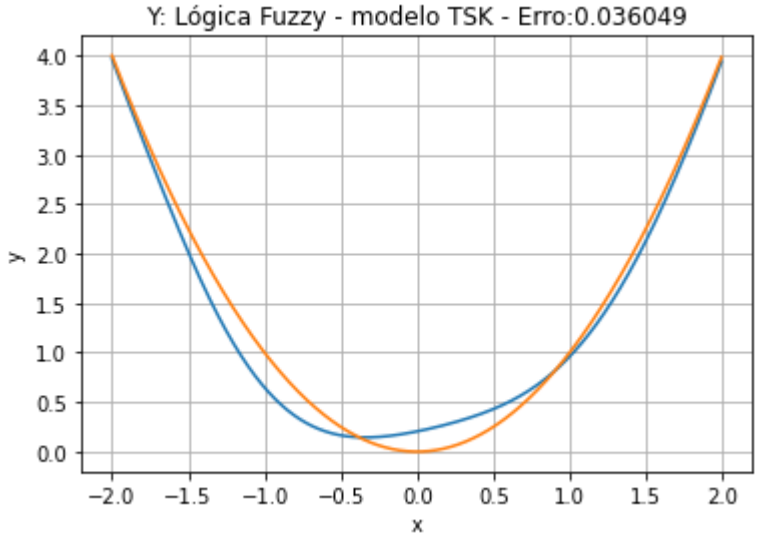
epoca: 104



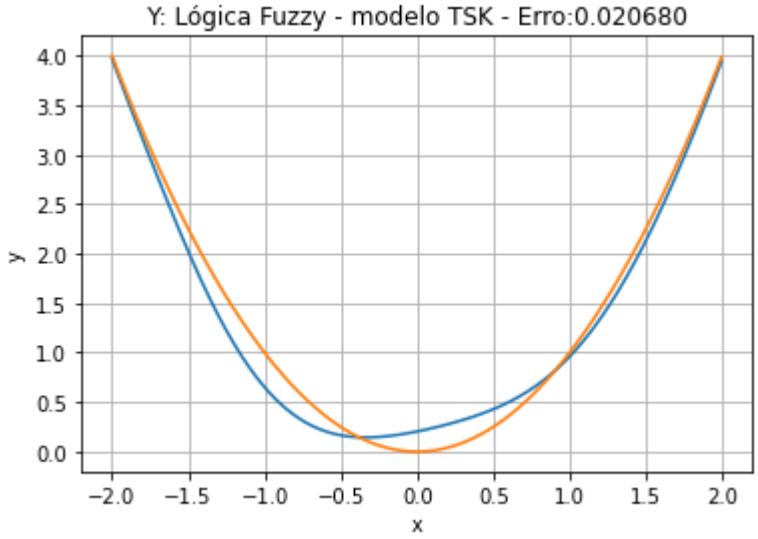
epoca: 105



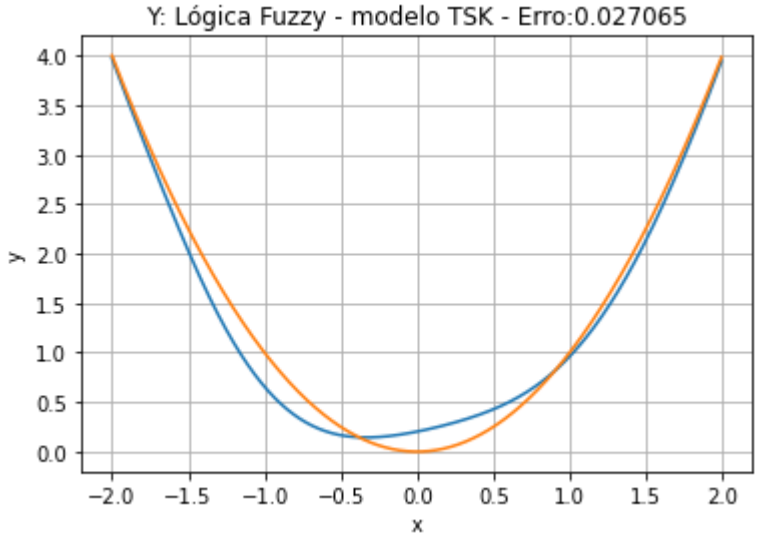
epoca: 106



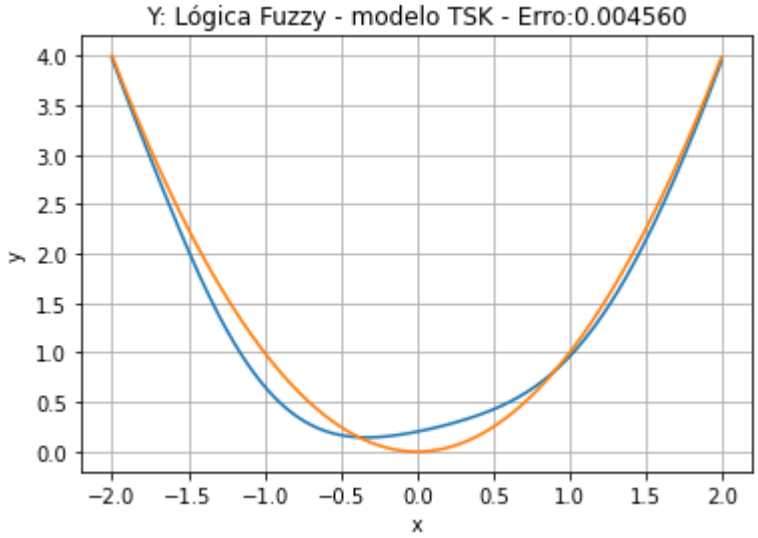
epoca: 107



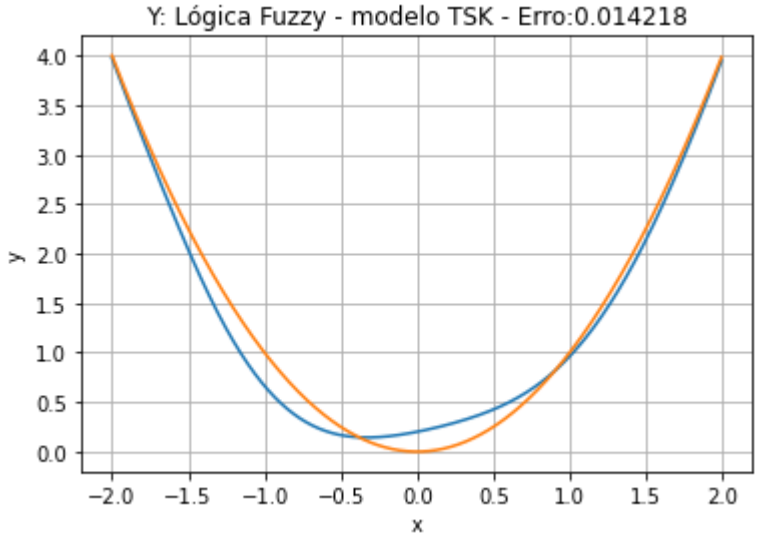
epoca: 108



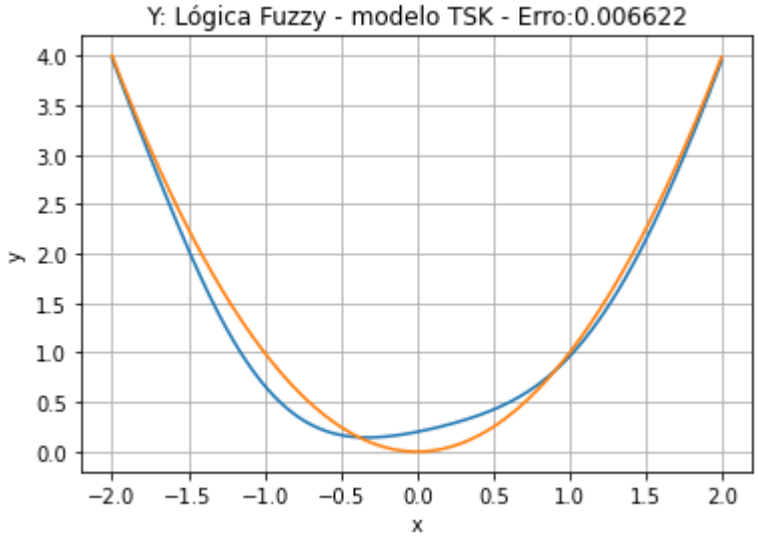
epoca: 109



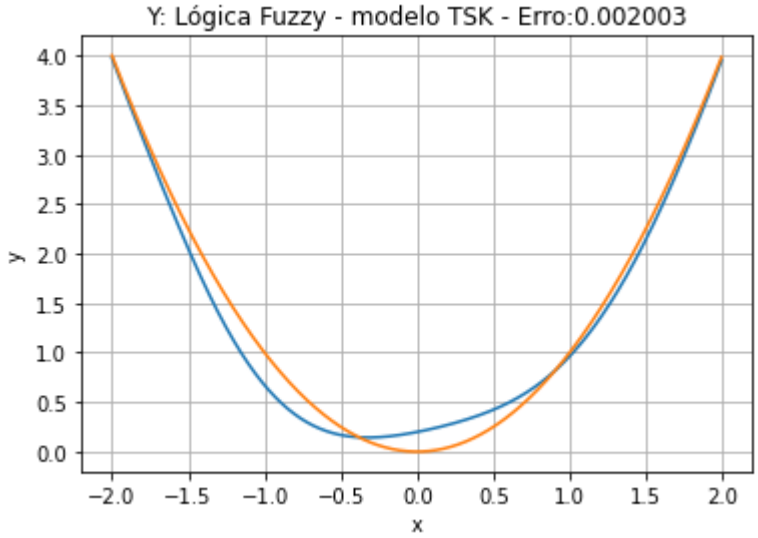
epoca: 110



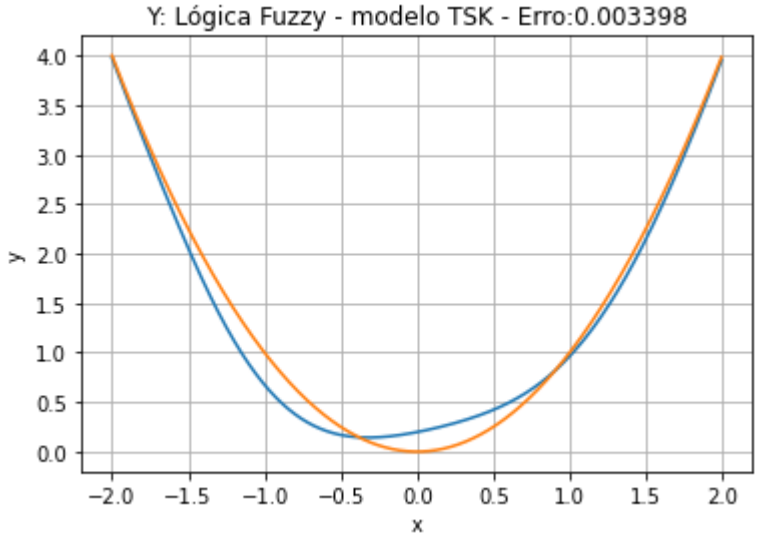
epoca: 111



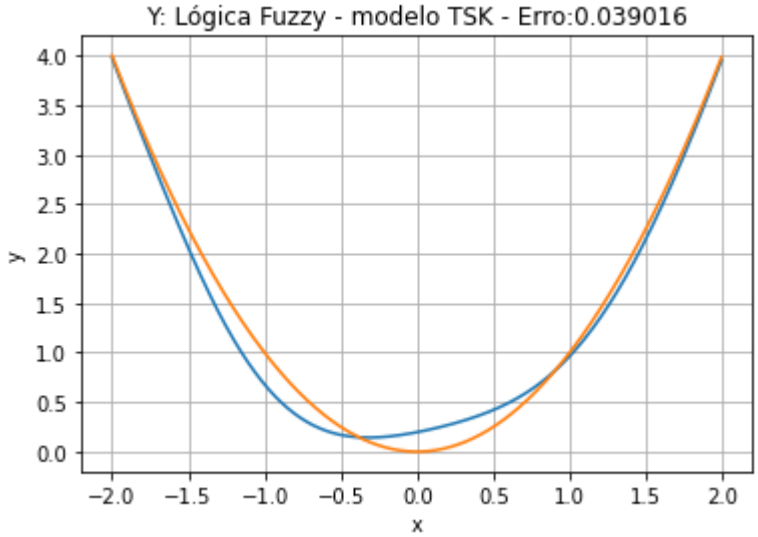
epoca: 112



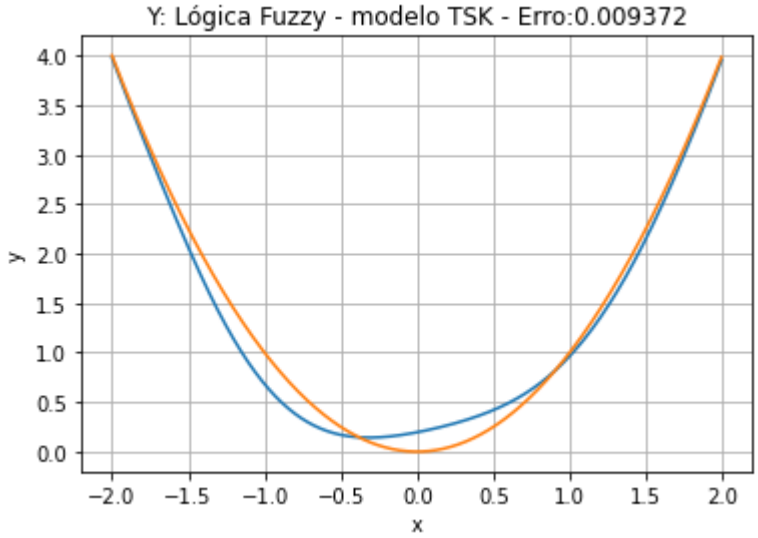
epoca: 113



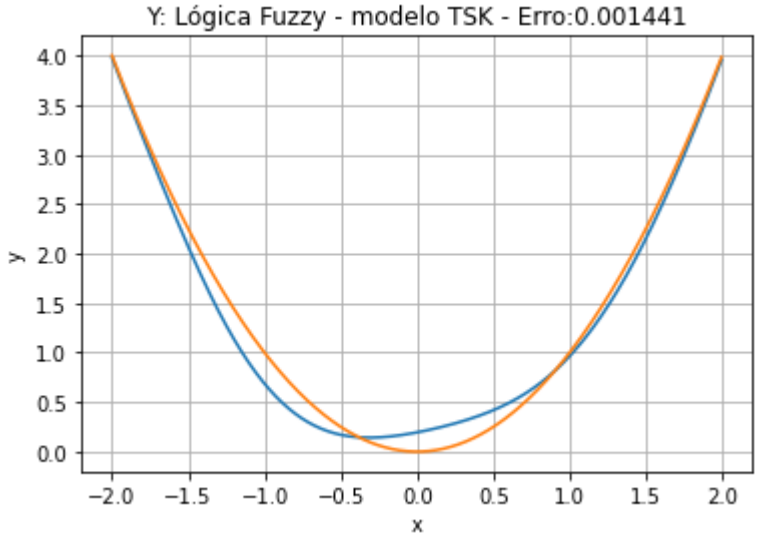
epoca: 114



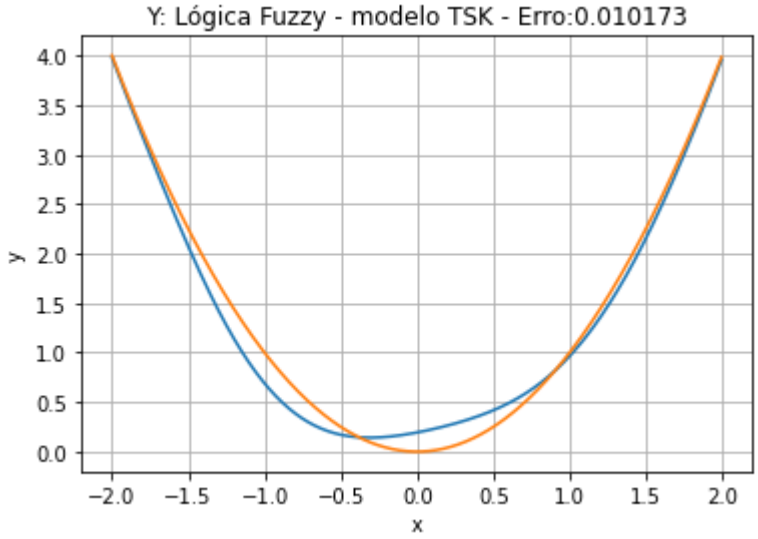
epoca: 115



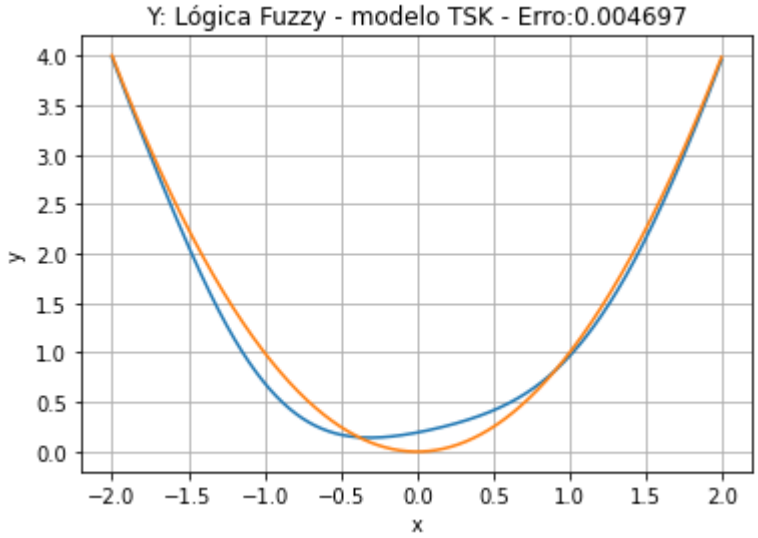
epoca: 116



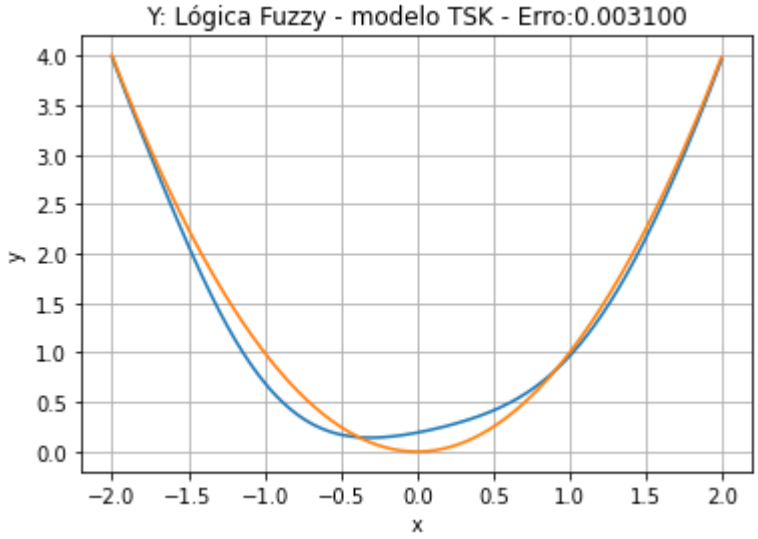
epoca: 117



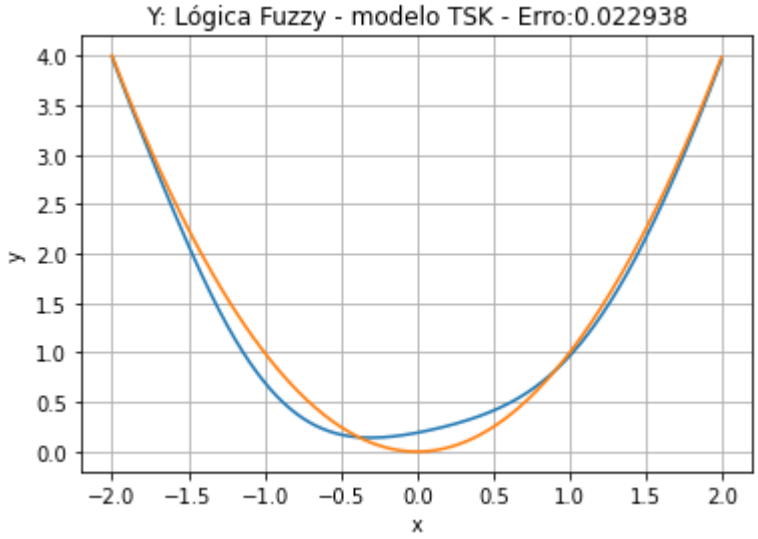
epoca: 118



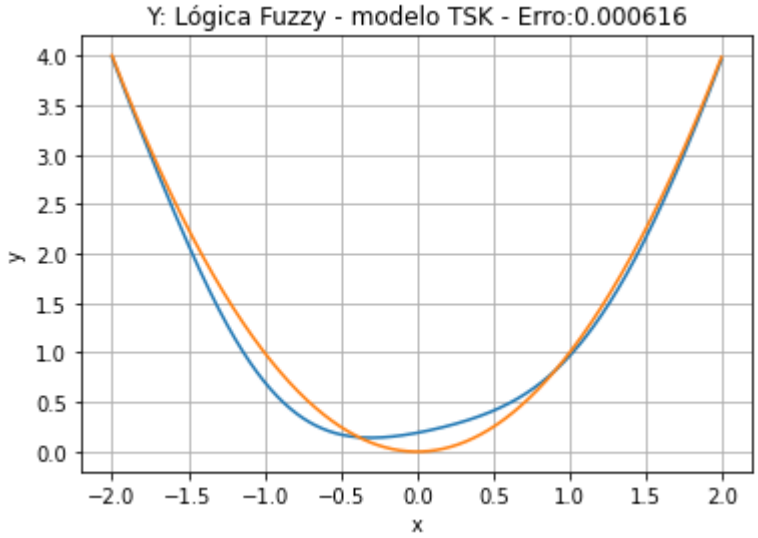
epoca: 119



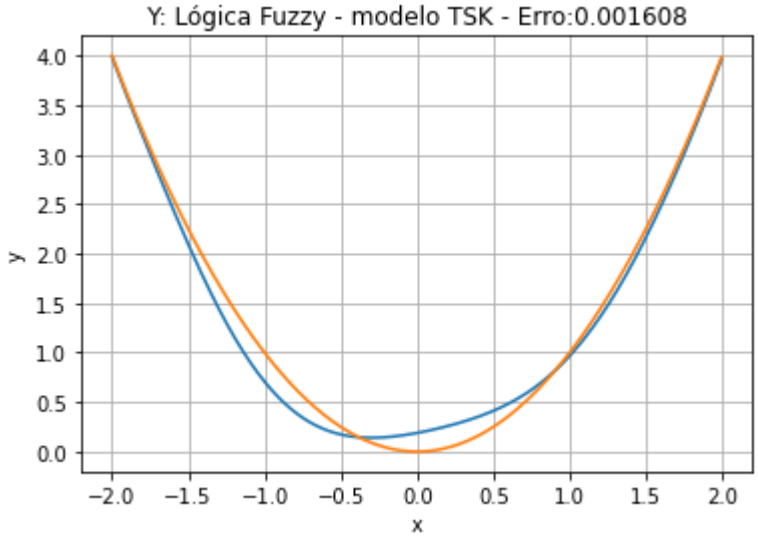
epoca: 120



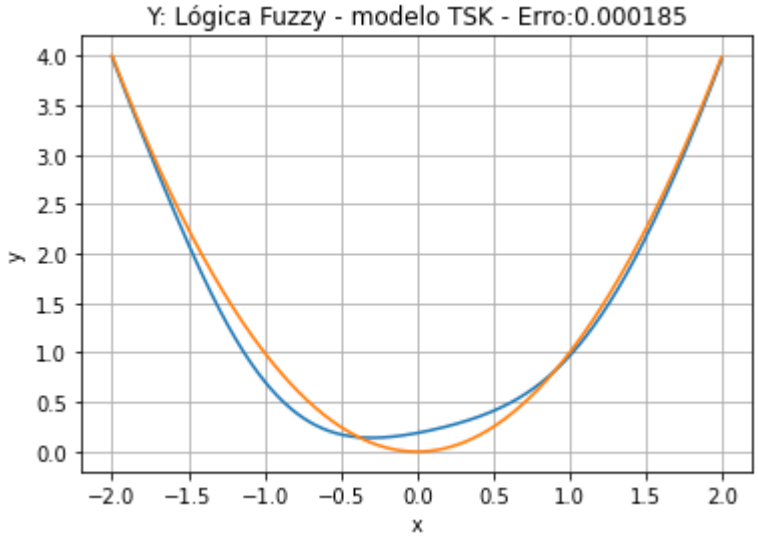
epoca: 121



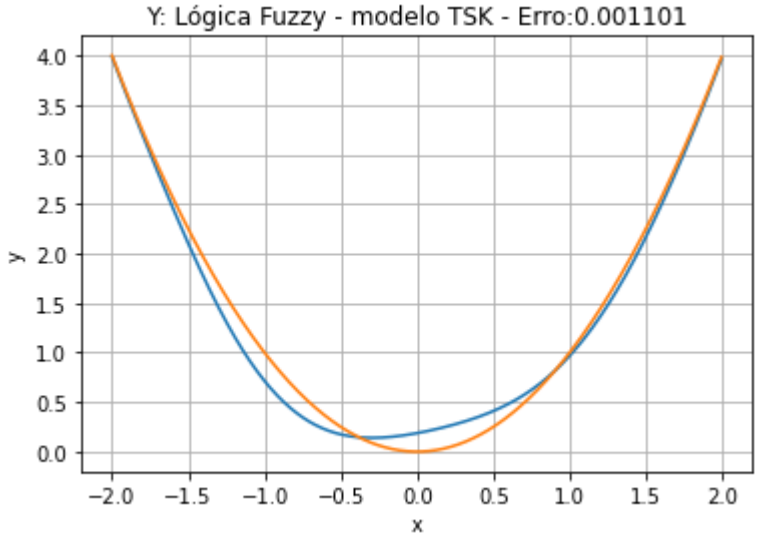
epoca: 122



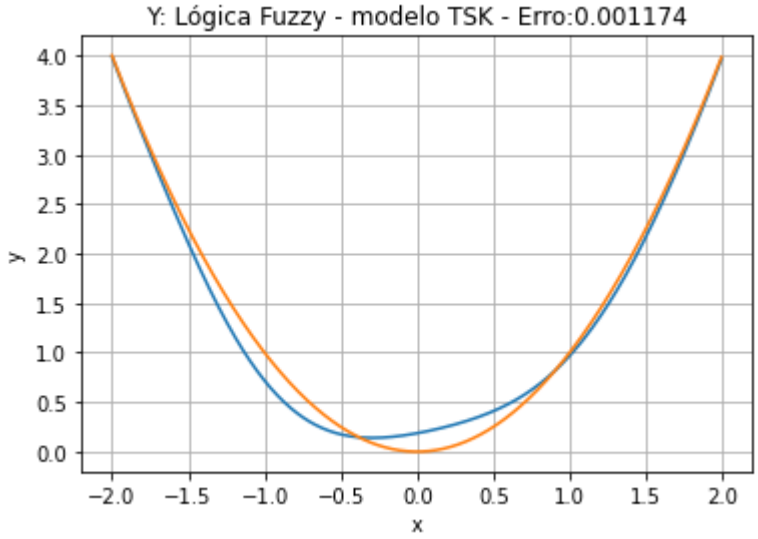
epoca: 123



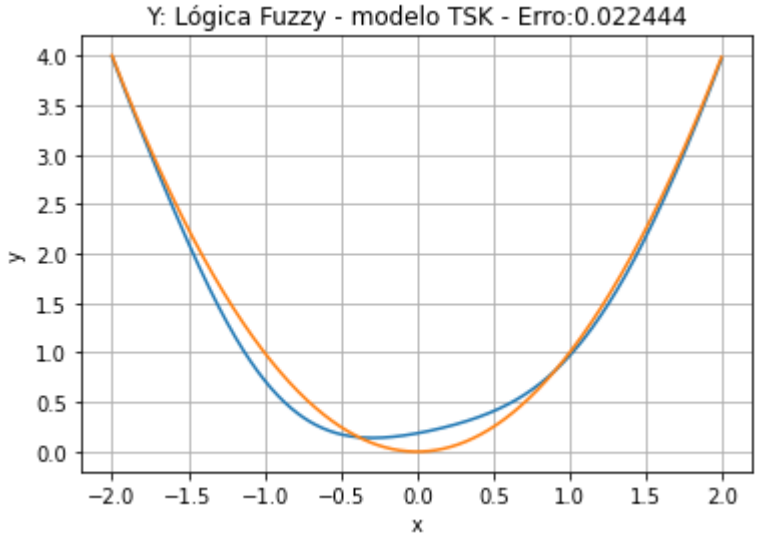
epoca: 124



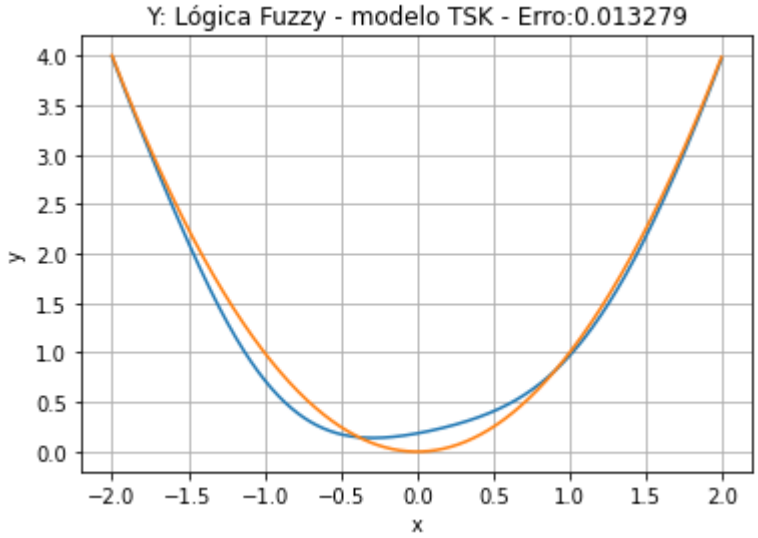
epoca: 125



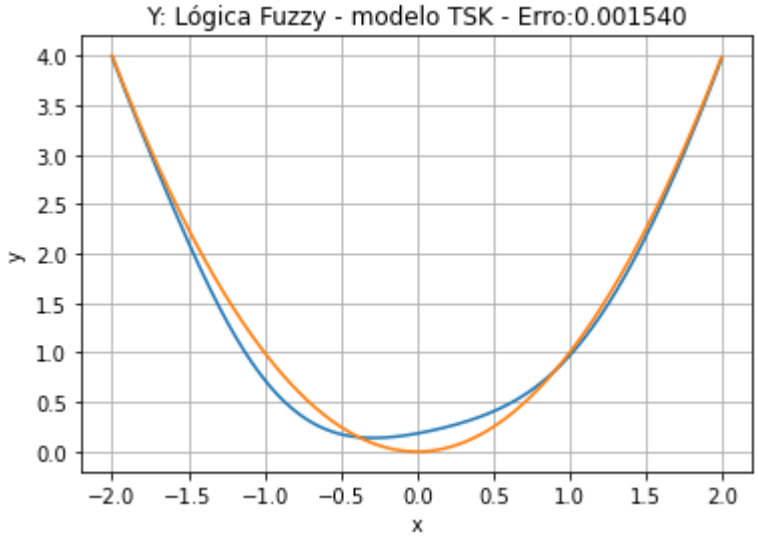
epoca: 126



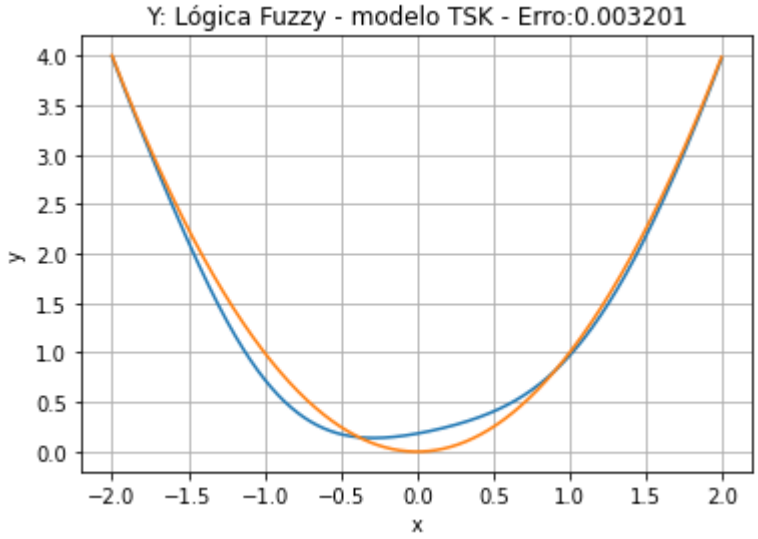
epoca: 127



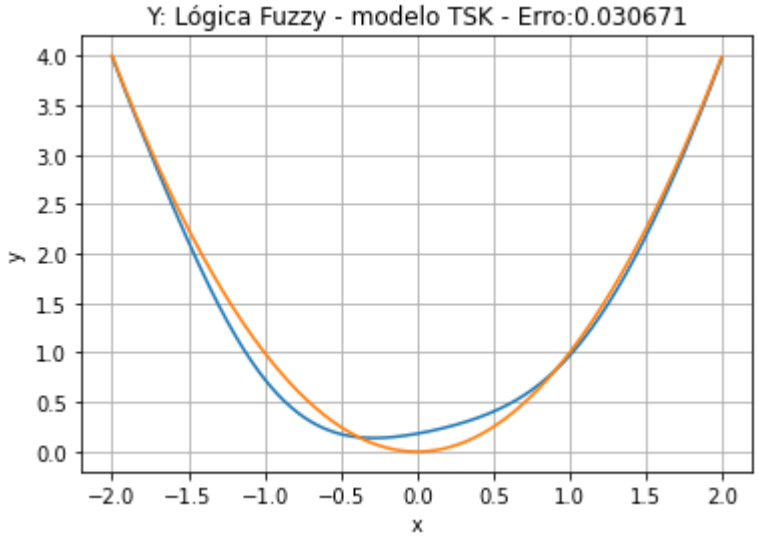
epoca: 128



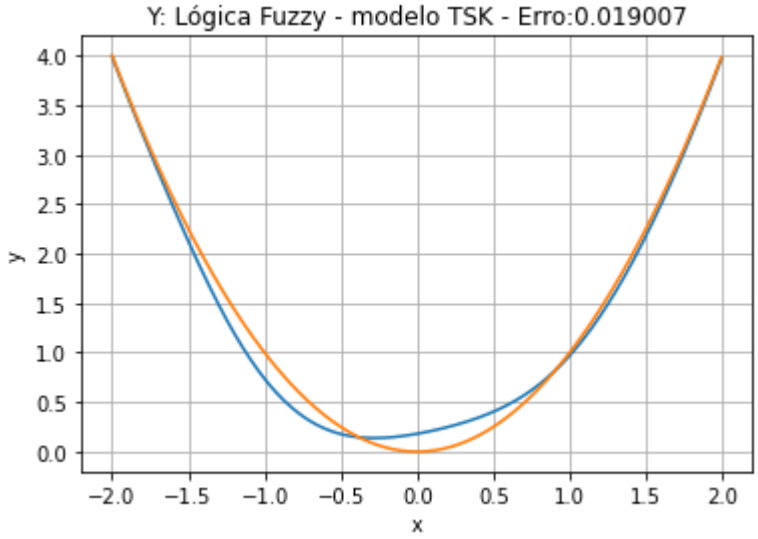
epoca: 129



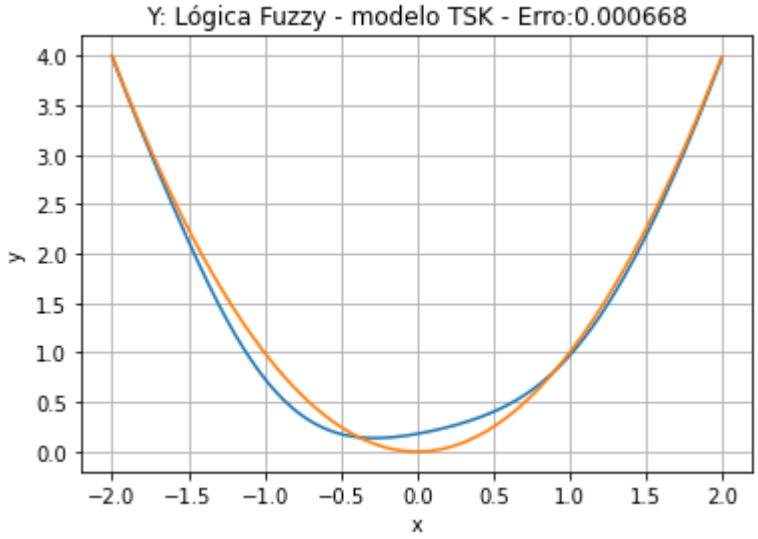
epoca: 130



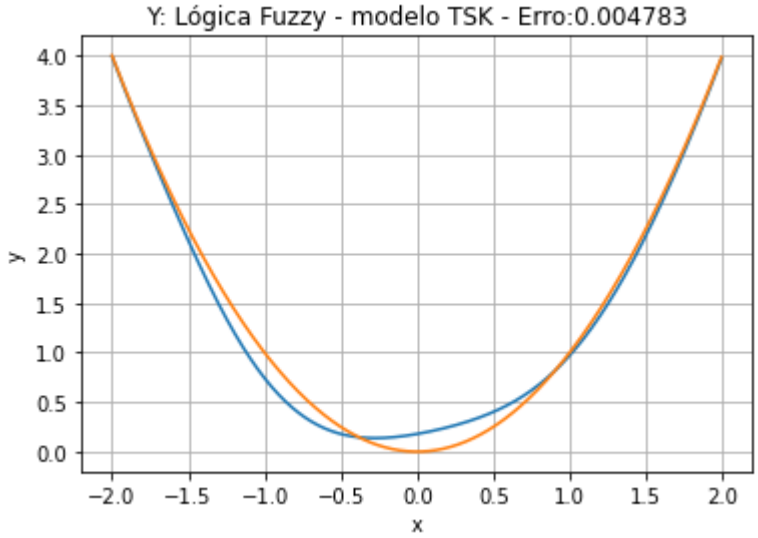
epoca: 131



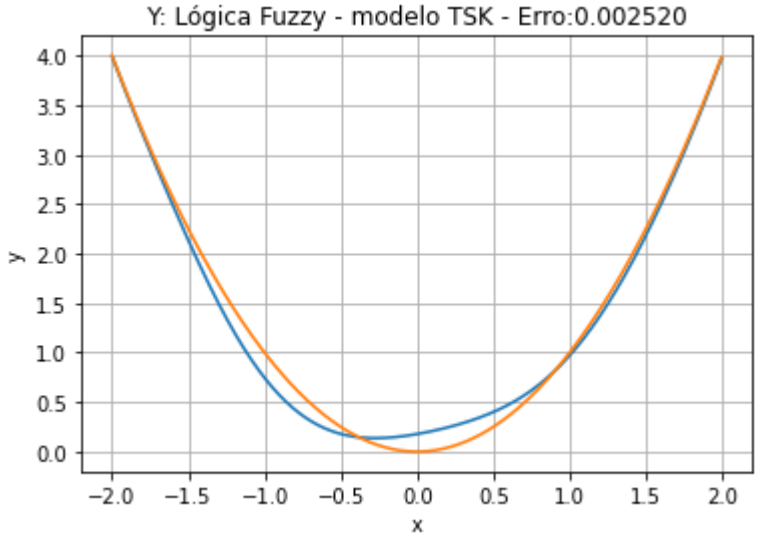
epoca: 132



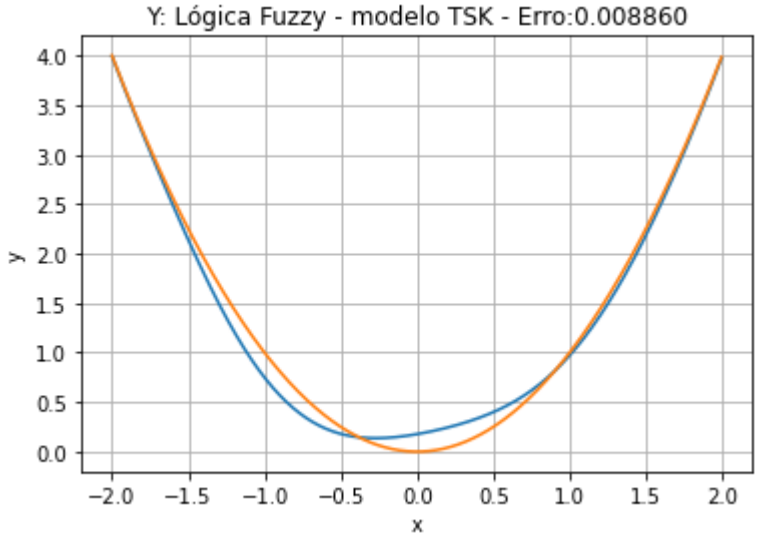
epoca: 133



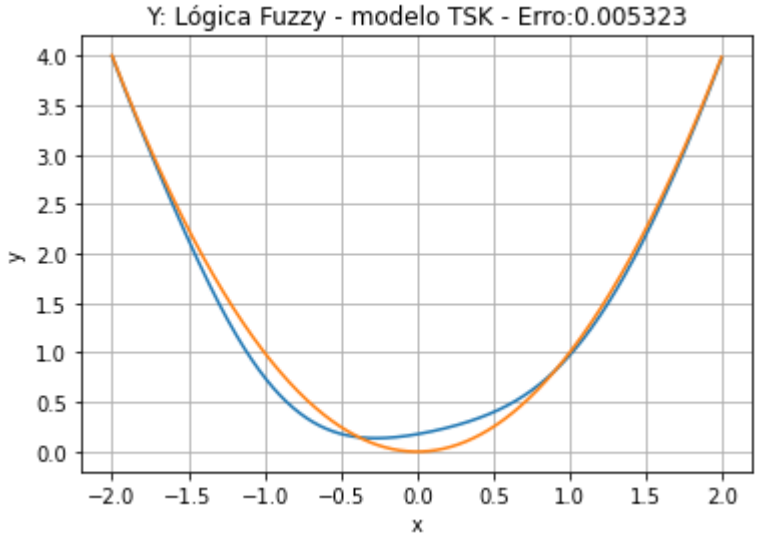
epoca: 134



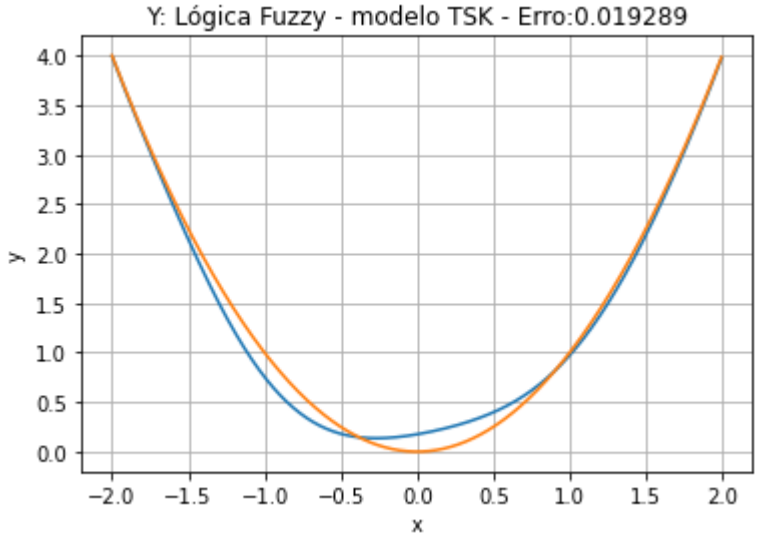
epoca: 135



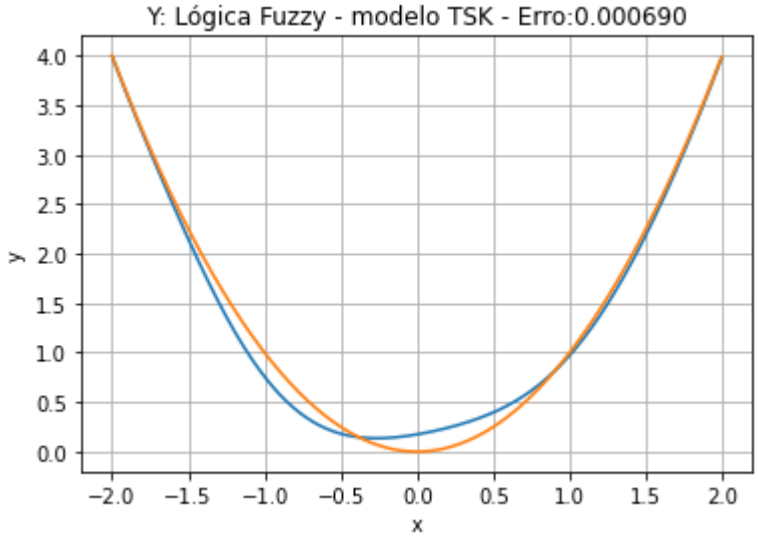
epoca: 136



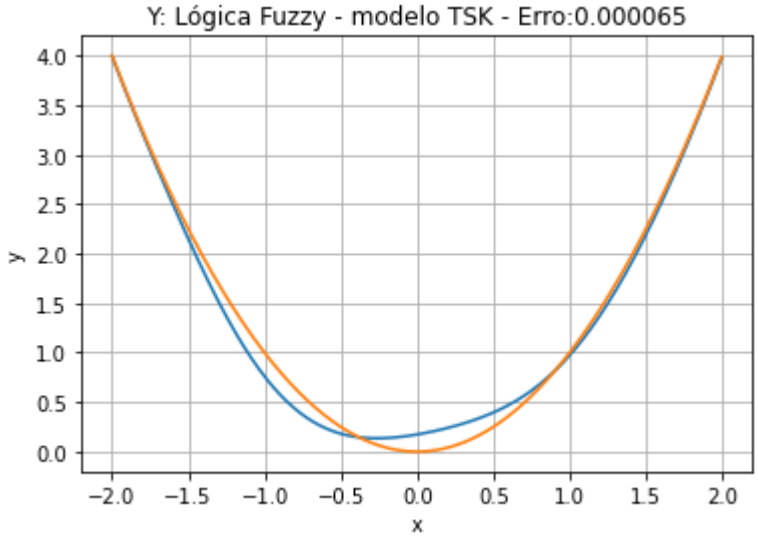
epoca: 137



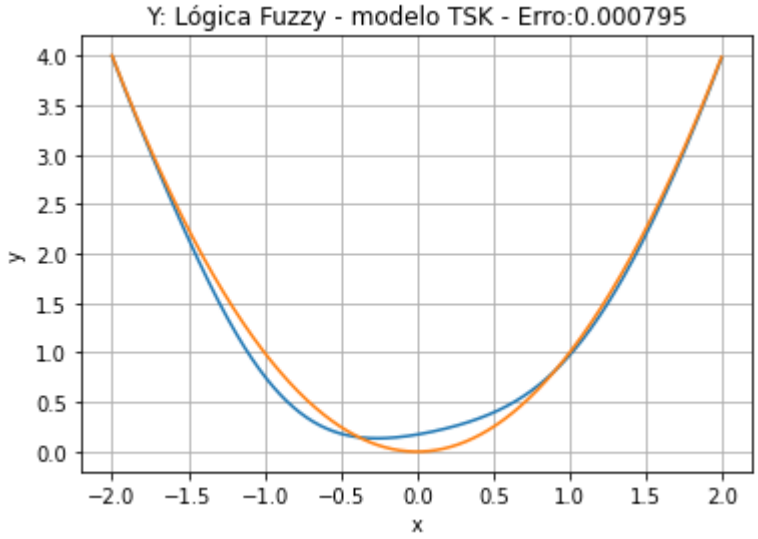
epoca: 138



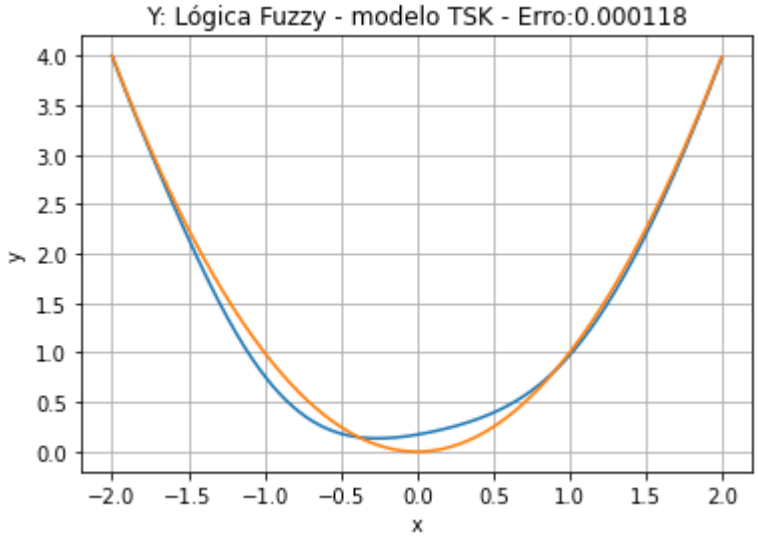
epoca: 139



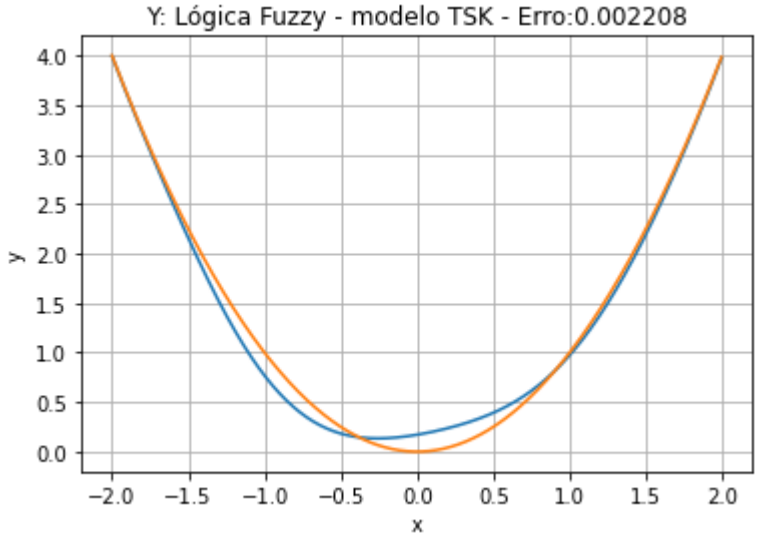
epoca: 140



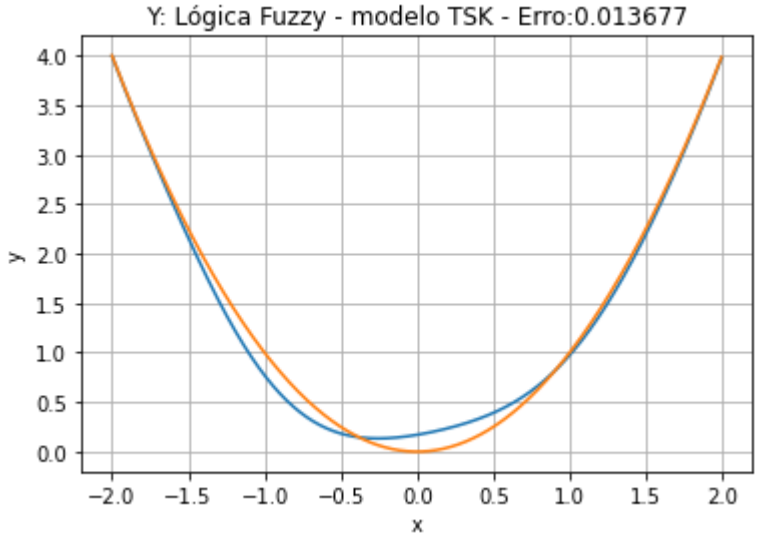
epoca: 141



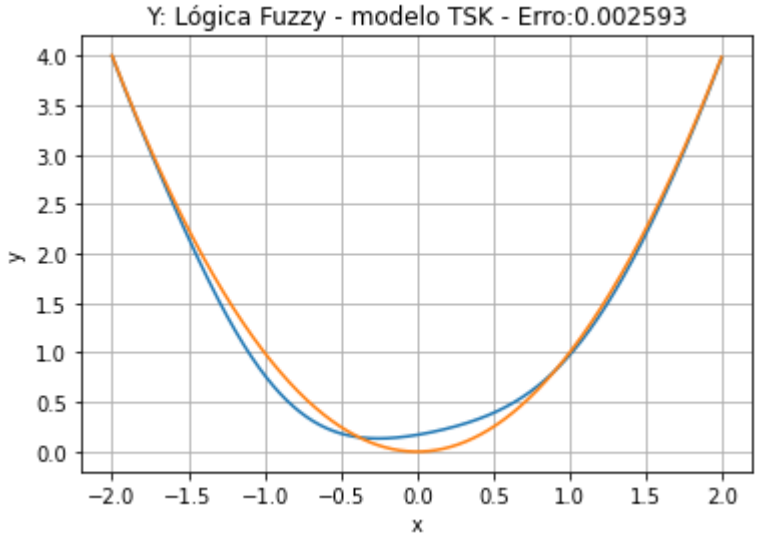
epoca: 142



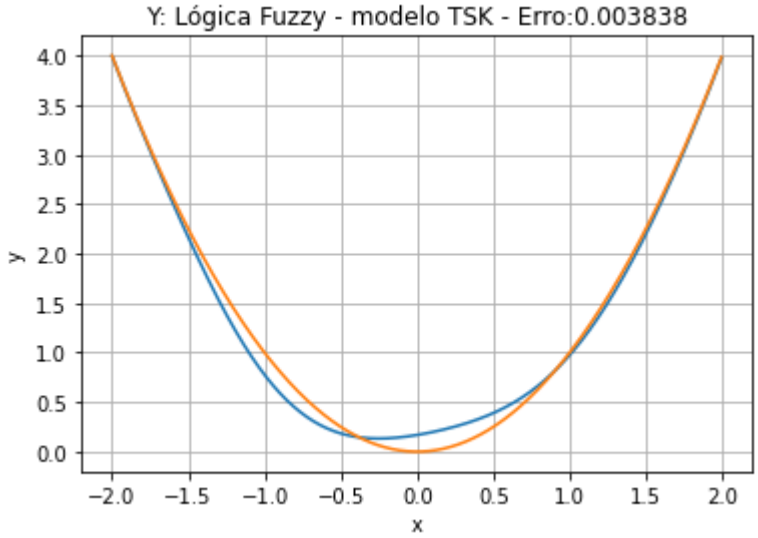
epoca: 143



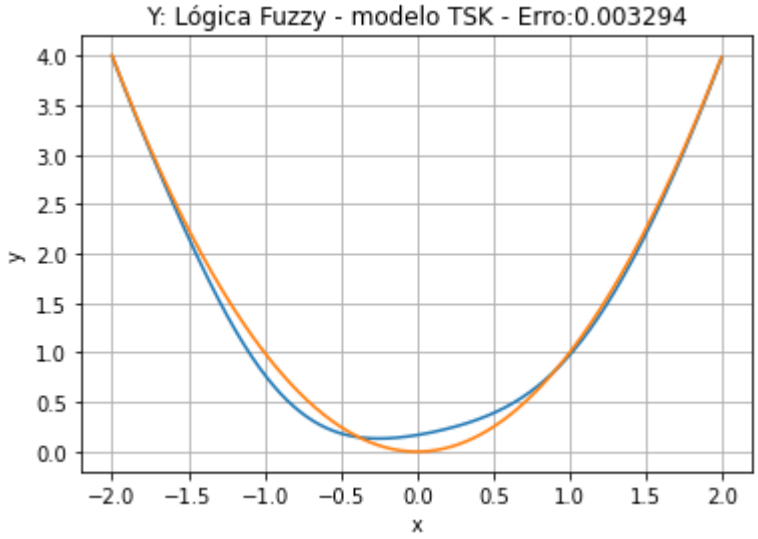
epoca: 144



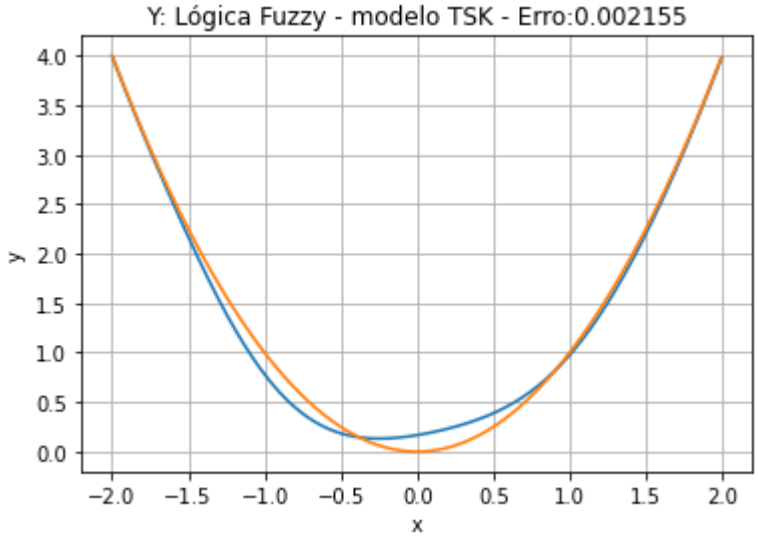
epoca: 145



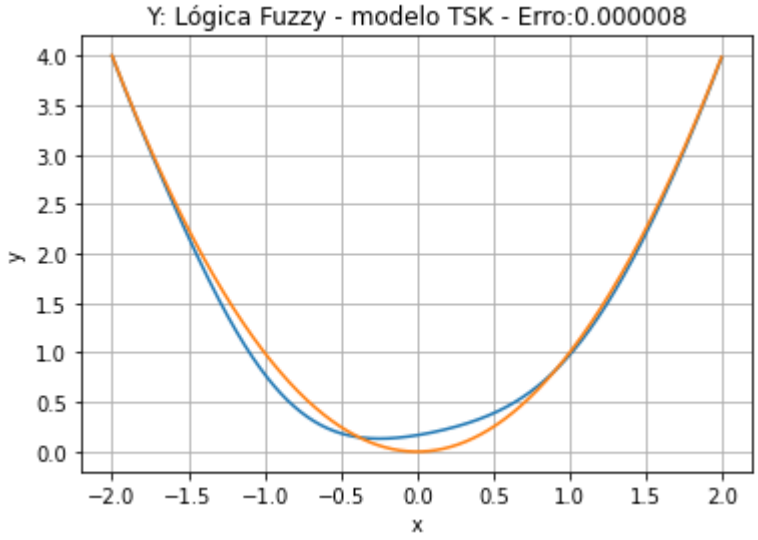
epoca: 146



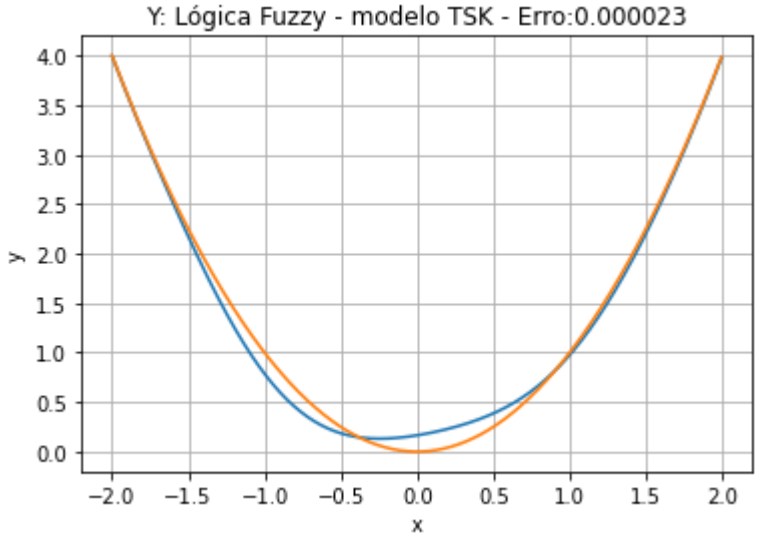
epoca: 147



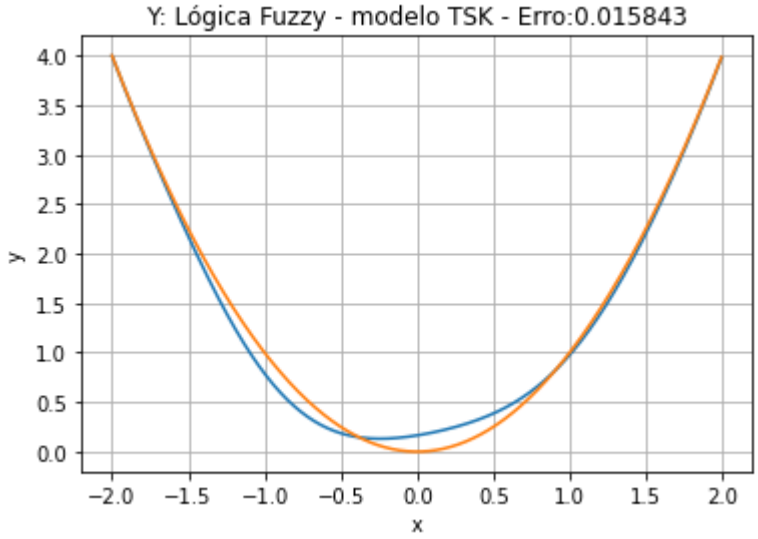
epoca: 148



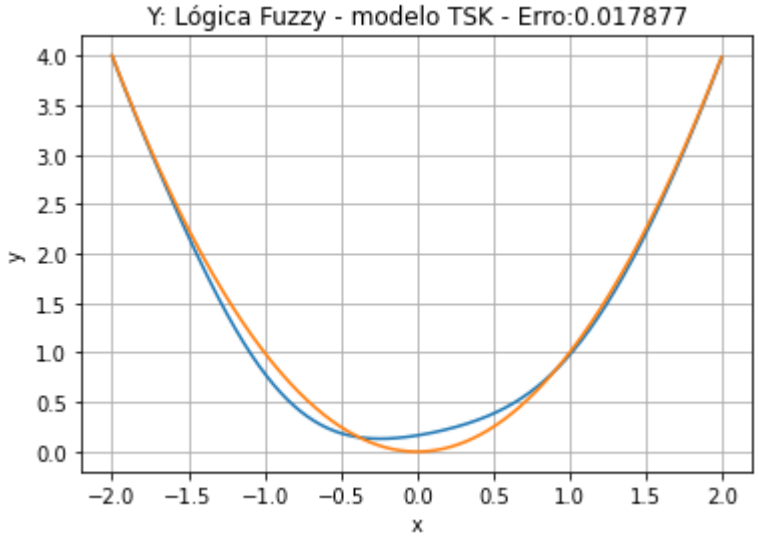
epoca: 149



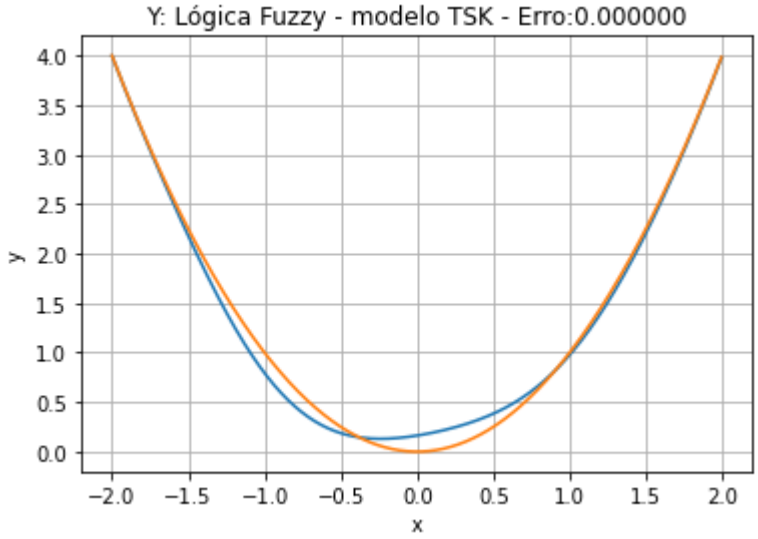
epoca: 150



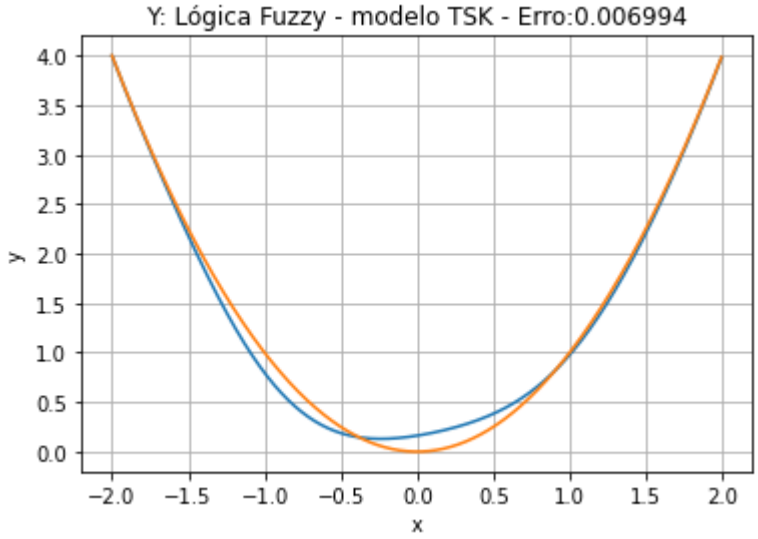
epoca: 151



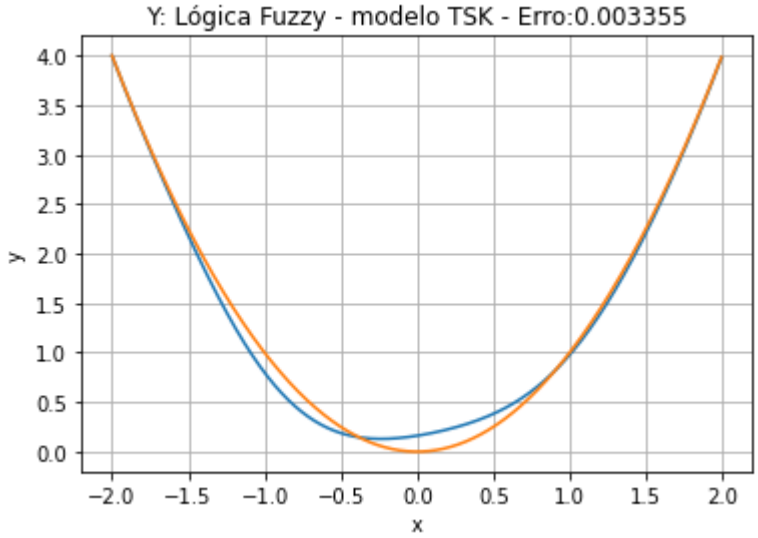
epoca: 152



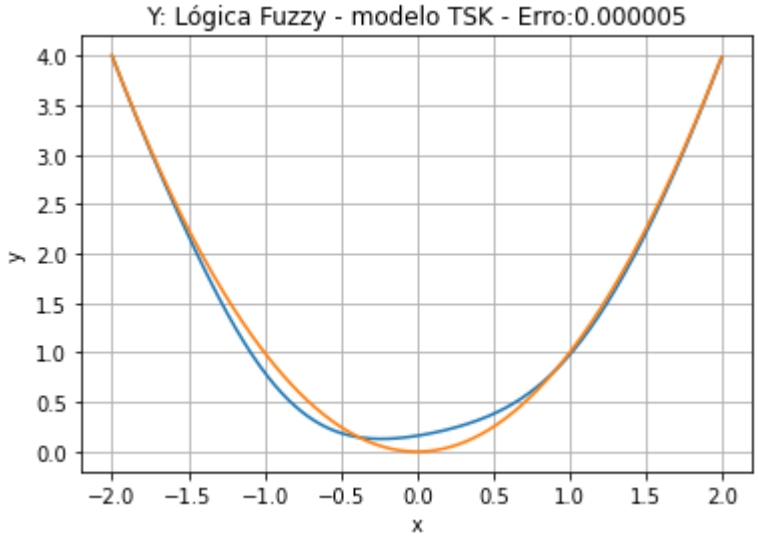
epoca: 153



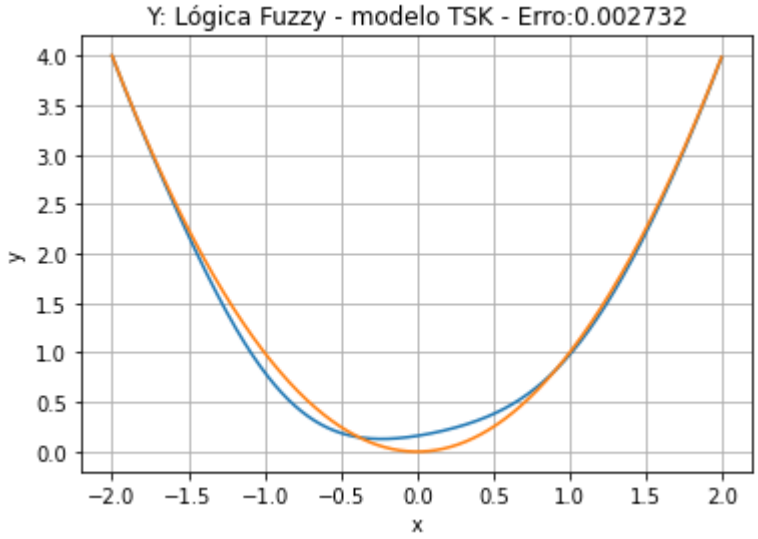
epoca: 154



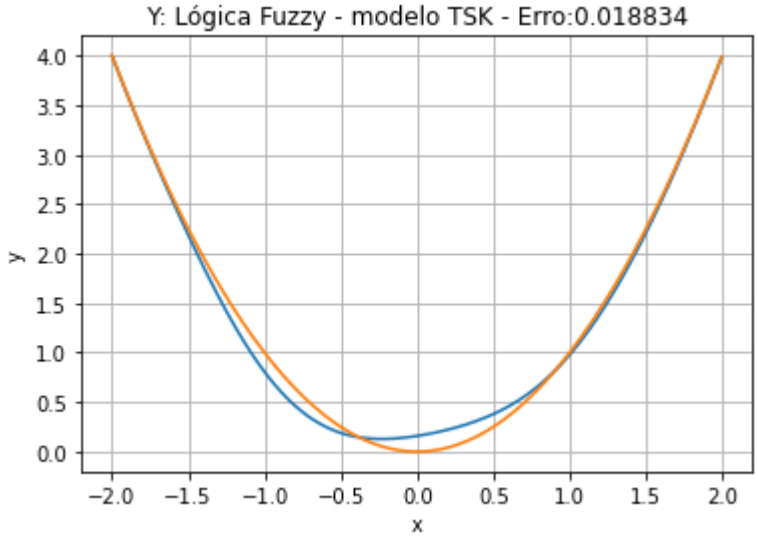
epoca: 155



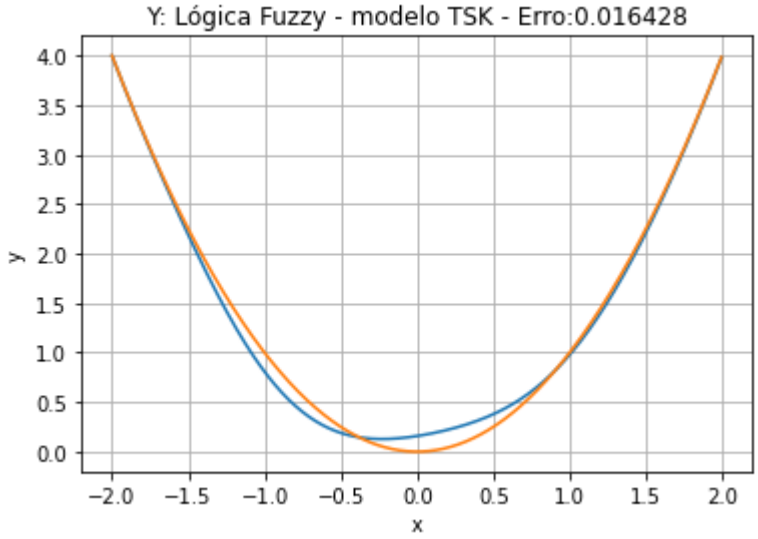
epoca: 156



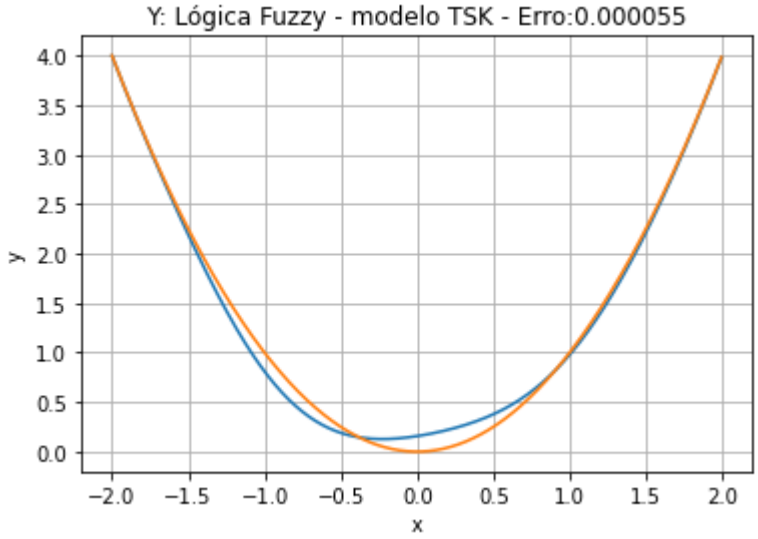
epoca: 157



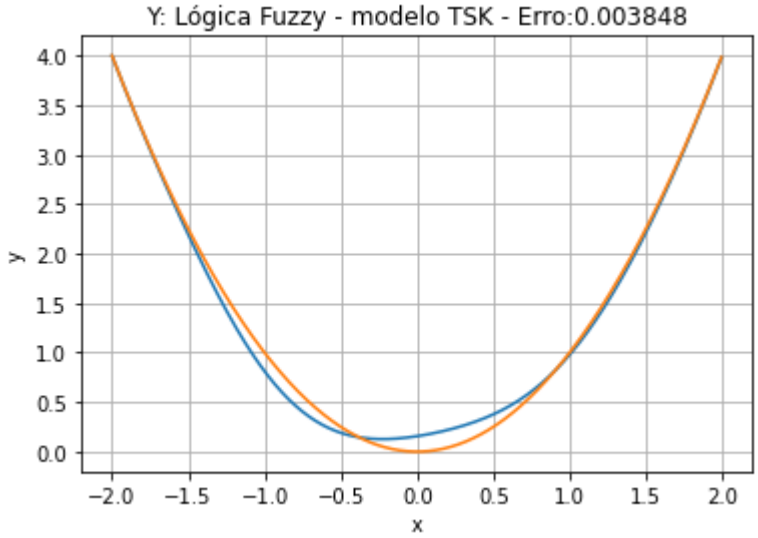
epoca: 158



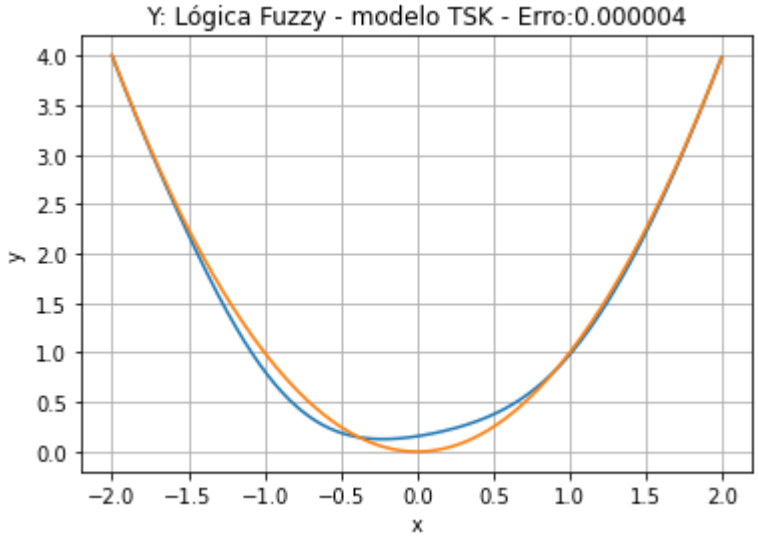
epoca: 159



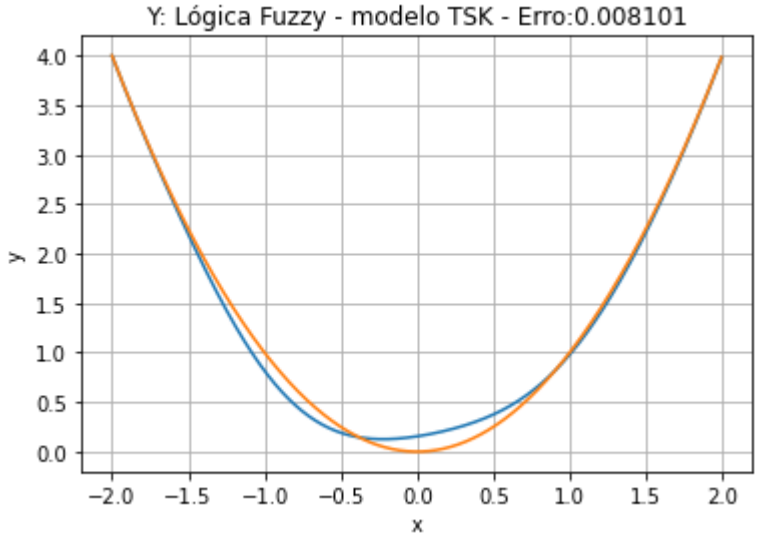
epoca: 160



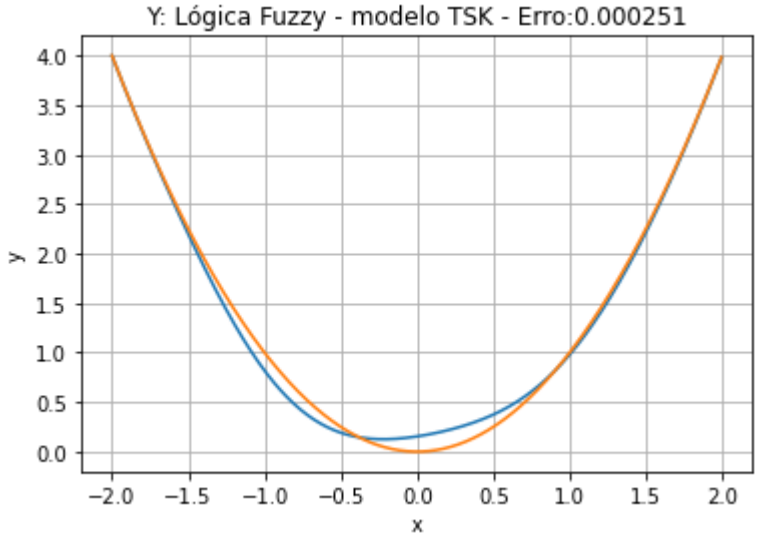
epoca: 161



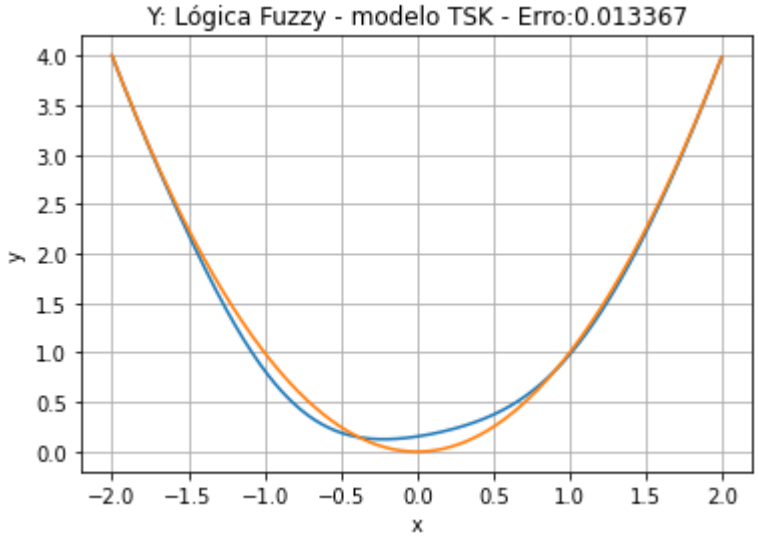
epoca: 162



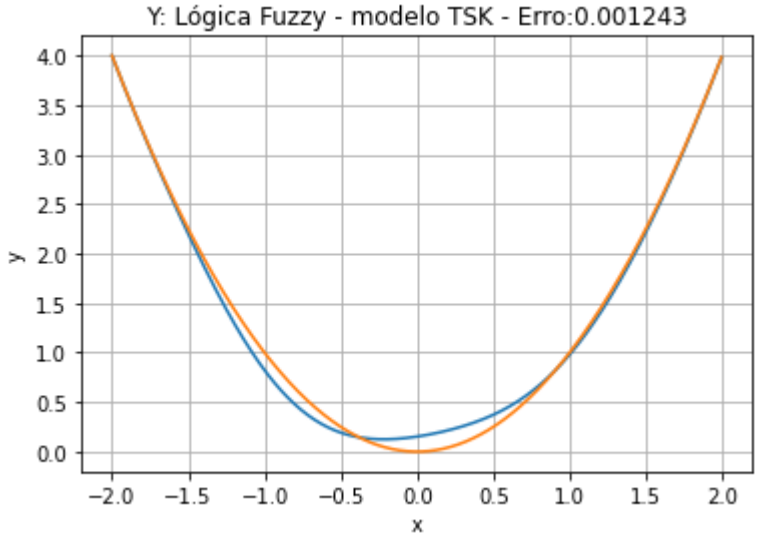
epoca: 163



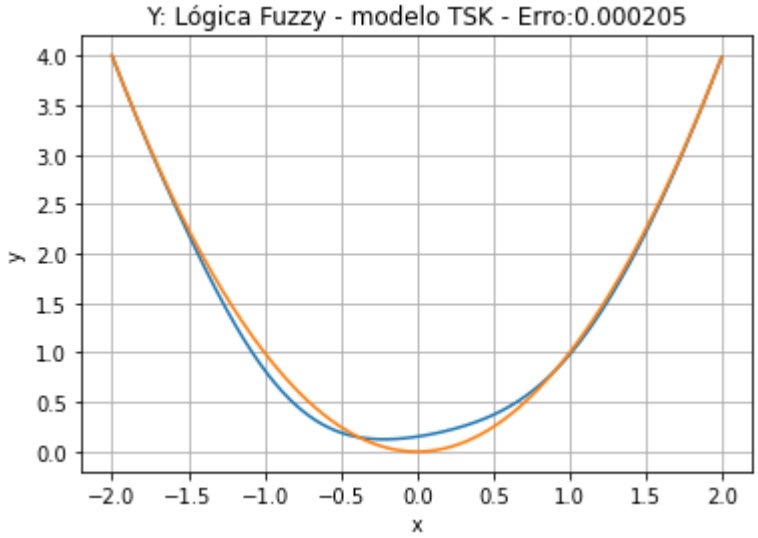
epoca: 164



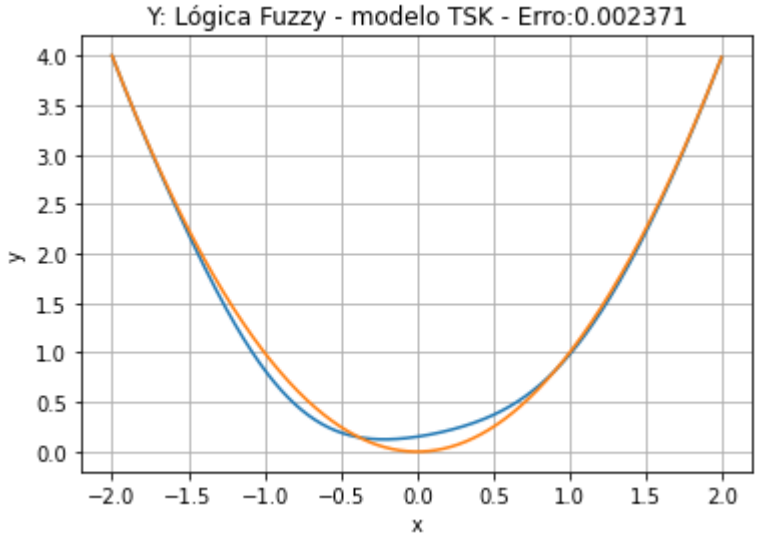
epoca: 165



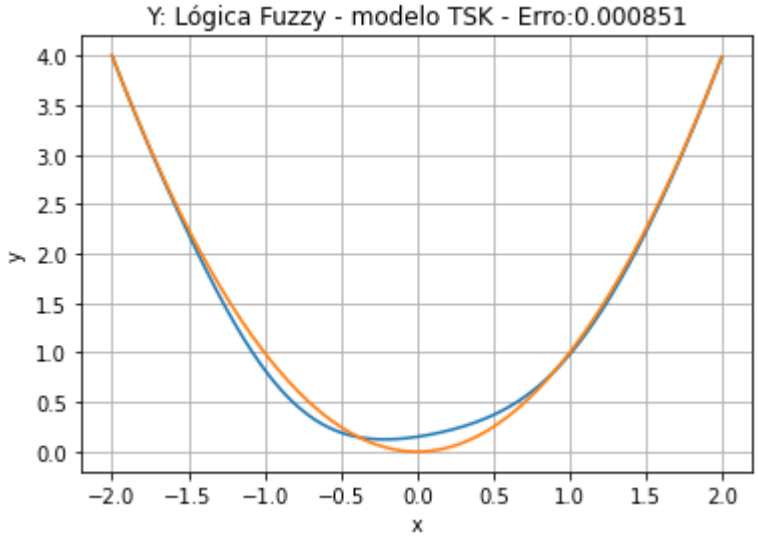
epoca: 166



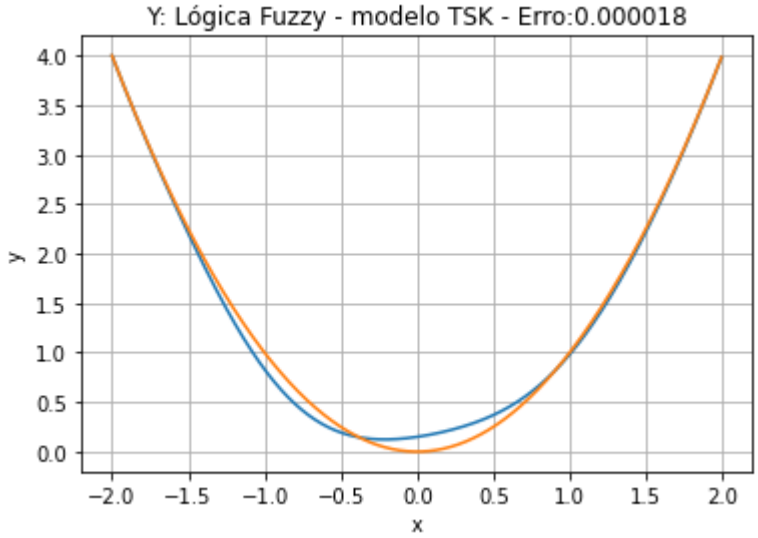
epoca: 167



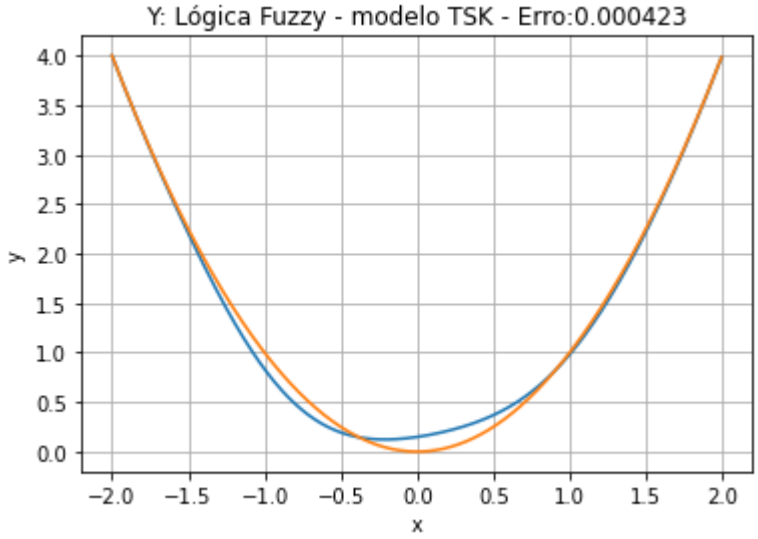
epoca: 168



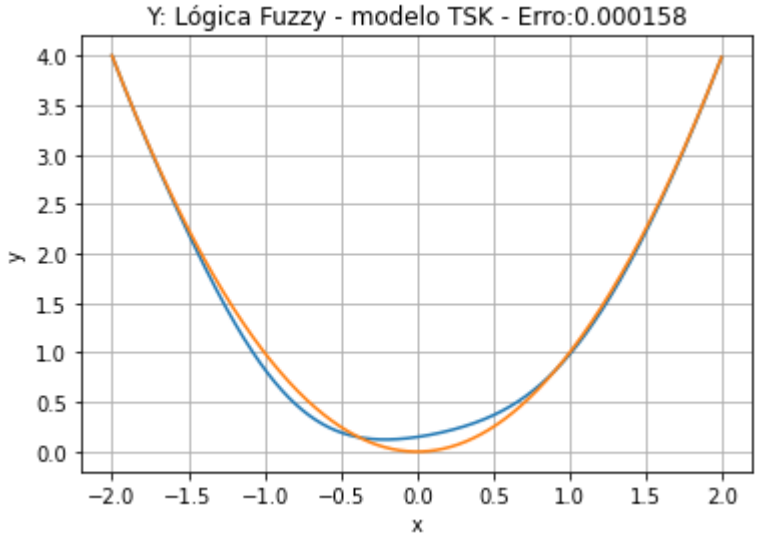
epoca: 169



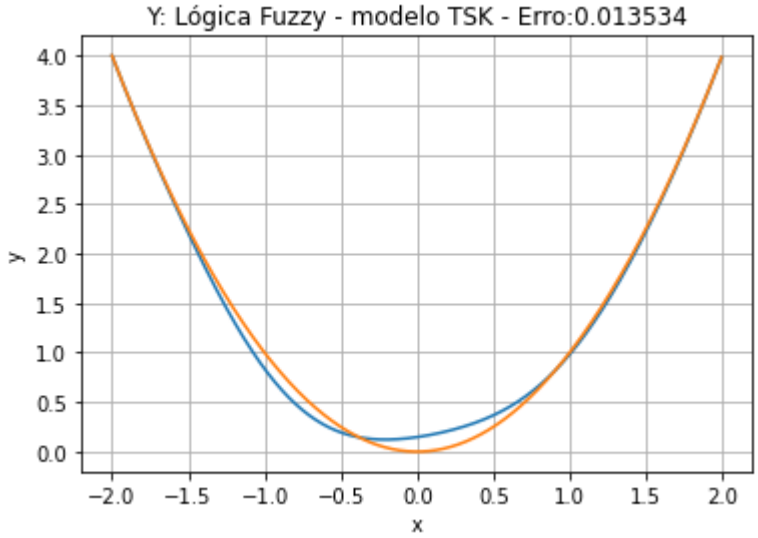
epoca: 170



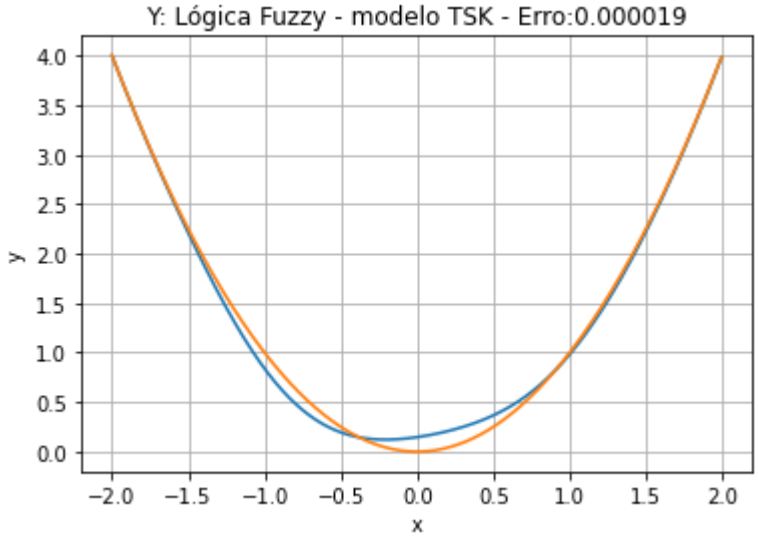
epoca: 171



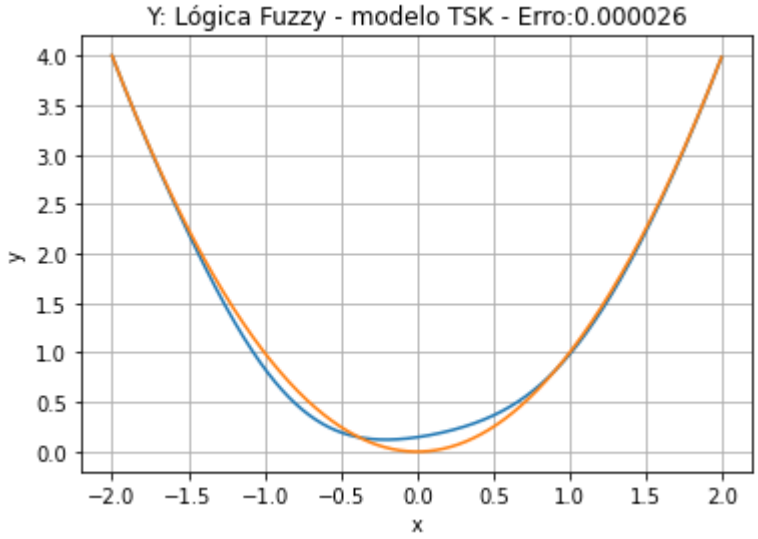
epoca: 172



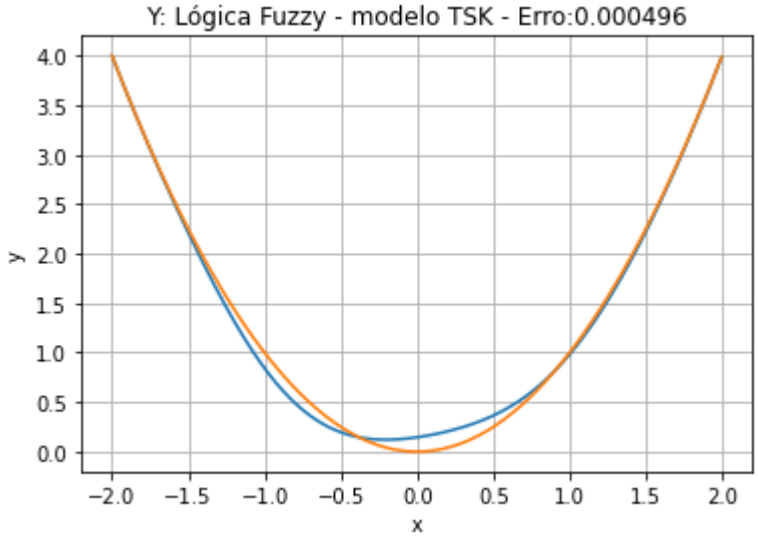
epoca: 173



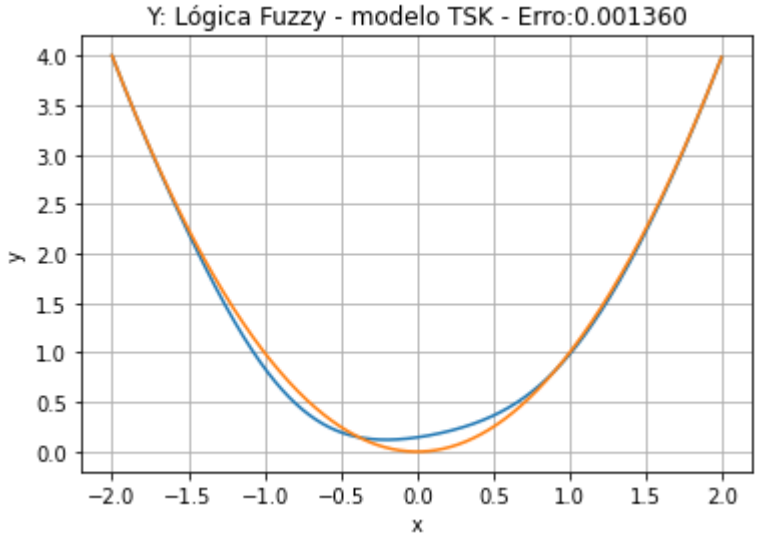
epoca: 174



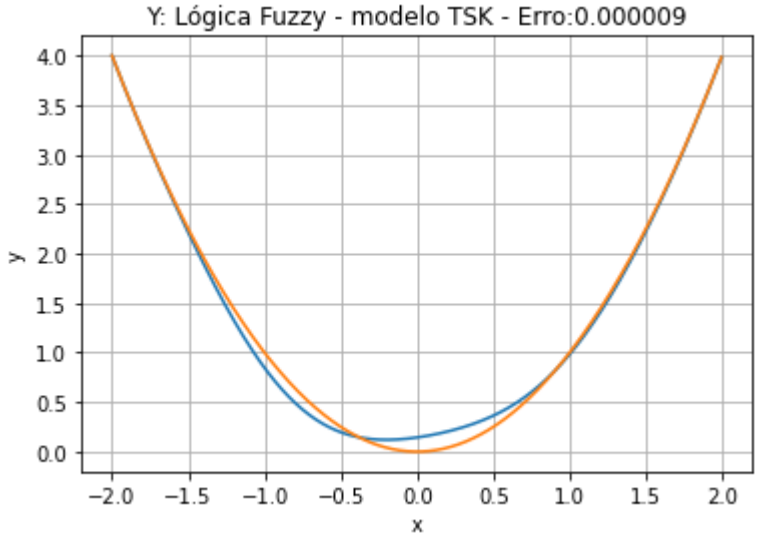
epoca: 175



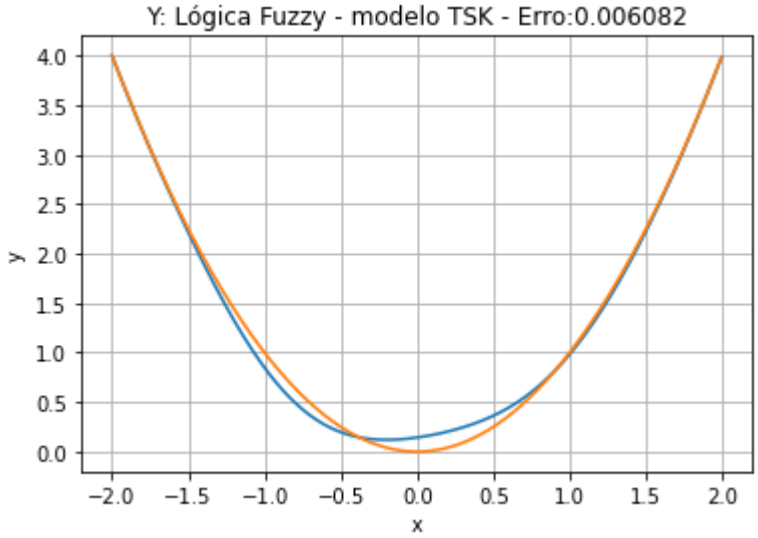
epoca: 176



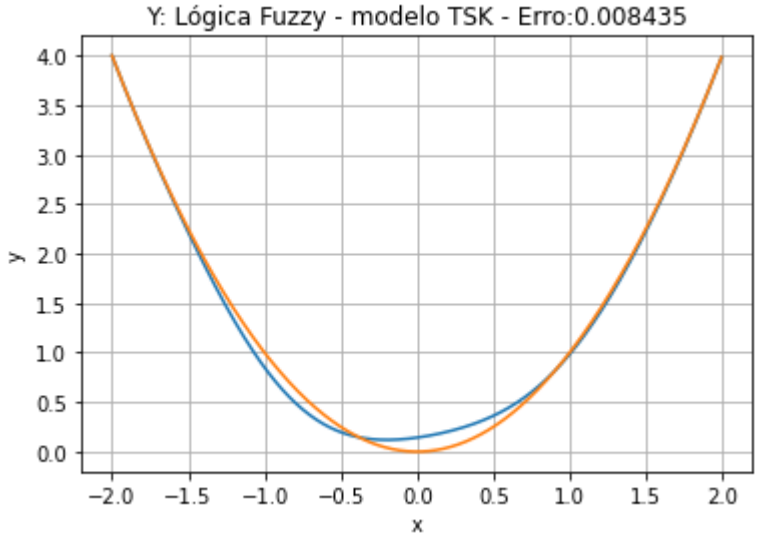
epoca: 177



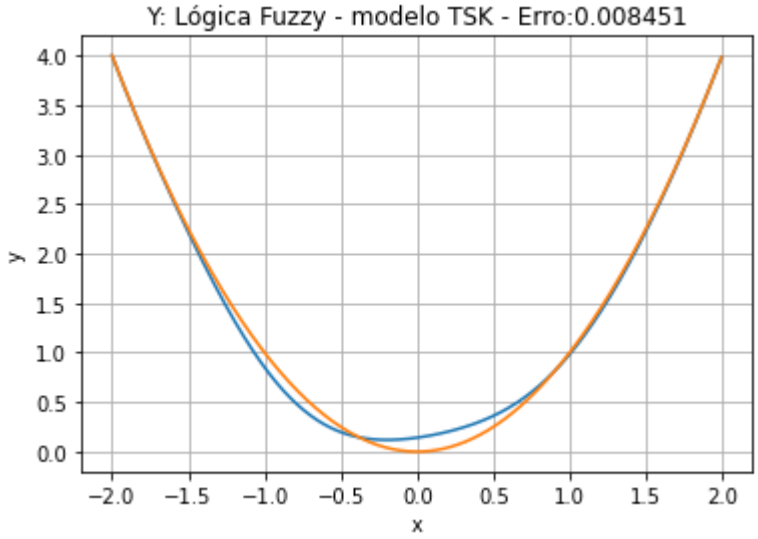
epoca: 178



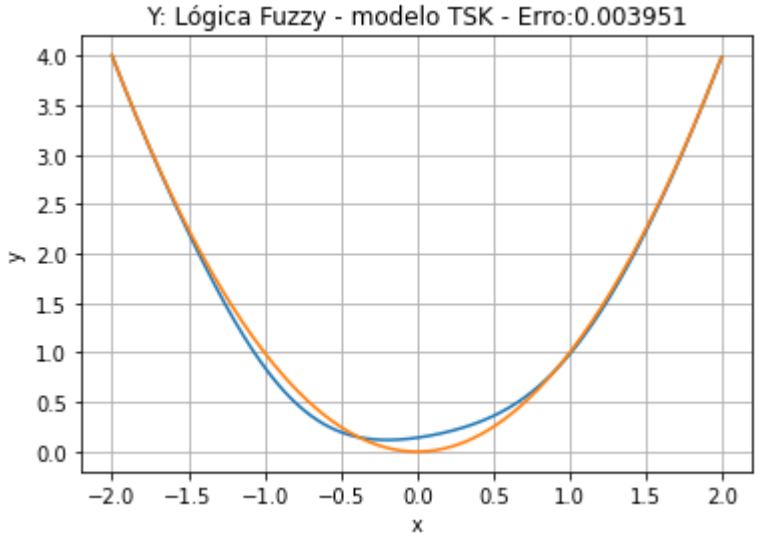
epoca: 179



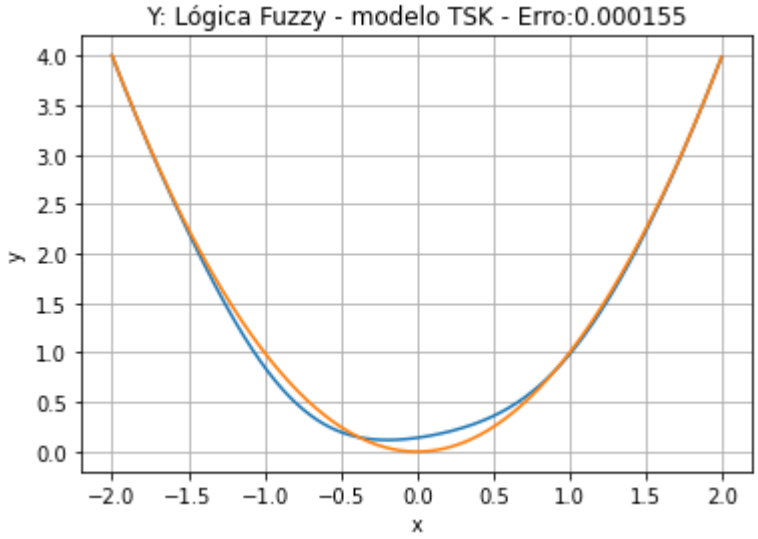
epoca: 180



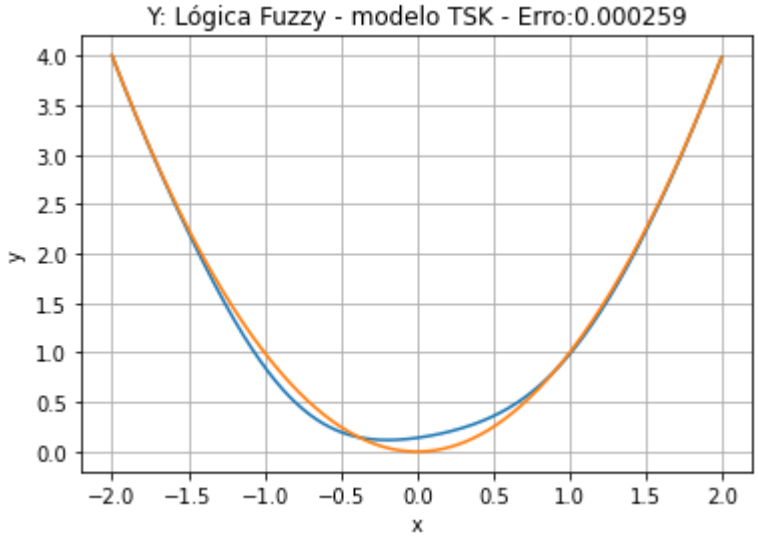
epoca: 181



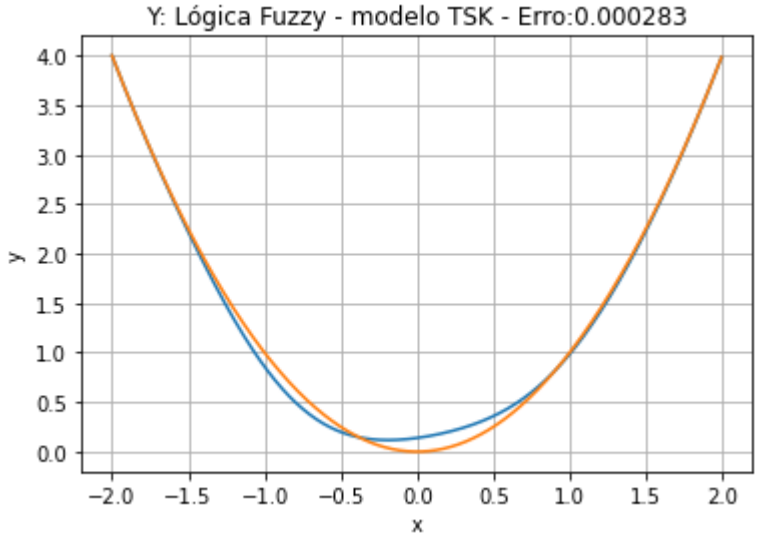
epoca: 182



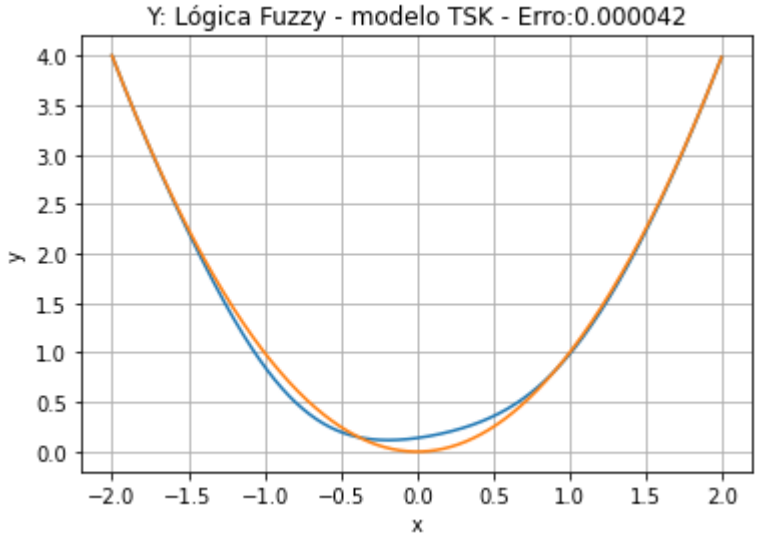
epoca: 183



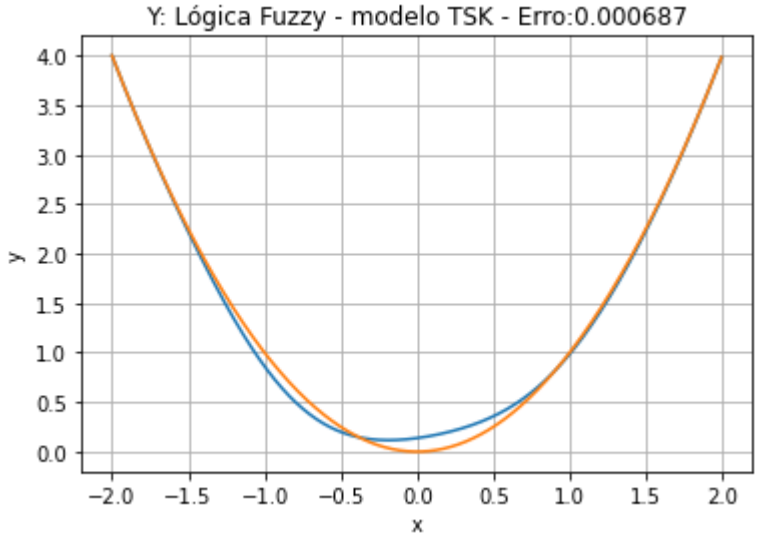
epoca: 184



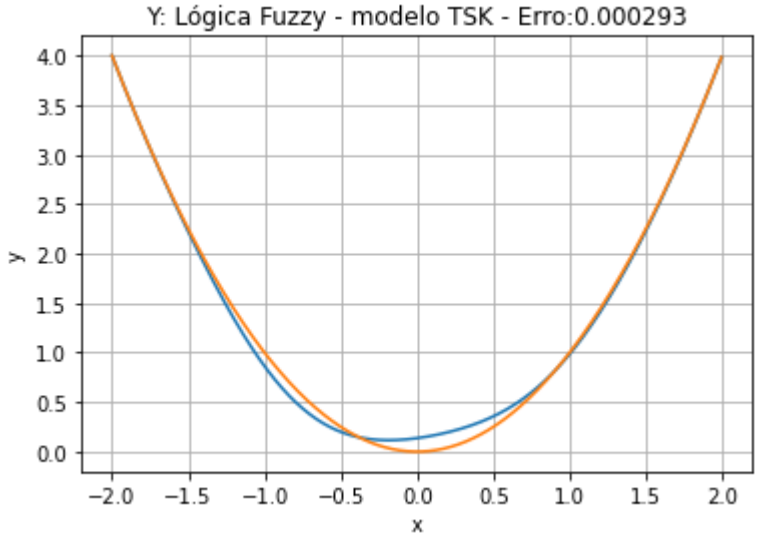
epoca: 185



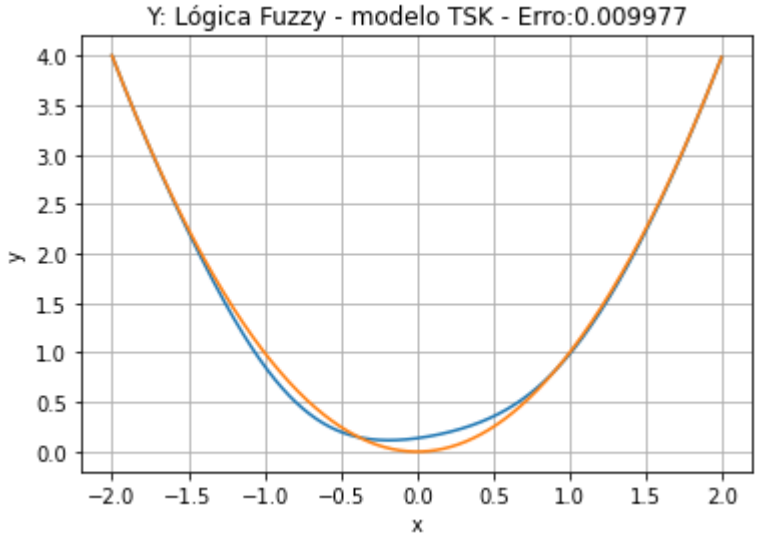
epoca: 186



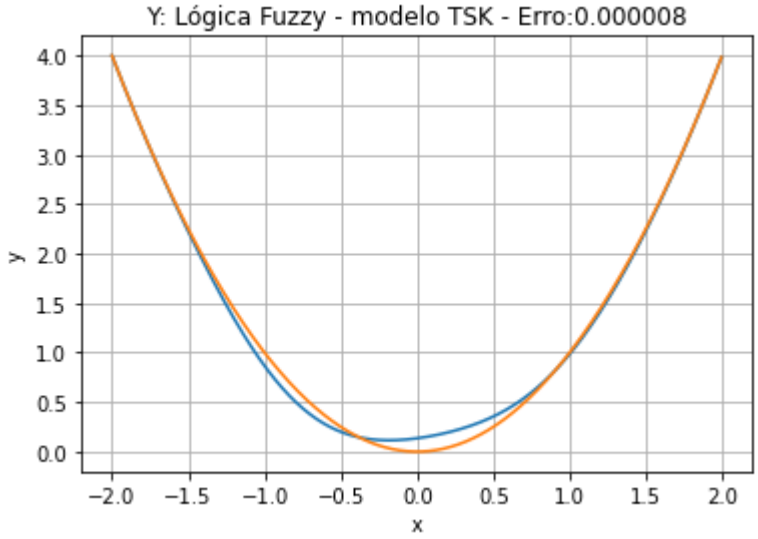
epoca: 187



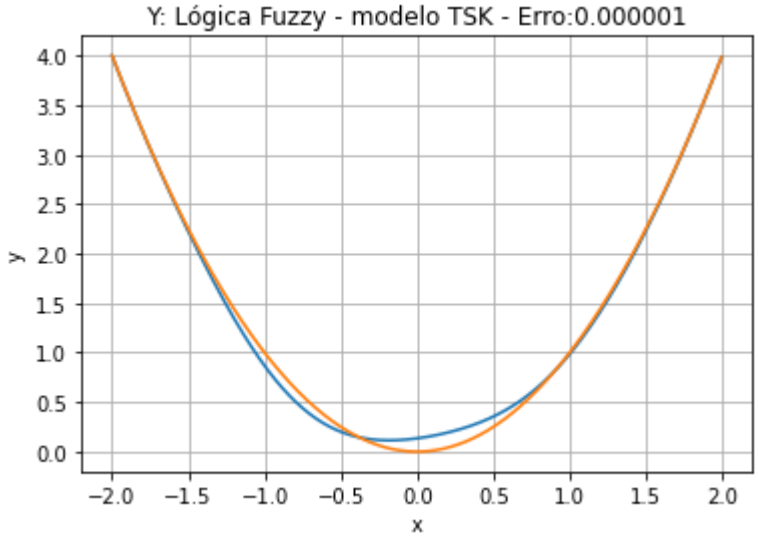
epoca: 188



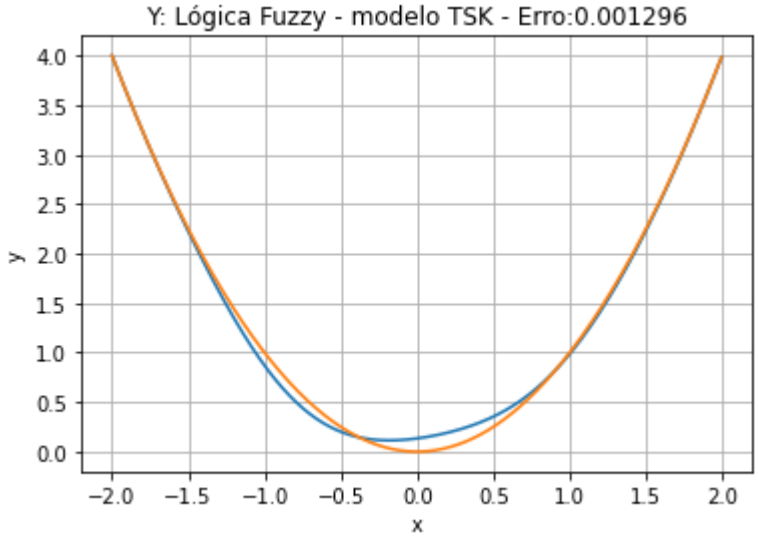
epoca: 189



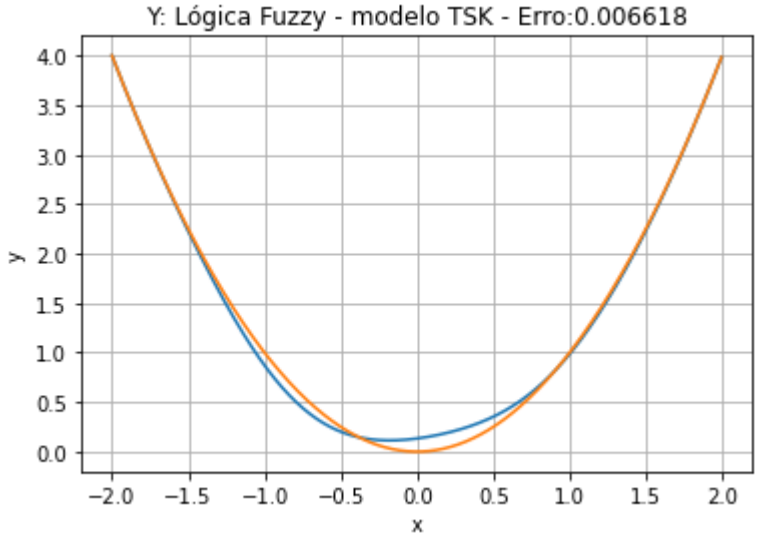
epoca: 190



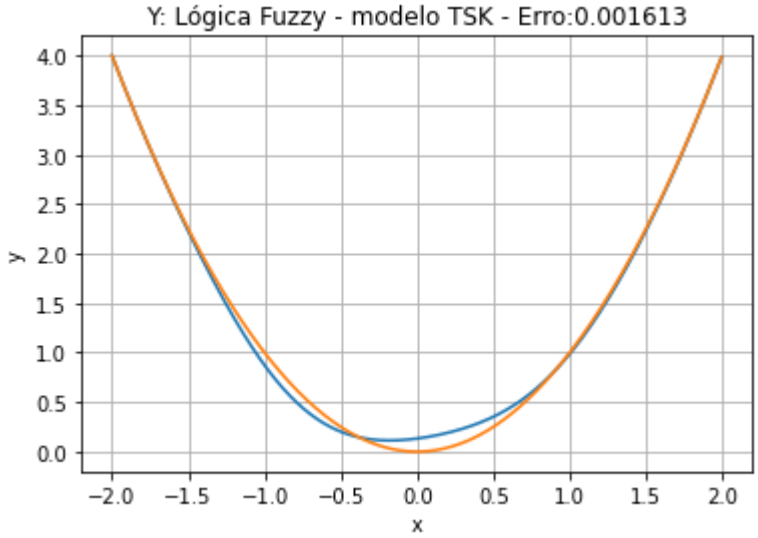
epoca: 191



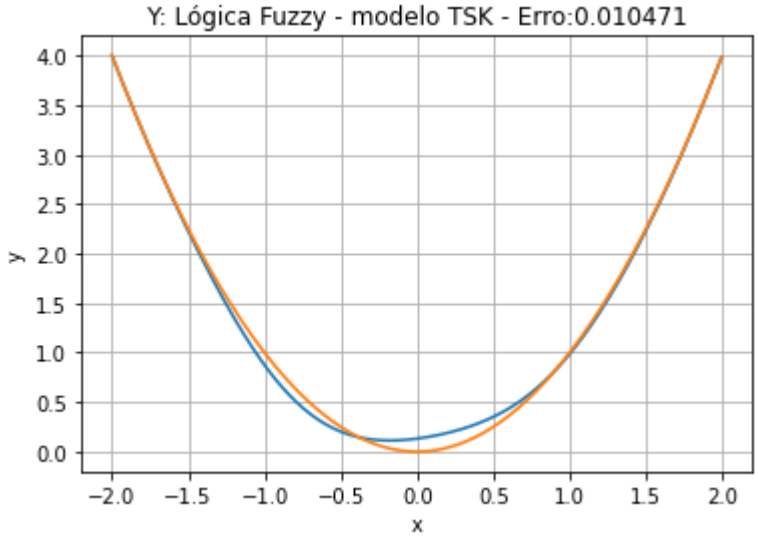
epoca: 192



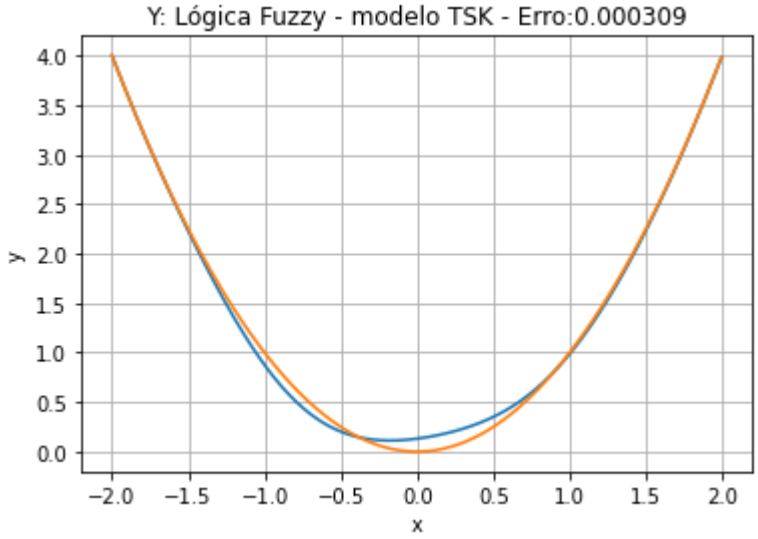
epoca: 193



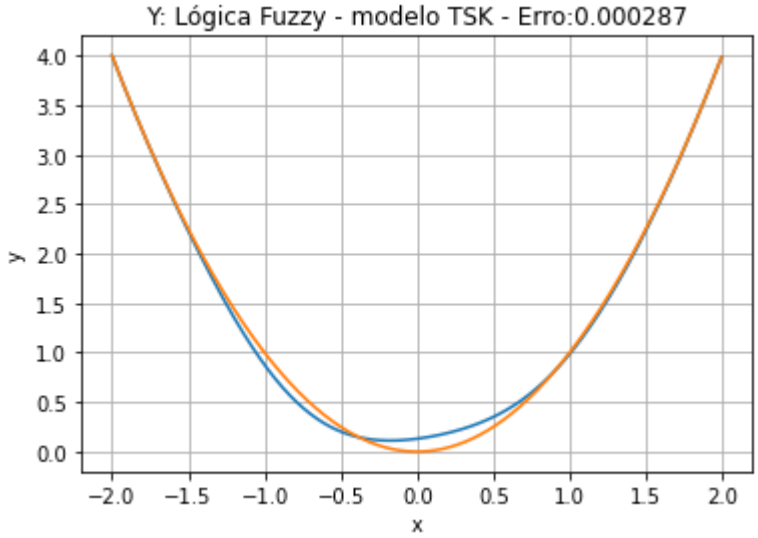
epoca: 194



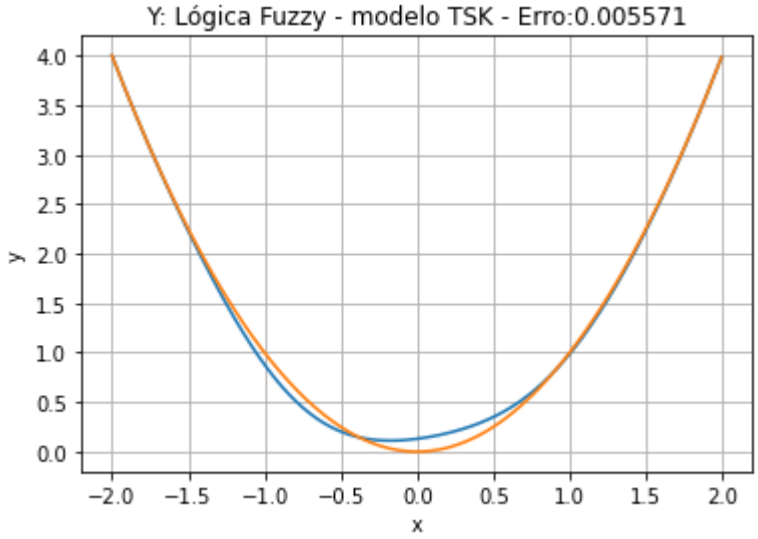
epoca: 195



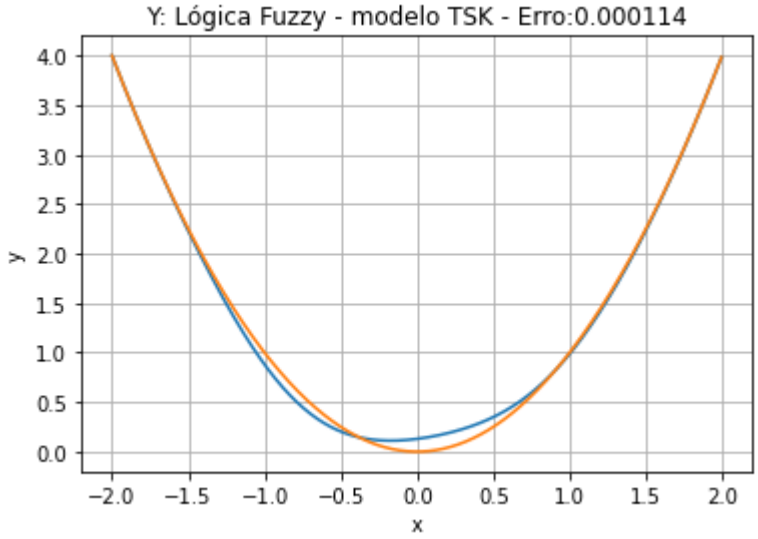
epoca: 196



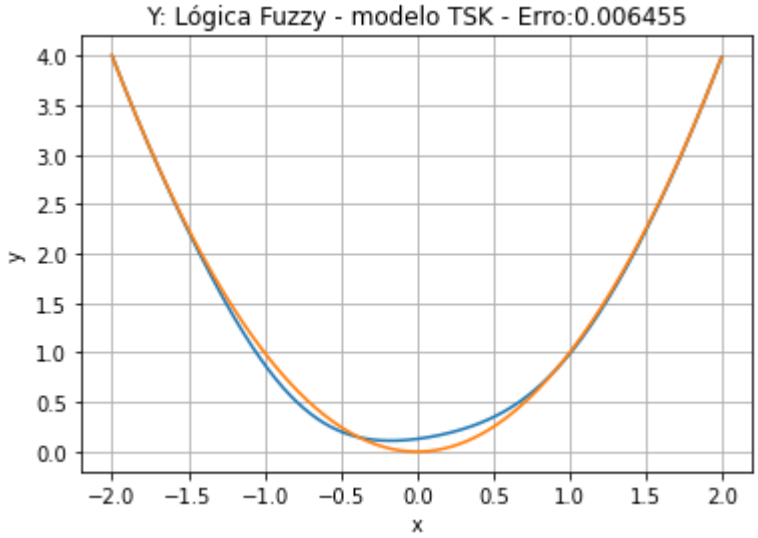
epoca: 197



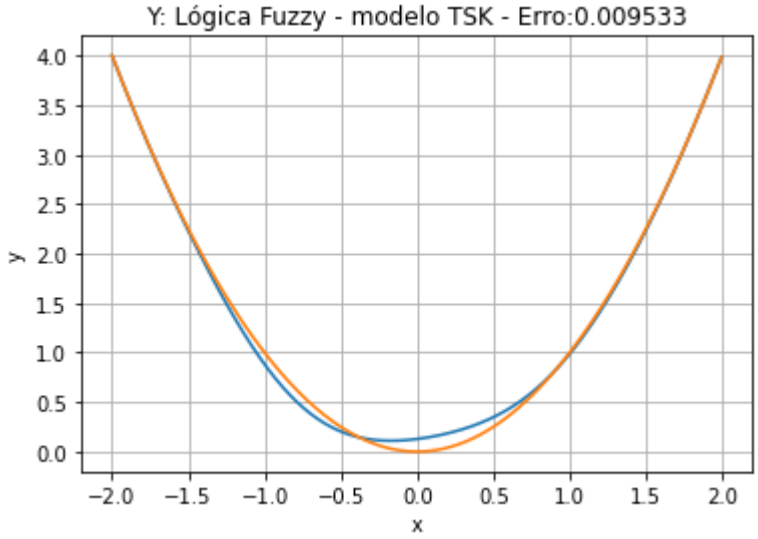
epoca: 198



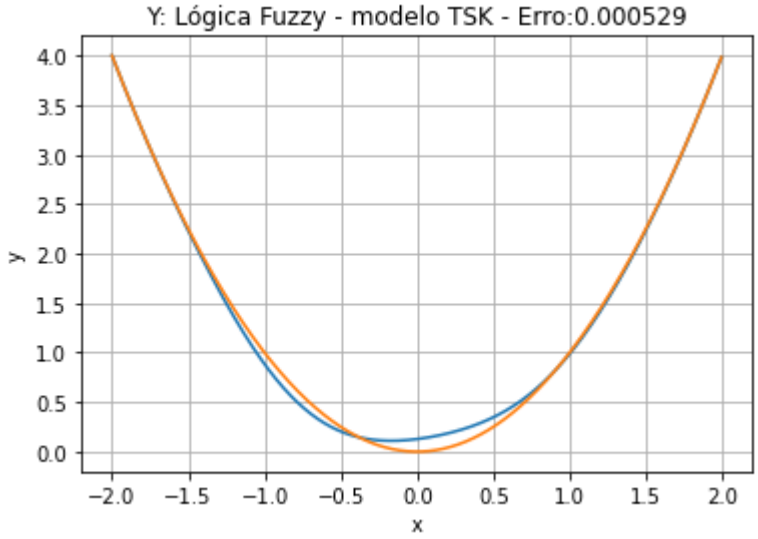
epoca: 199



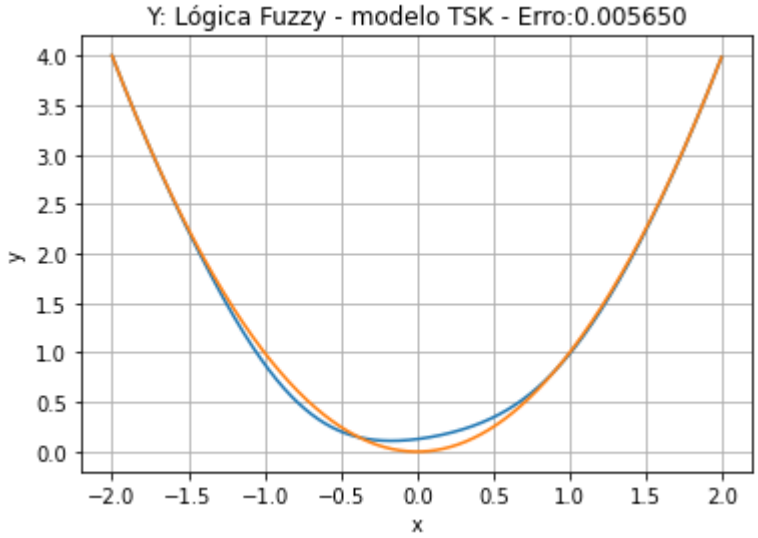
epoca: 200



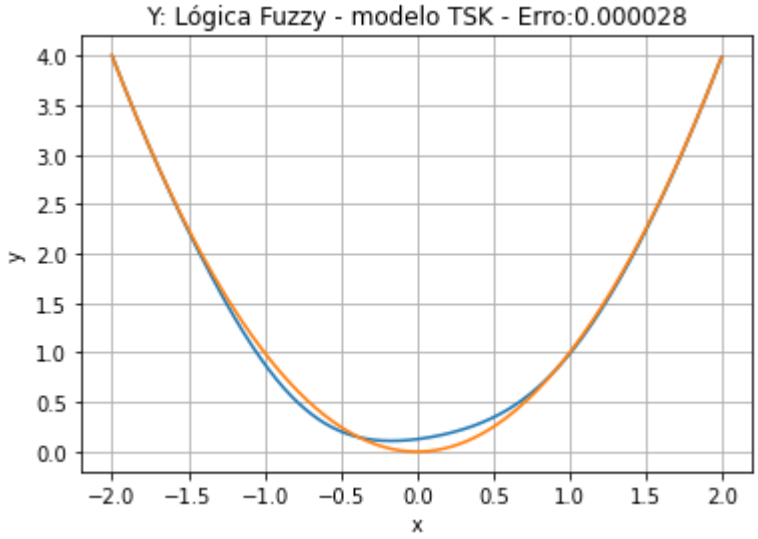
epoca: 201



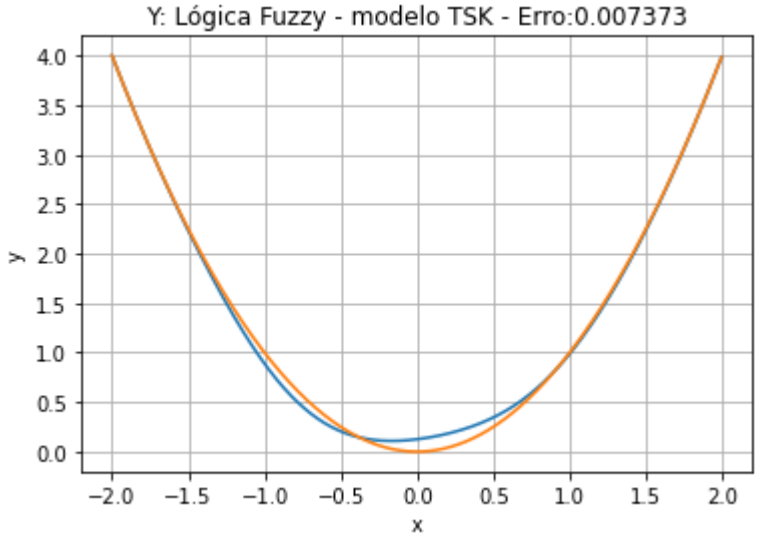
epoca: 202



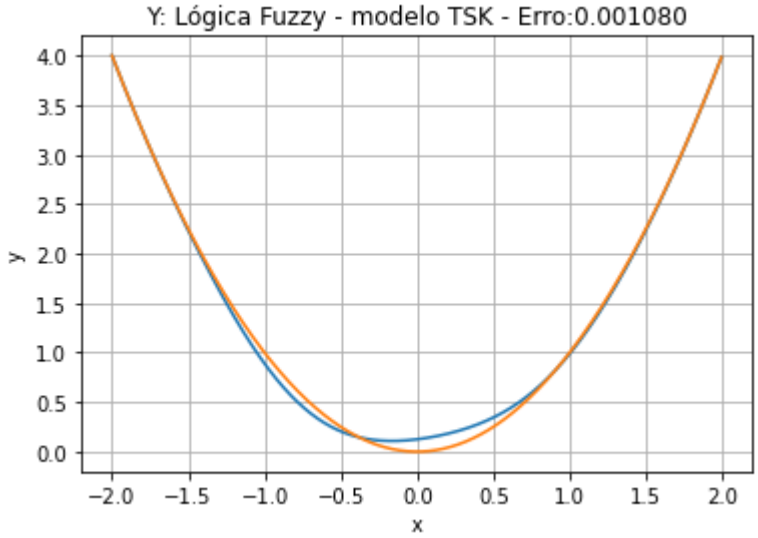
epoca: 203



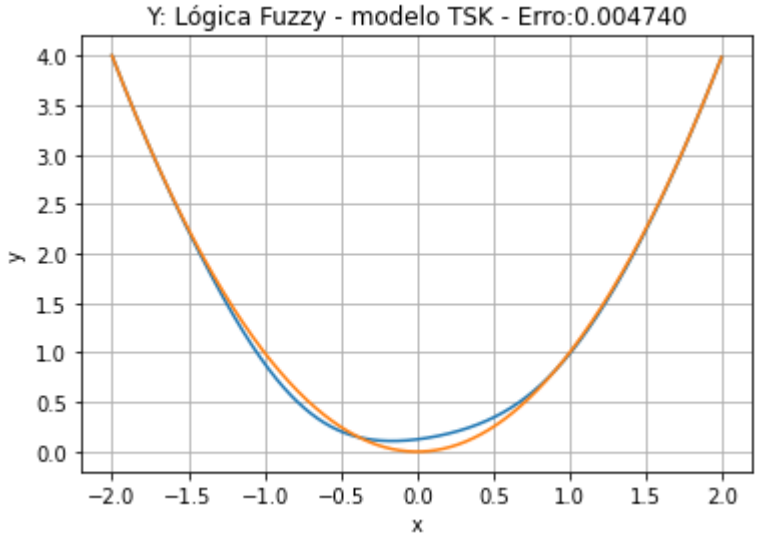
epoca: 204



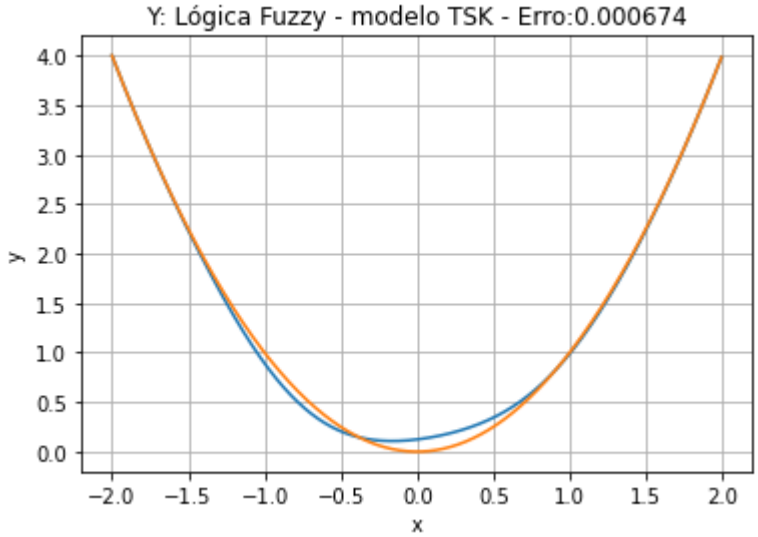
epoca: 205



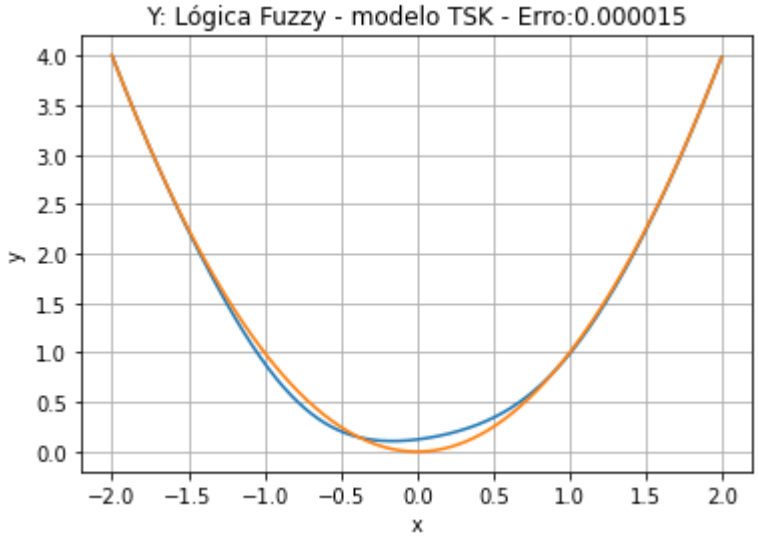
epoca: 206



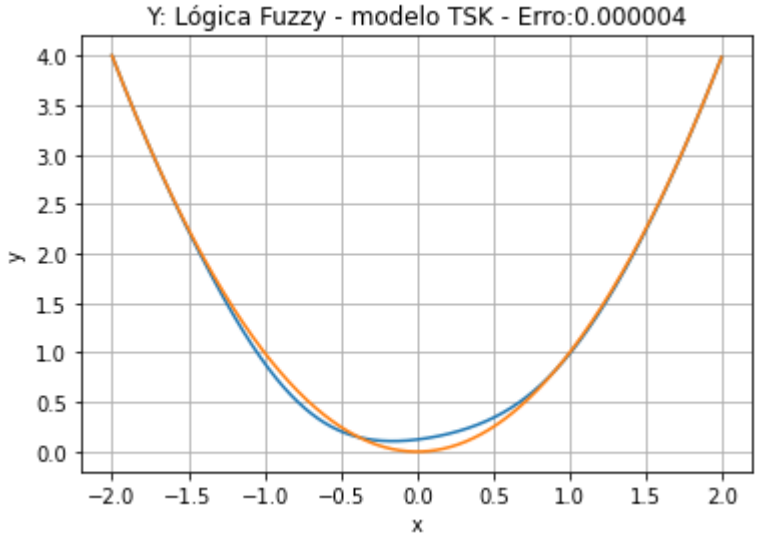
epoca: 207



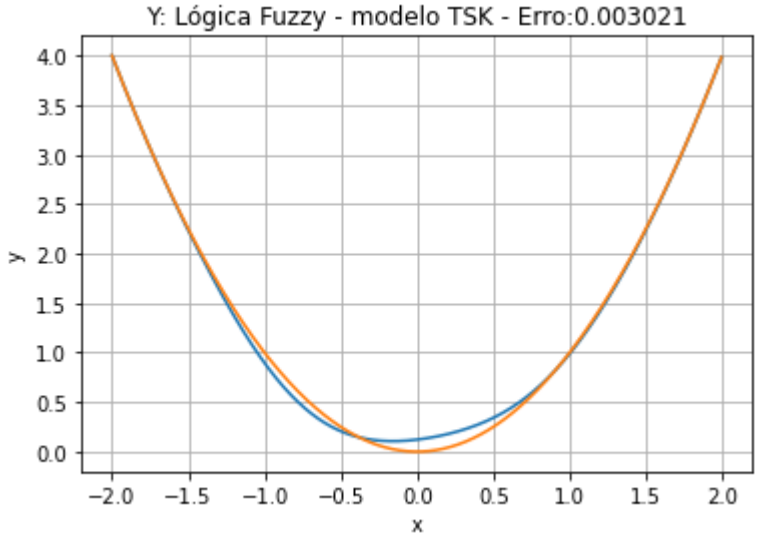
epoca: 208



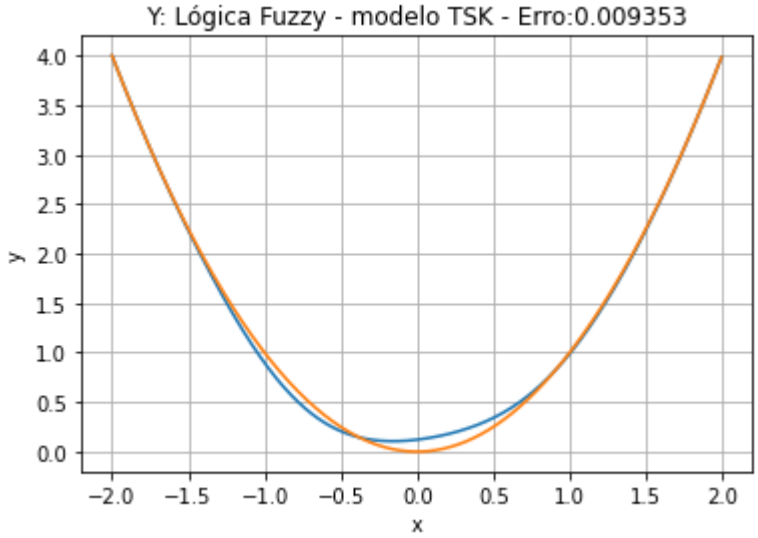
epoca: 209



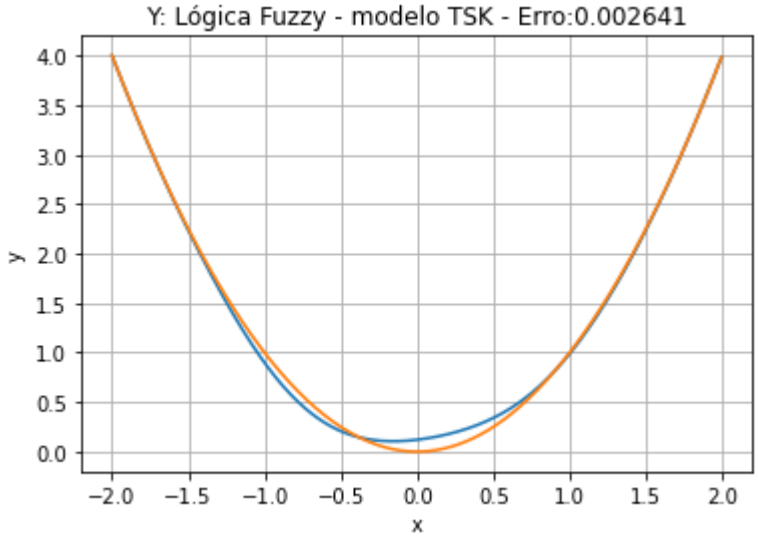
epoca: 210



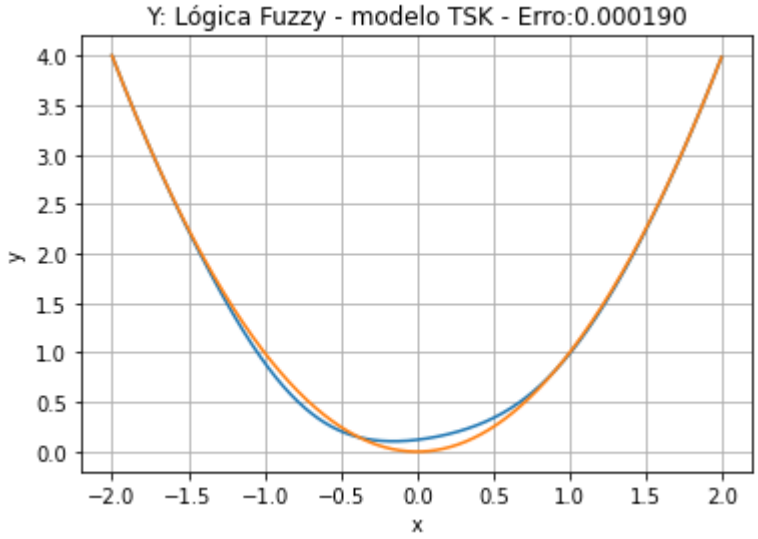
epoca: 211



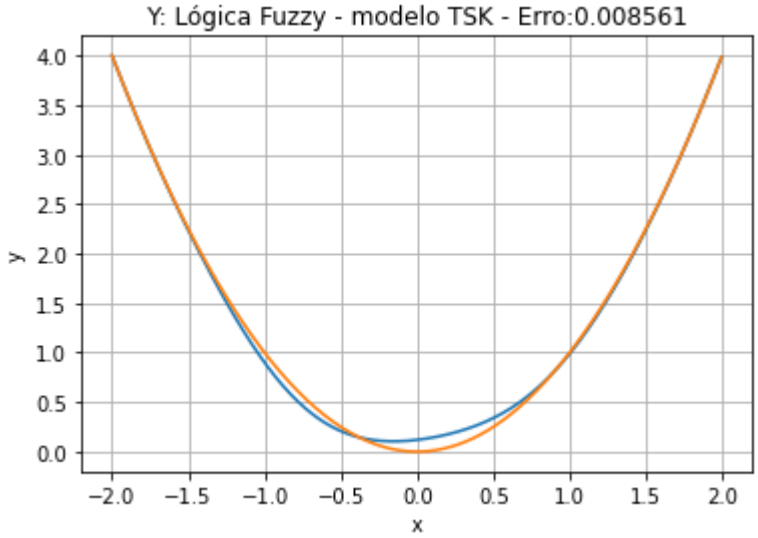
epoca: 212



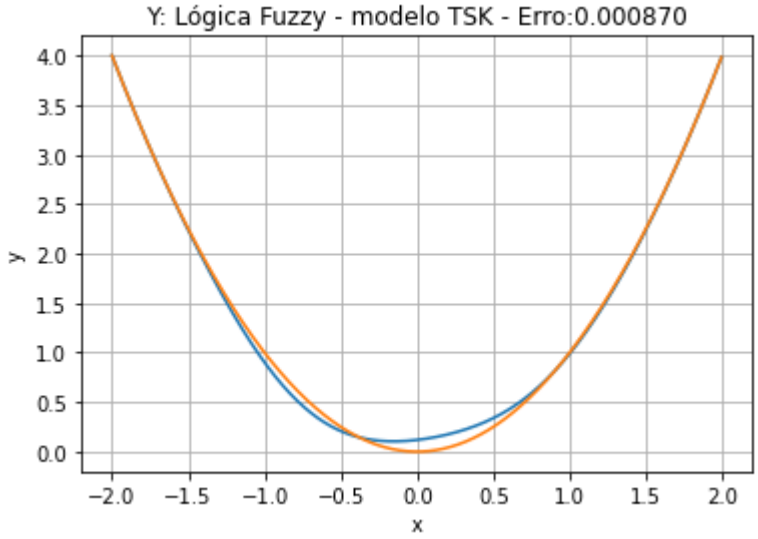
epoca: 213



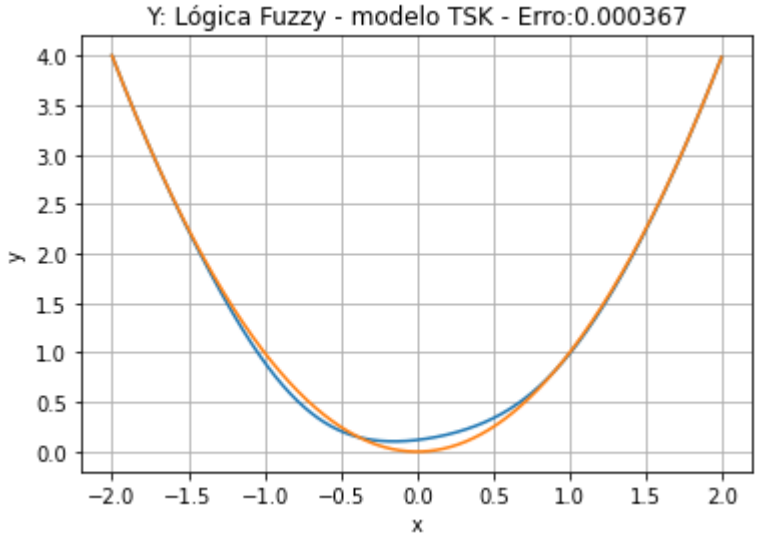
epoca: 214



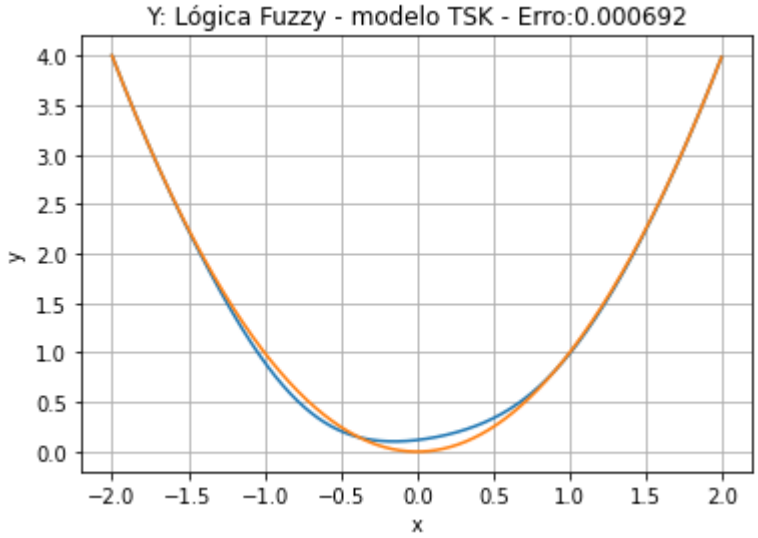
epoca: 215



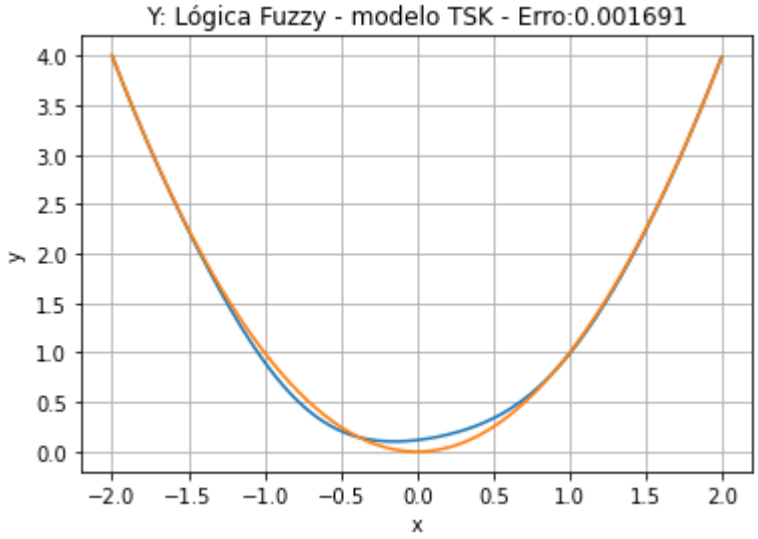
epoca: 216



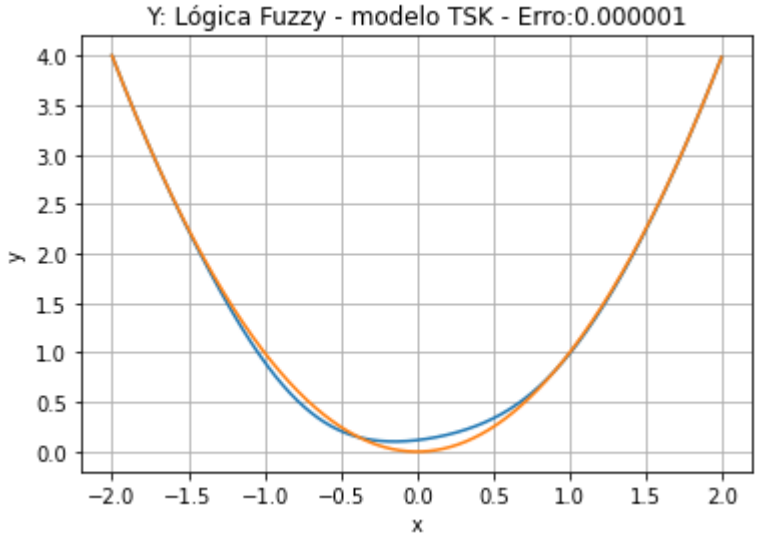
epoca: 217



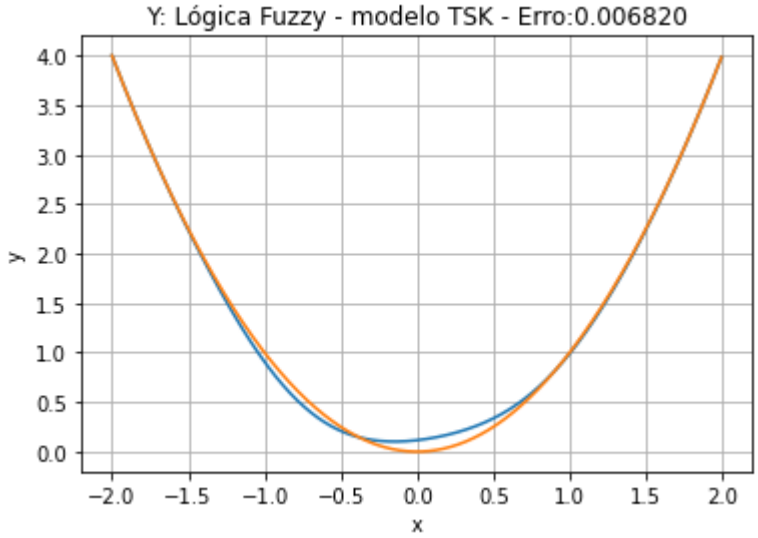
epoca: 218



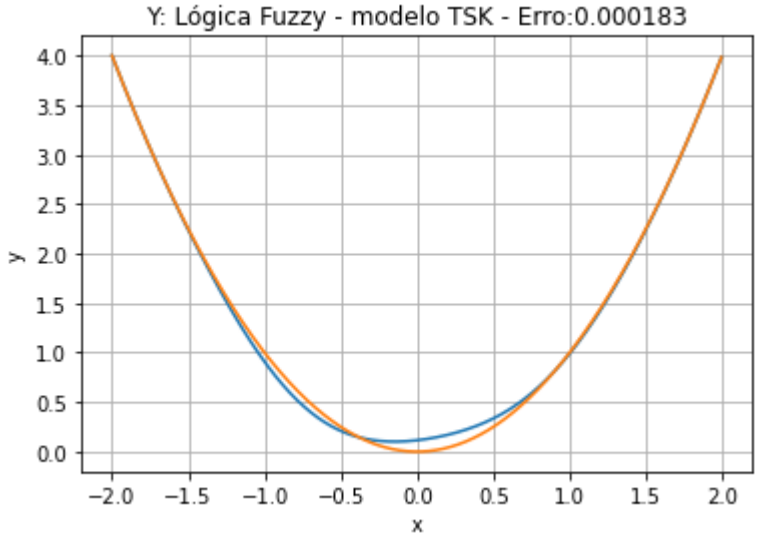
epoca: 219



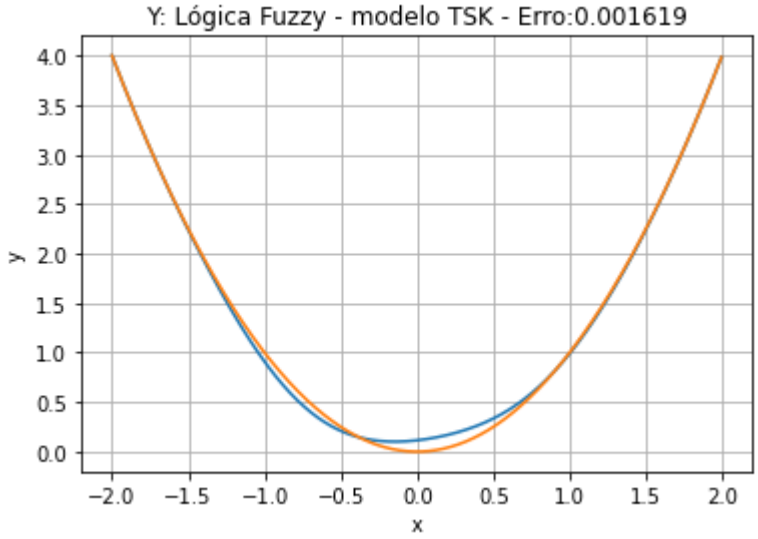
epoca: 220



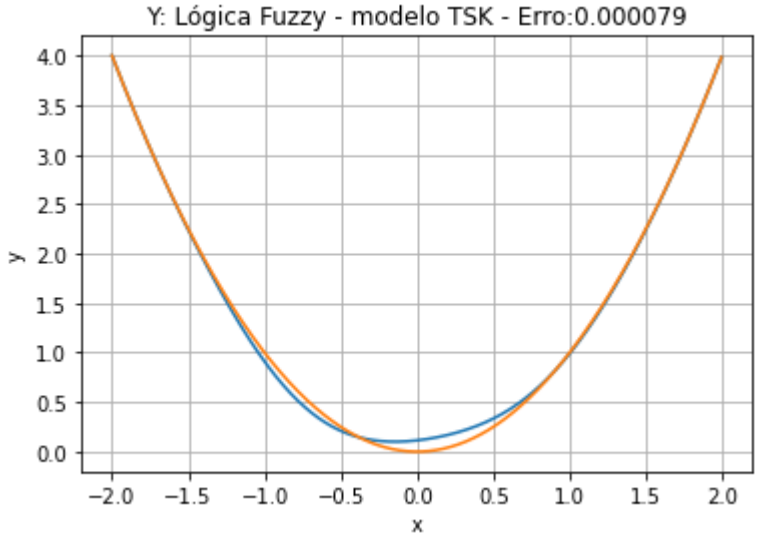
epoca: 221



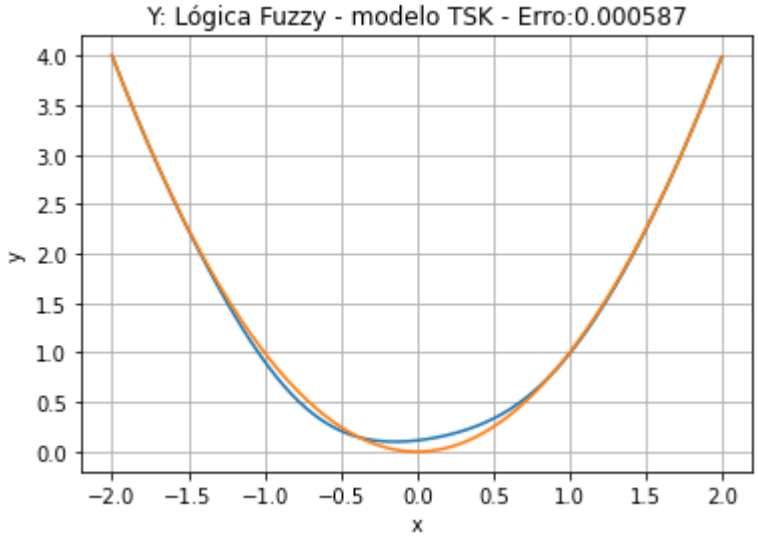
epoca: 222



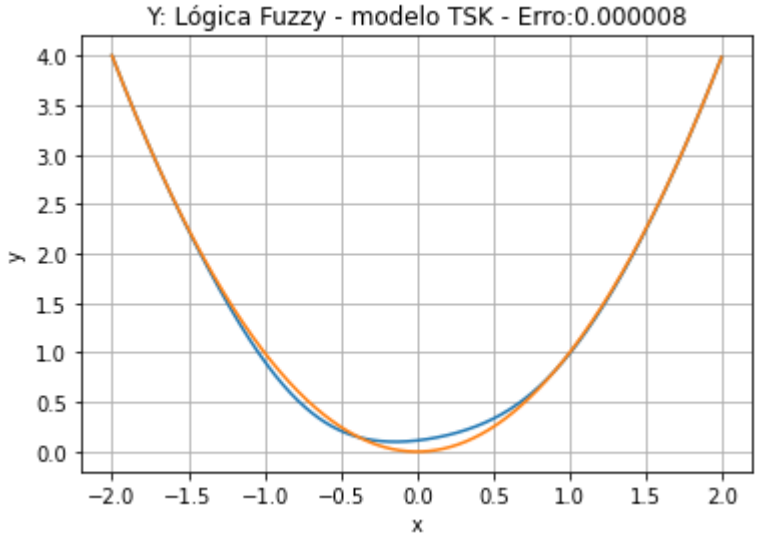
epoca: 223



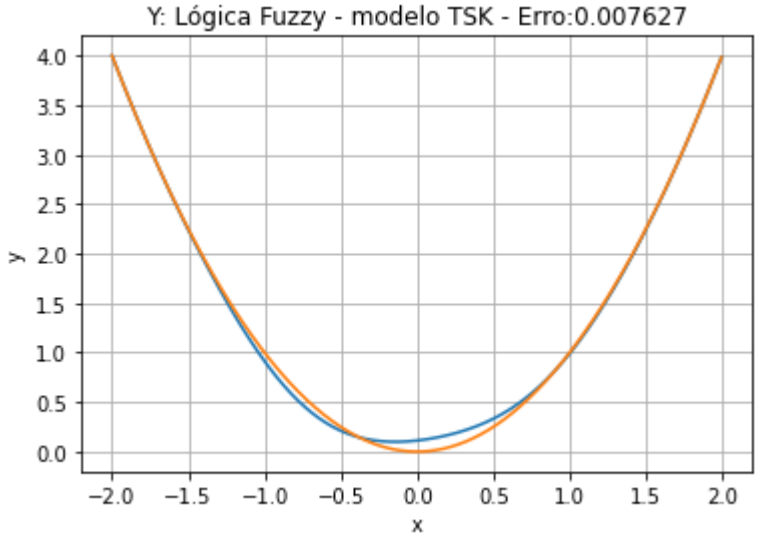
epoca: 224



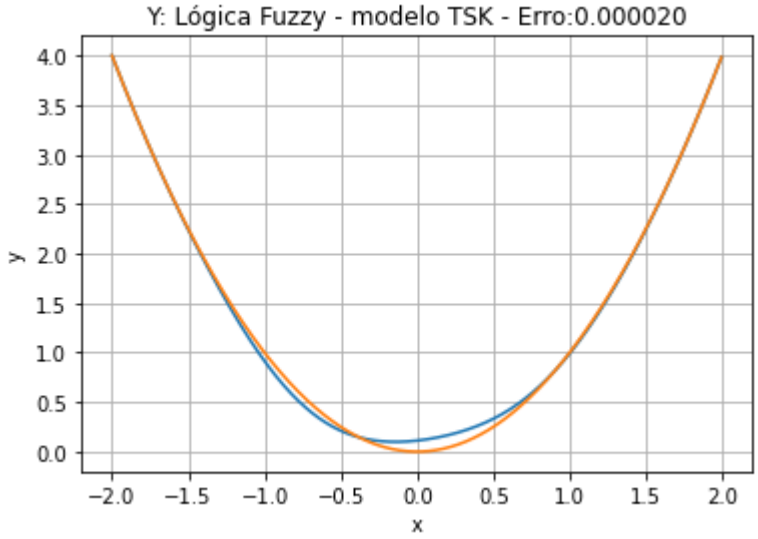
epoca: 225



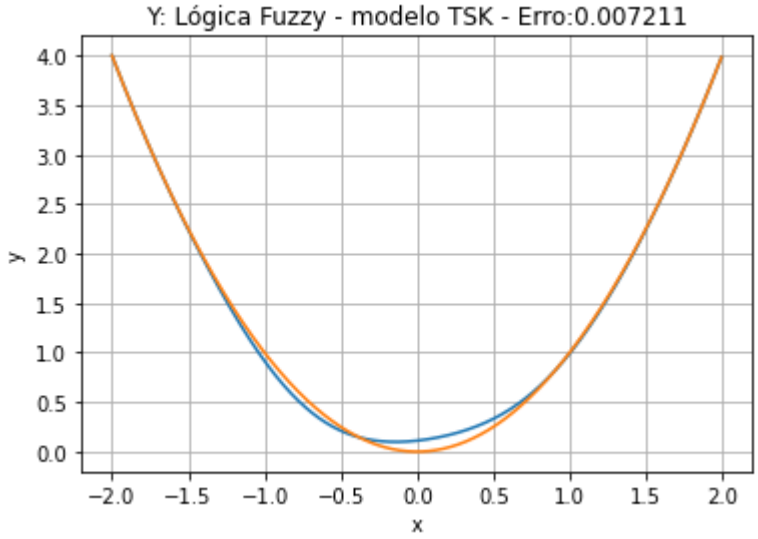
epoca: 226



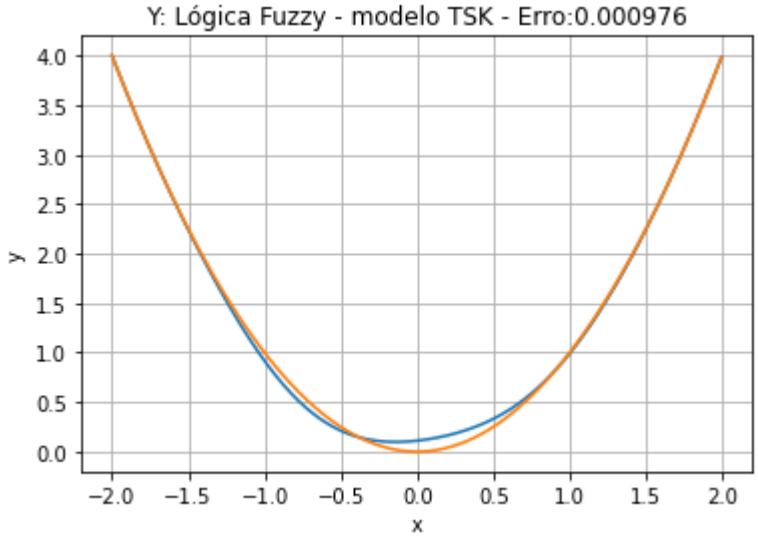
epoca: 227



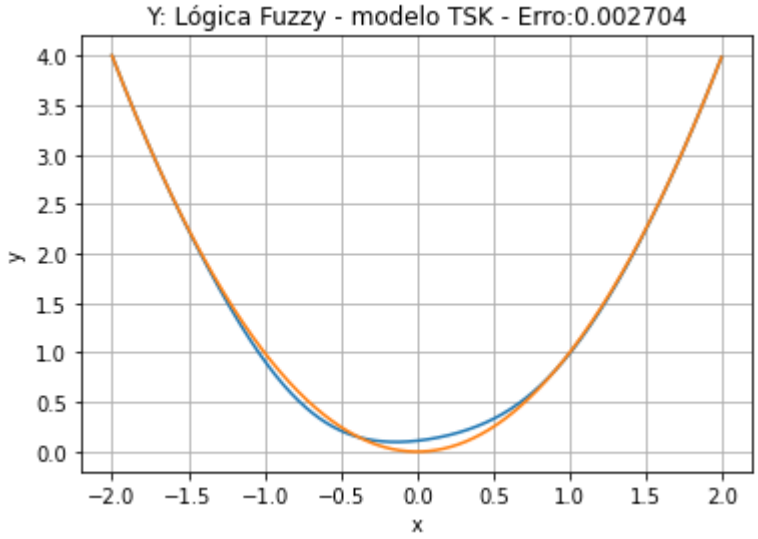
epoca: 228



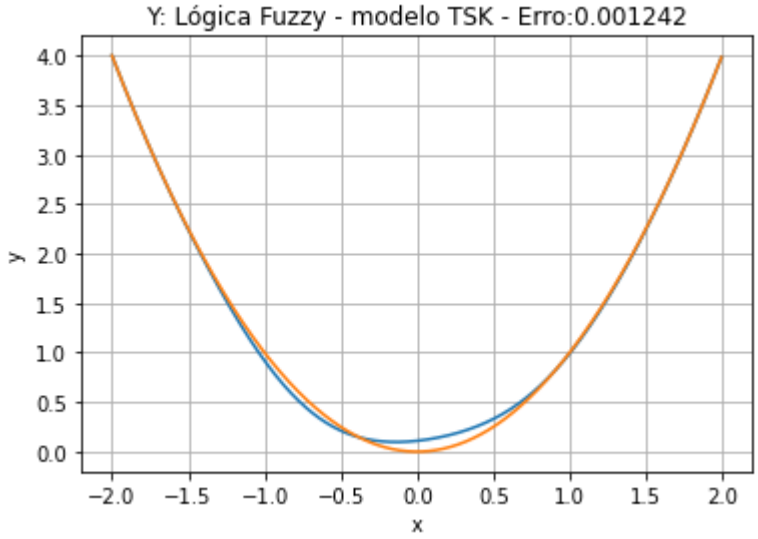
epoca: 229



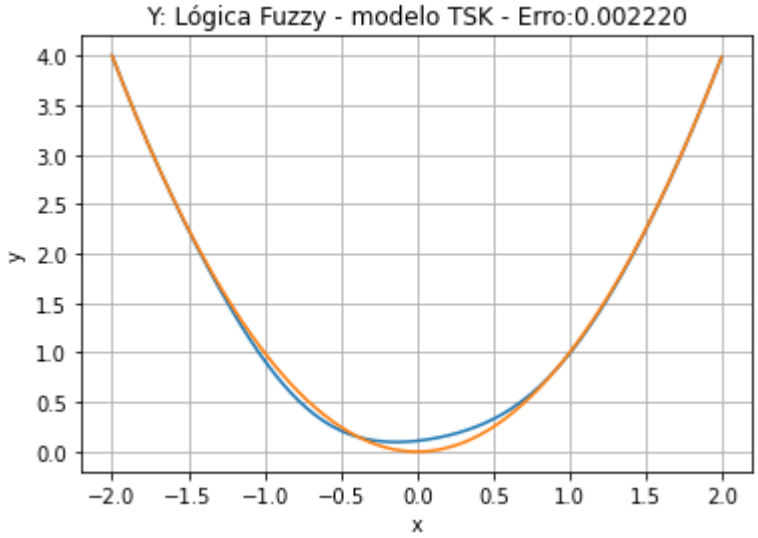
epoca: 230



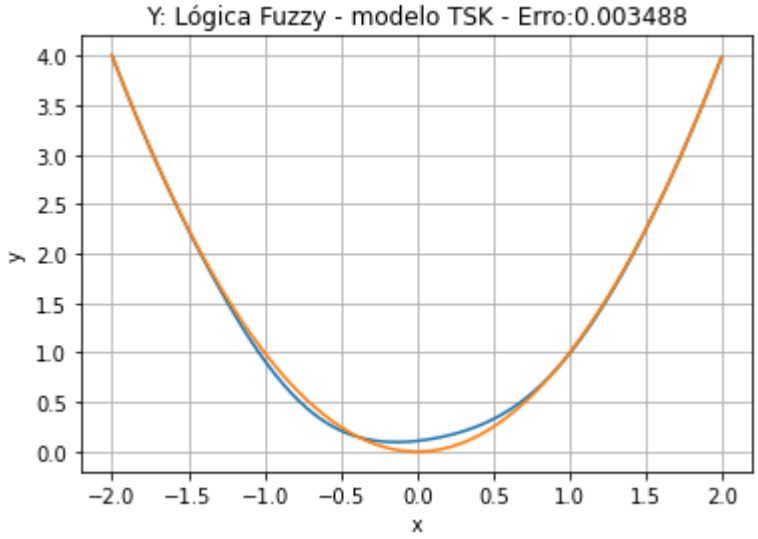
epoca: 231



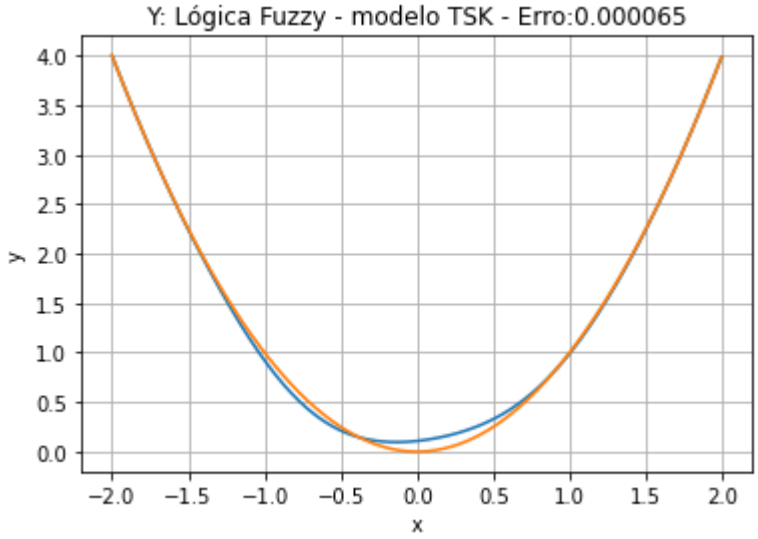
epoca: 232



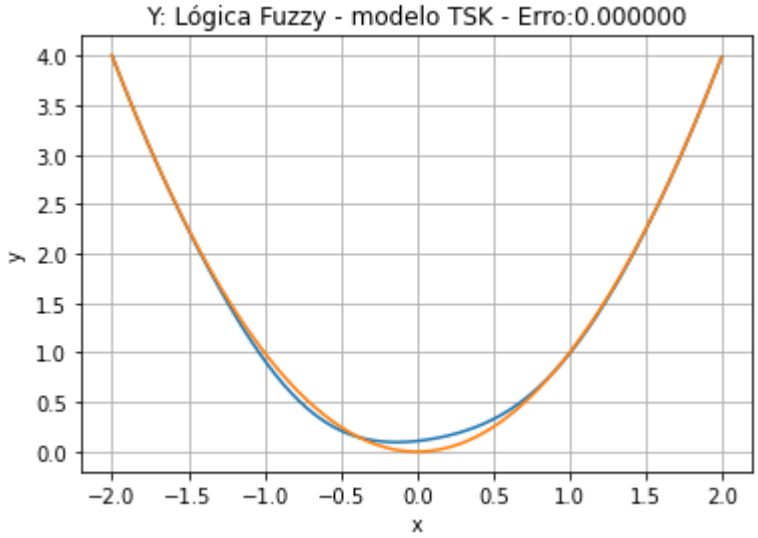
epoca: 233



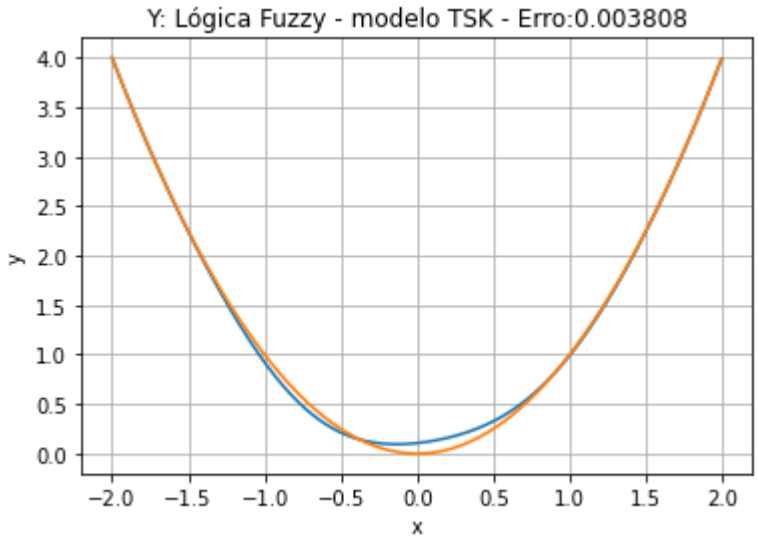
epoca: 234



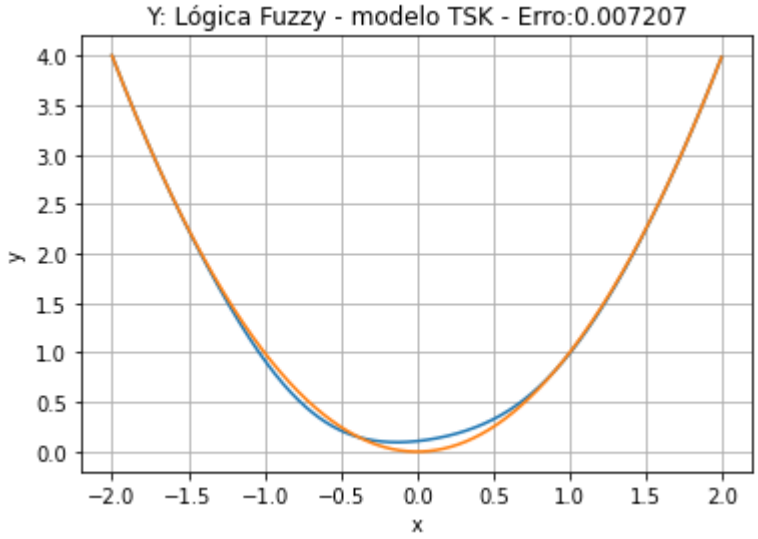
epoca: 235



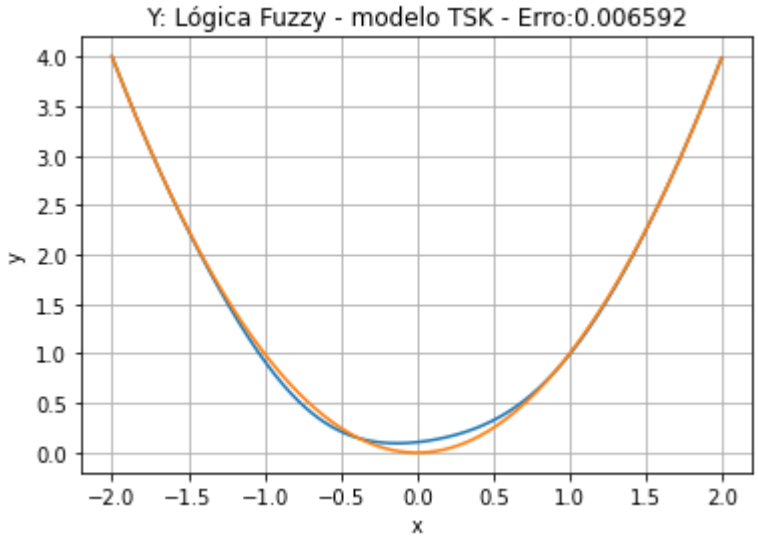
epoca: 236



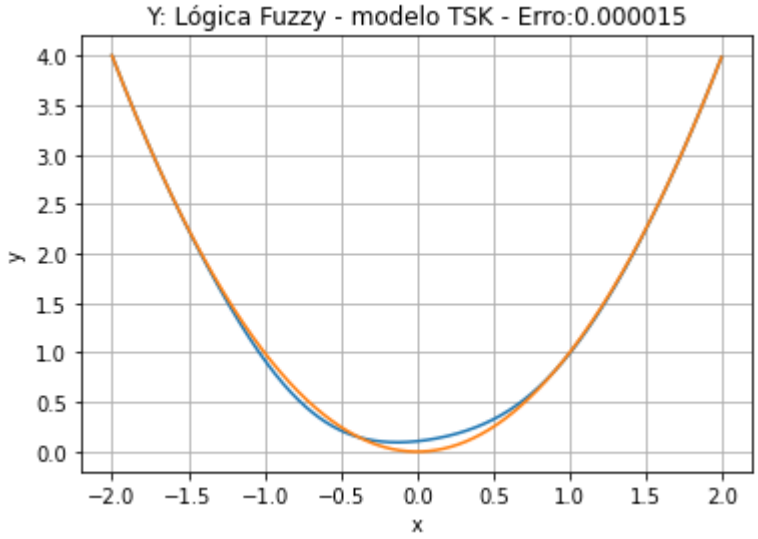
epoca: 237



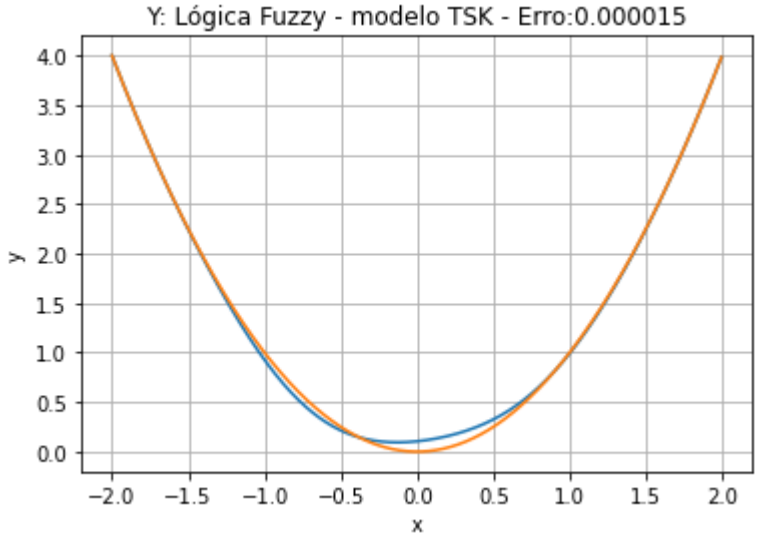
epoca: 238



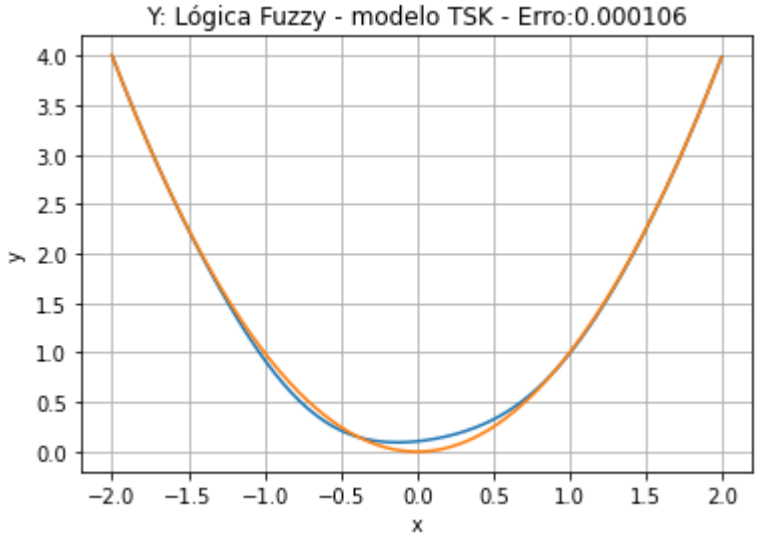
epoca: 239



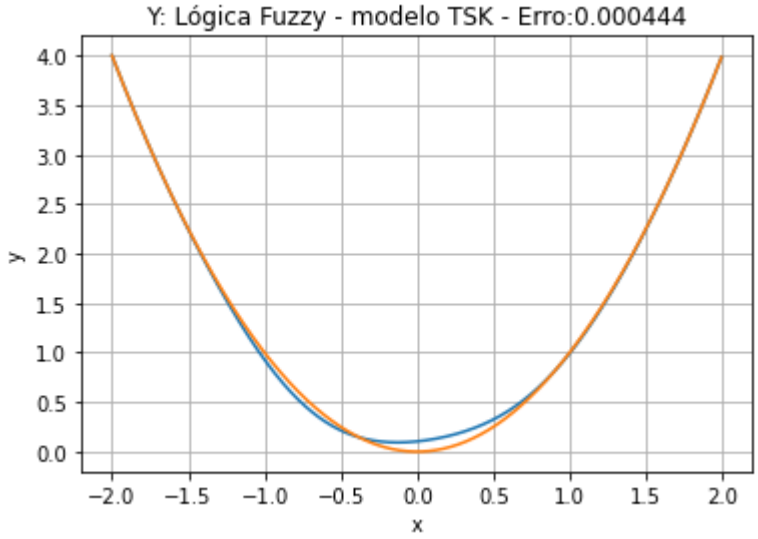
epoca: 240



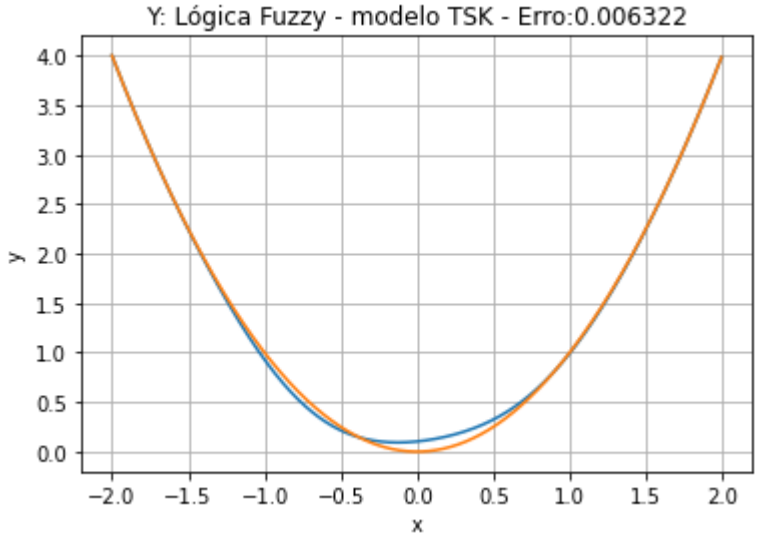
epoca: 241



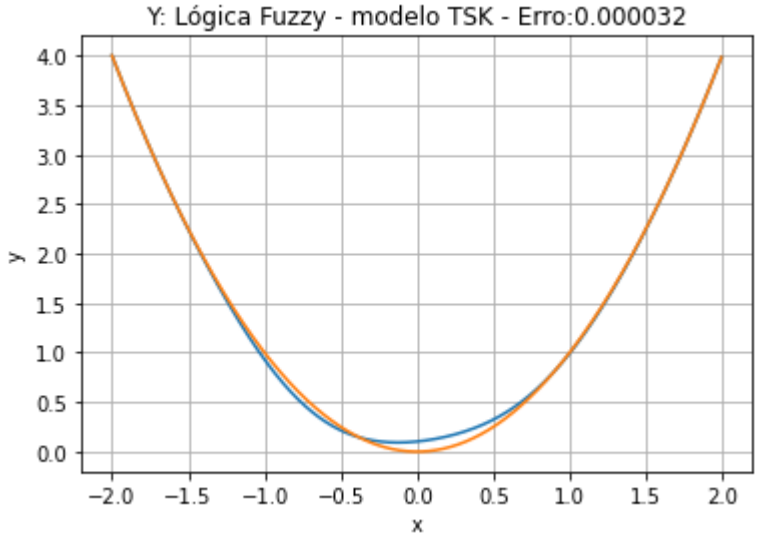
epoca: 242



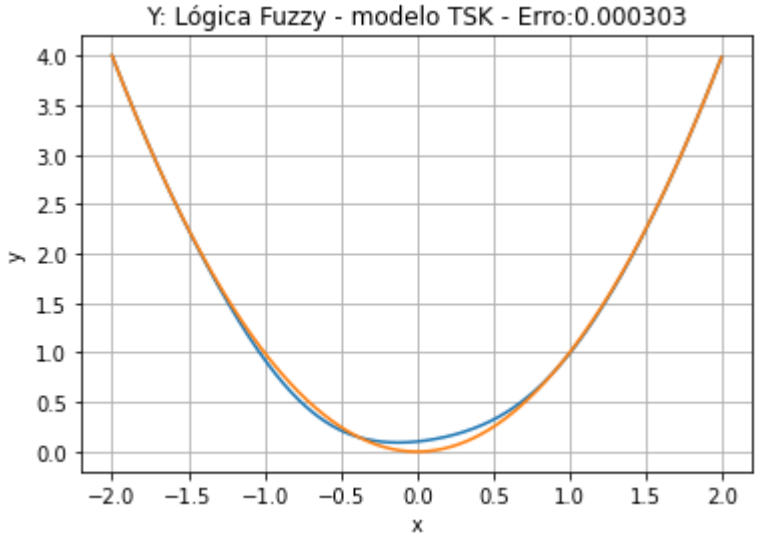
epoca: 243



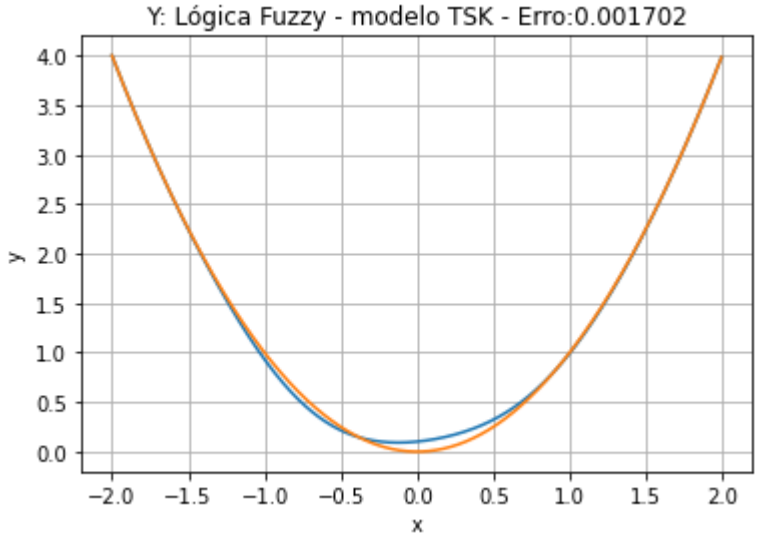
epoca: 244



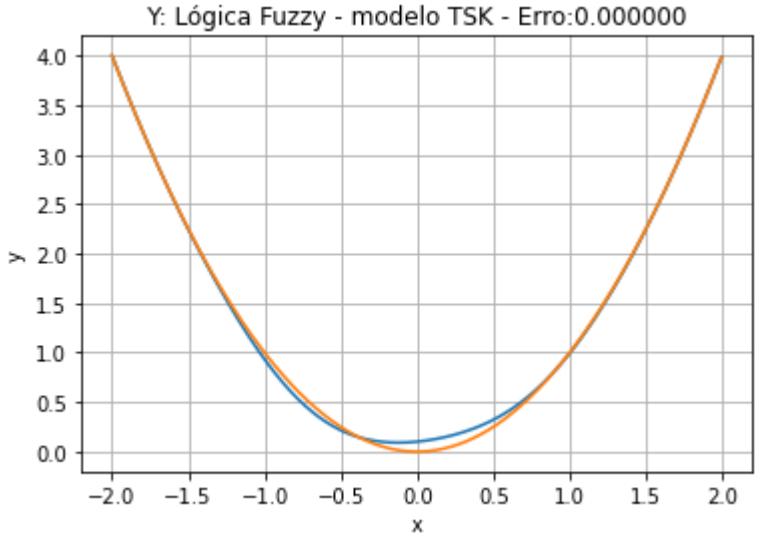
epoca: 245



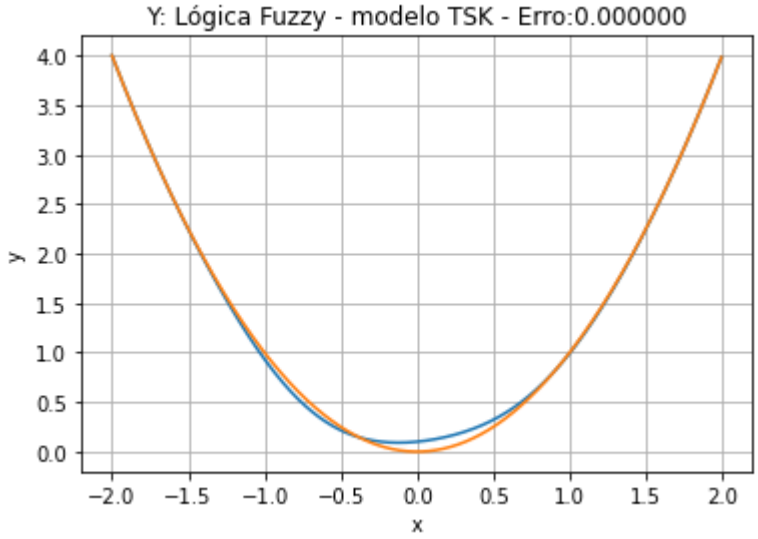
epoca: 246



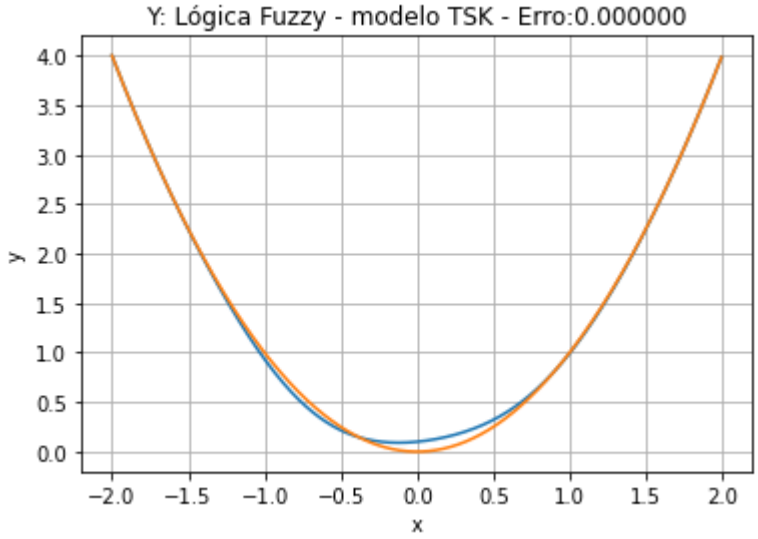
epoca: 247



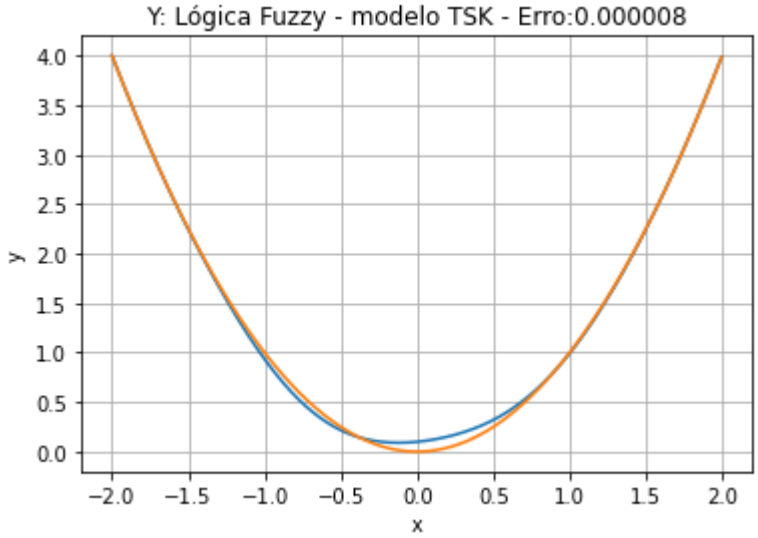
epoca: 248



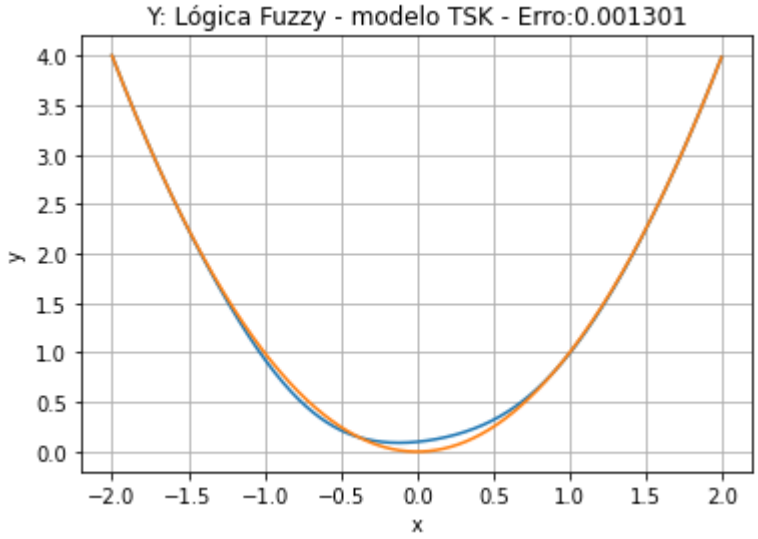
epoca: 249



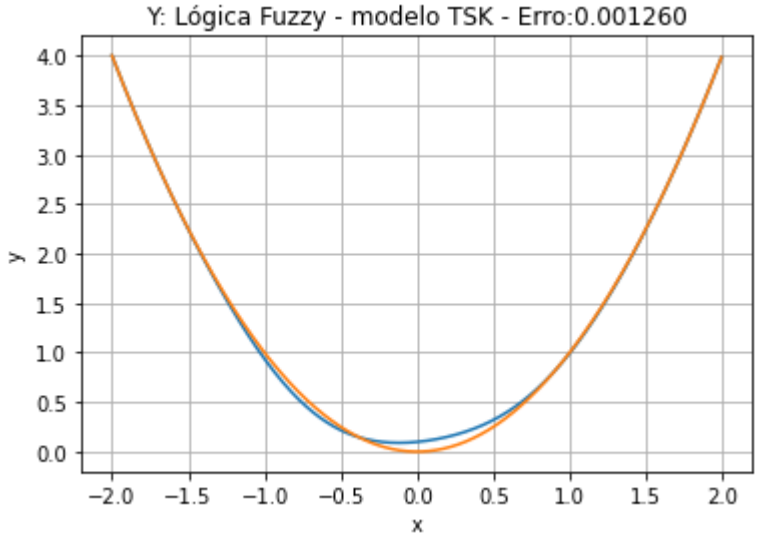
epoca: 250



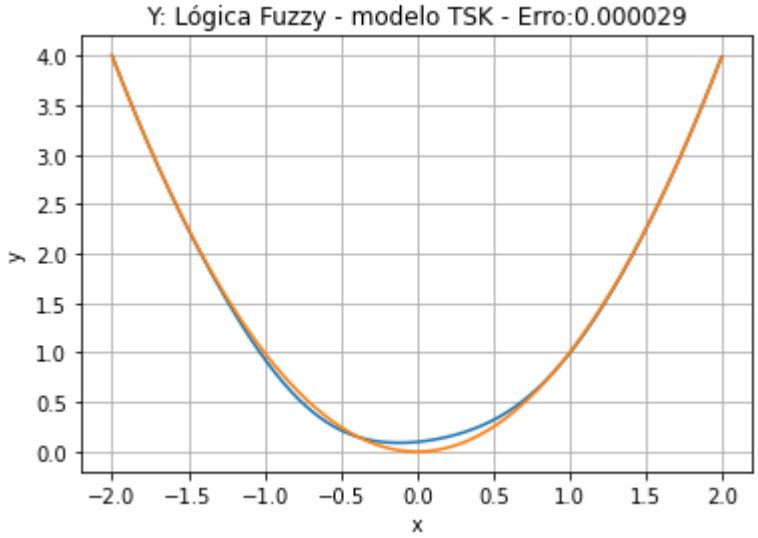
epoca: 251



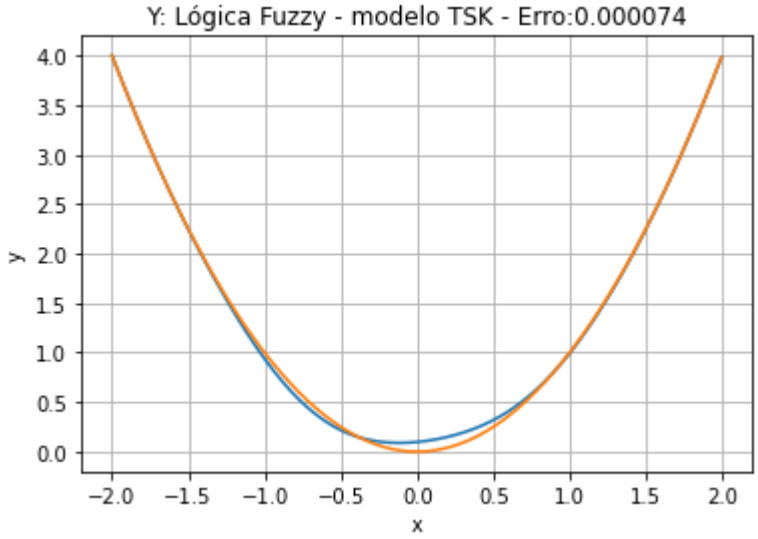
epoca: 252



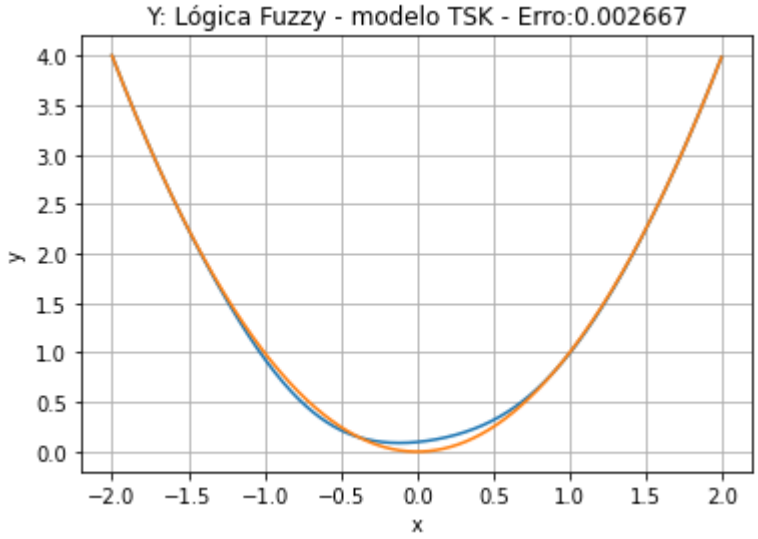
epoca: 253



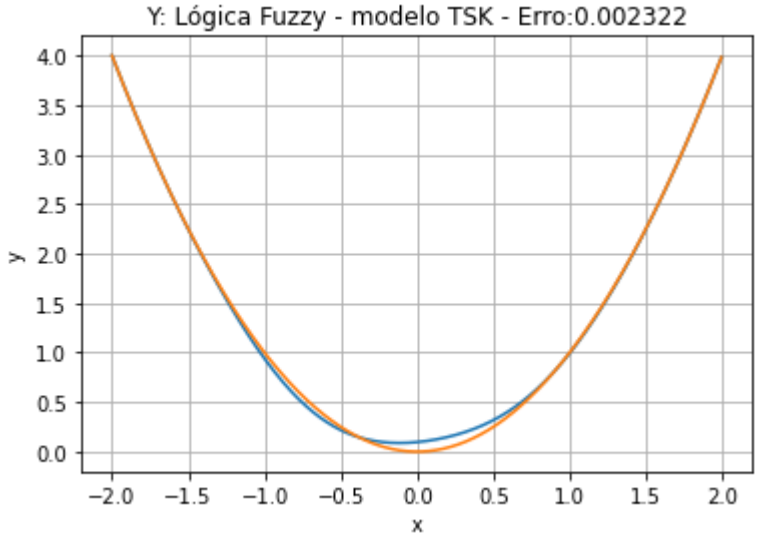
epoca: 254



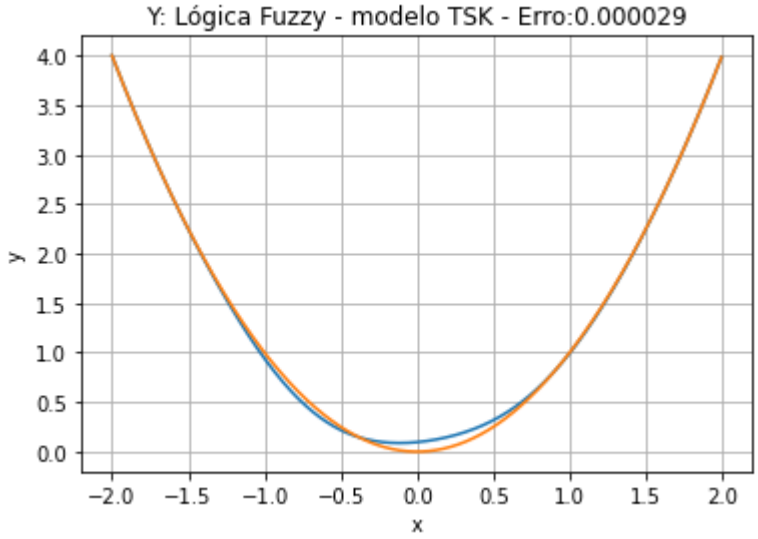
epoca: 255



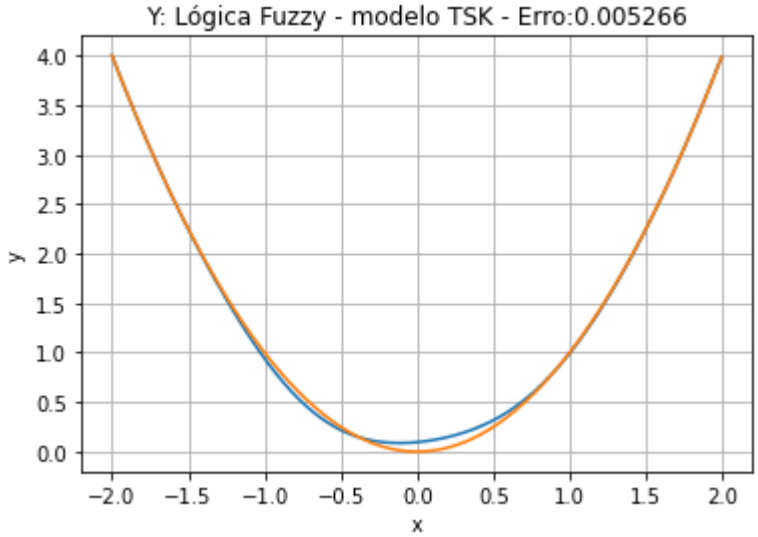
epoca: 256



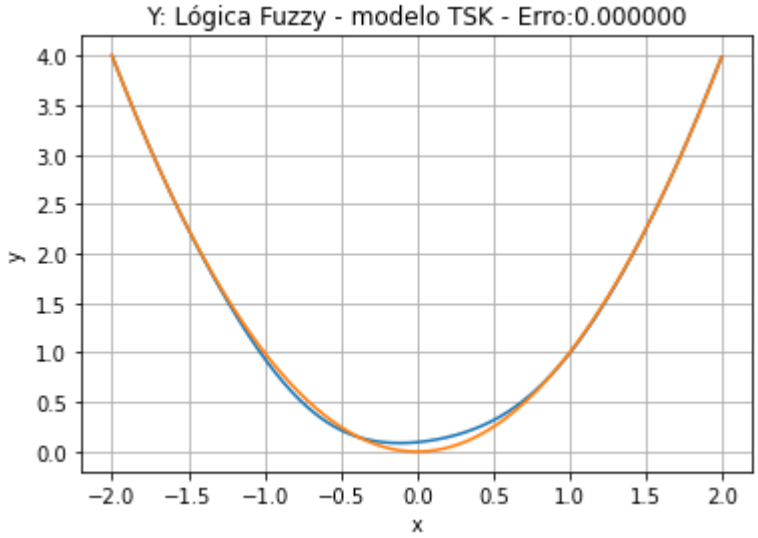
epoca: 257



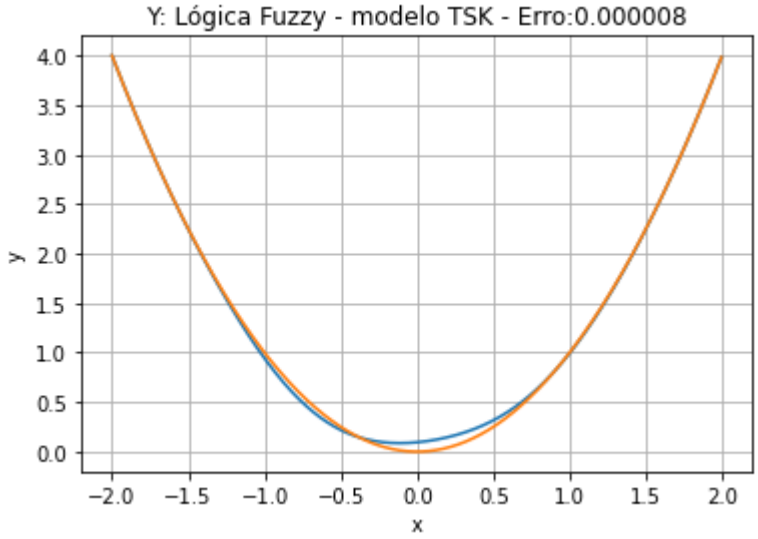
epoca: 258



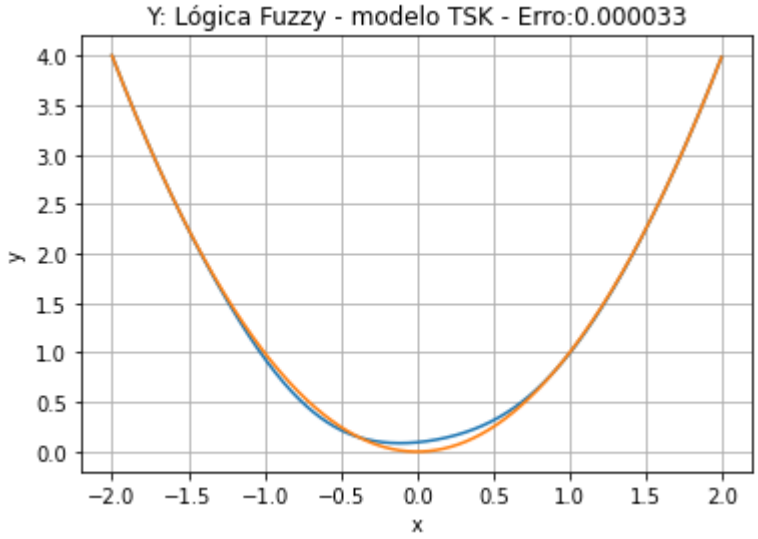
epoca: 259



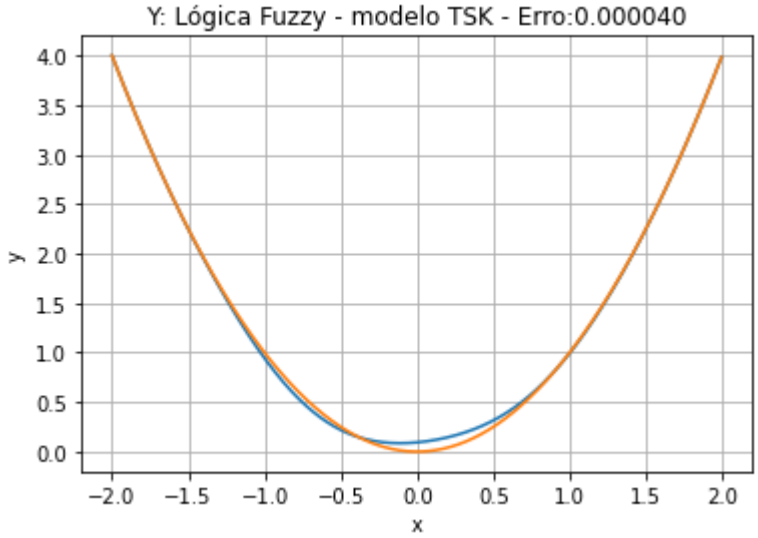
epoca: 260



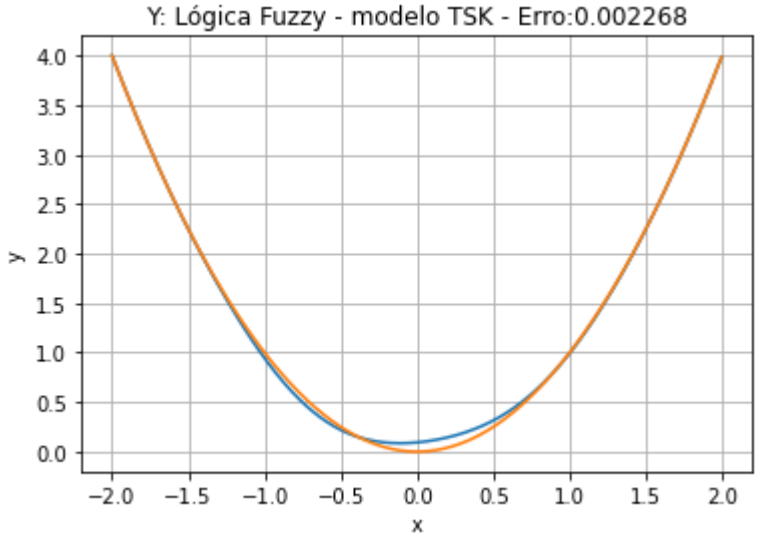
epoca: 261



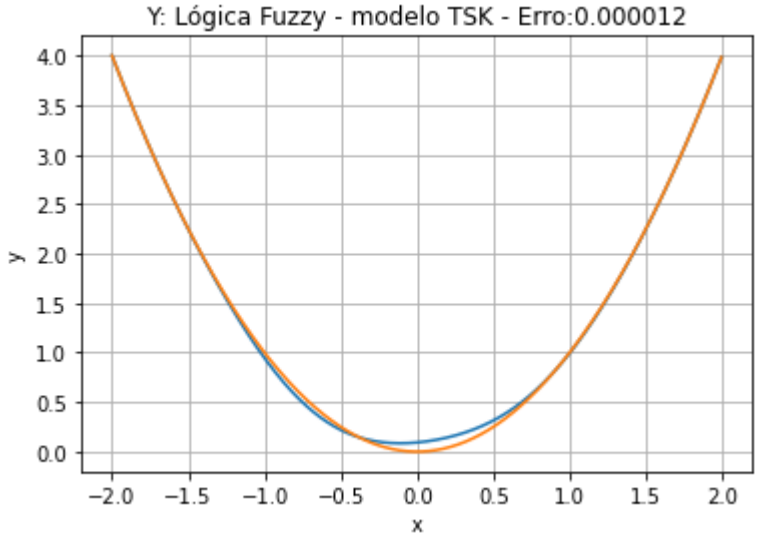
epoca: 262



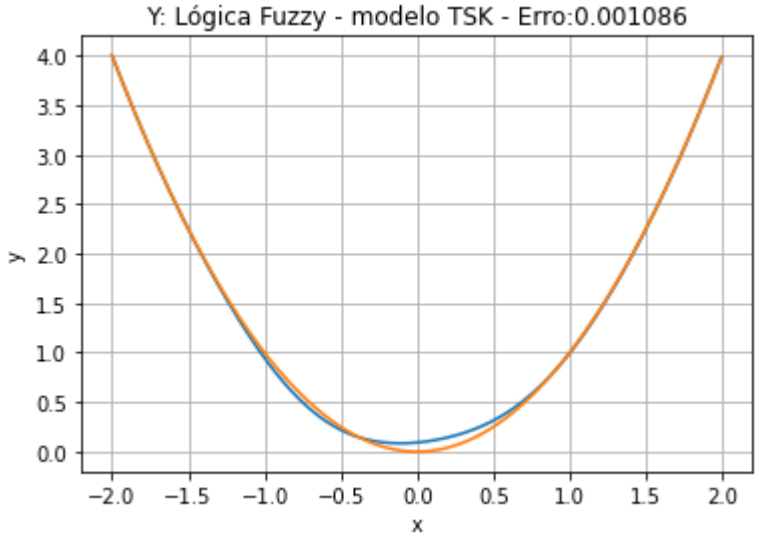
epoca: 263



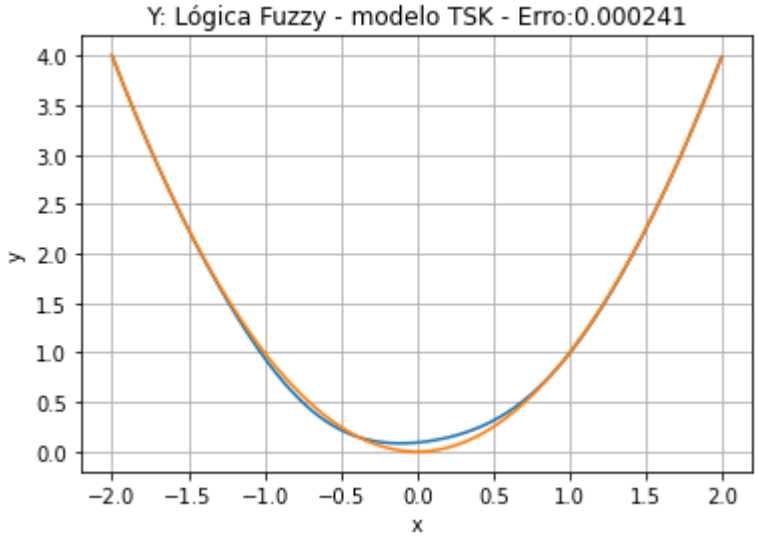
epoca: 264



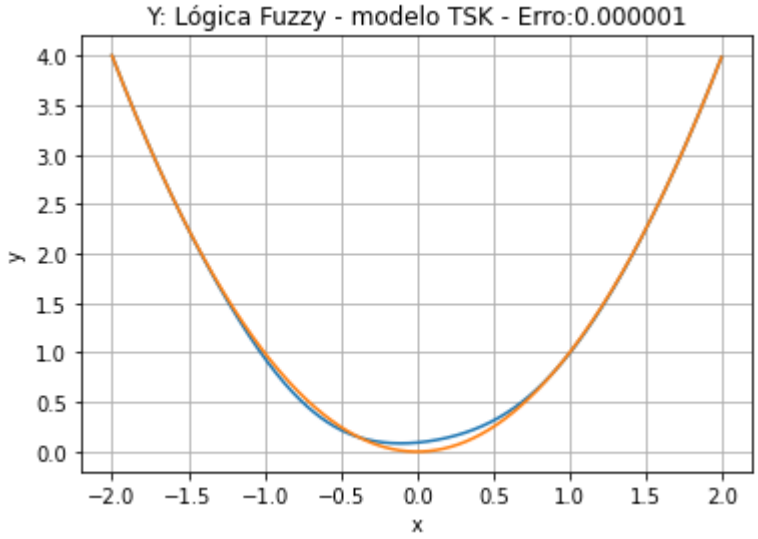
epoca: 265



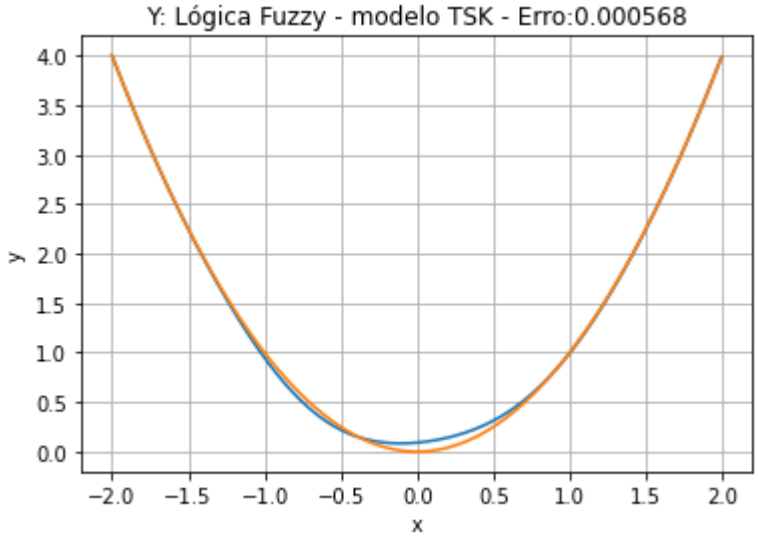
epoca: 266



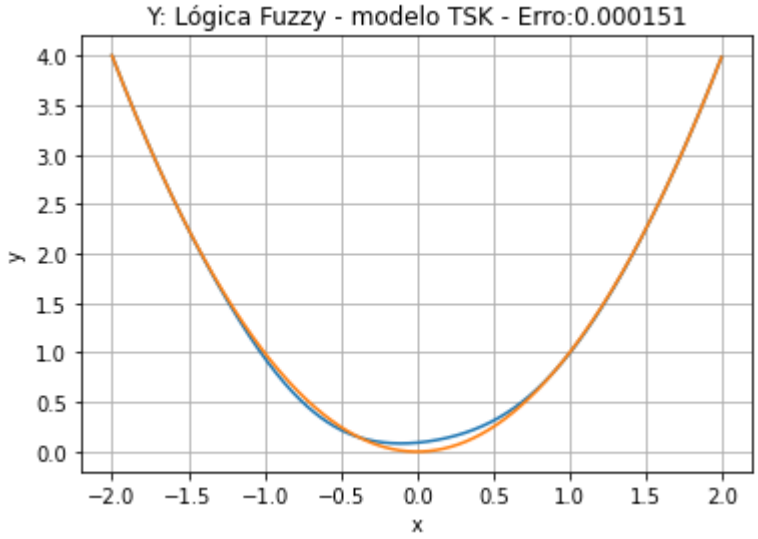
epoca: 267



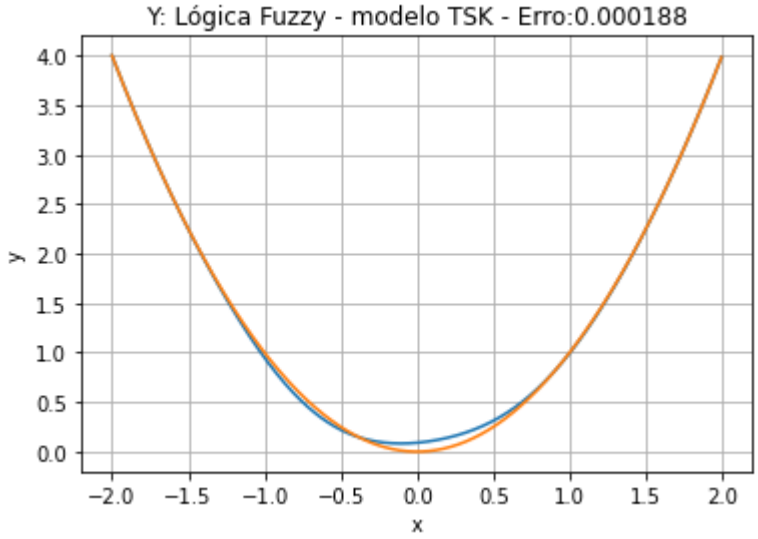
epoca: 268



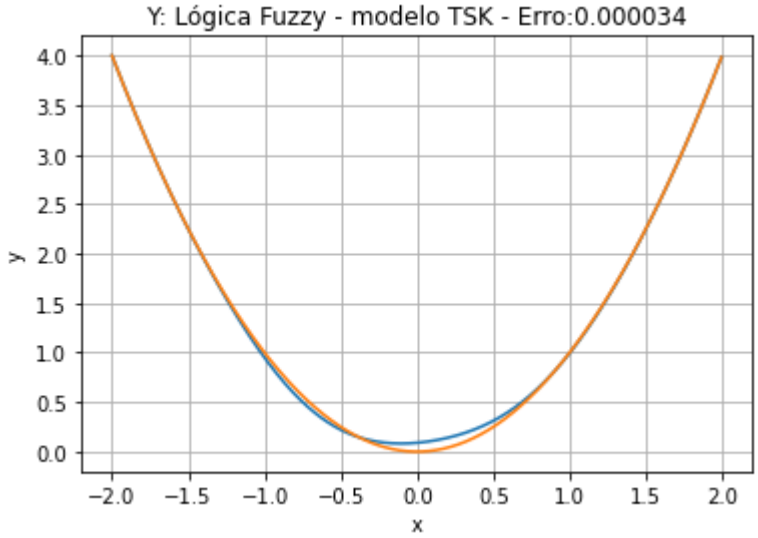
epoca: 269



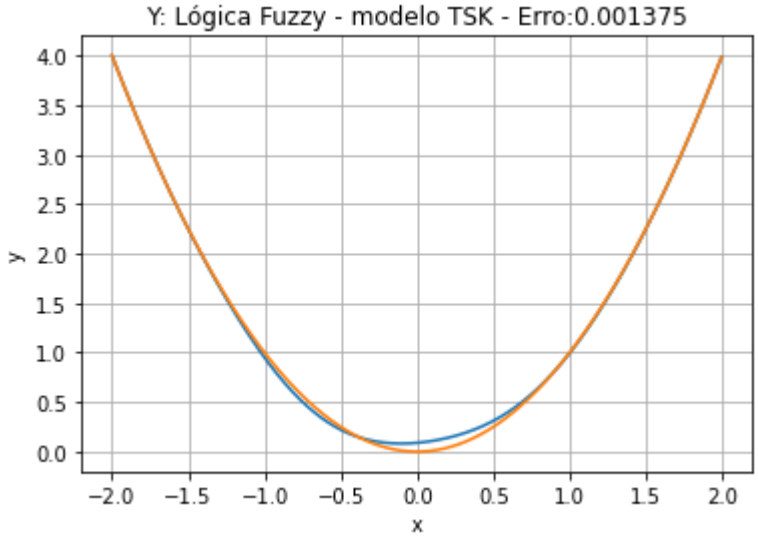
epoca: 270



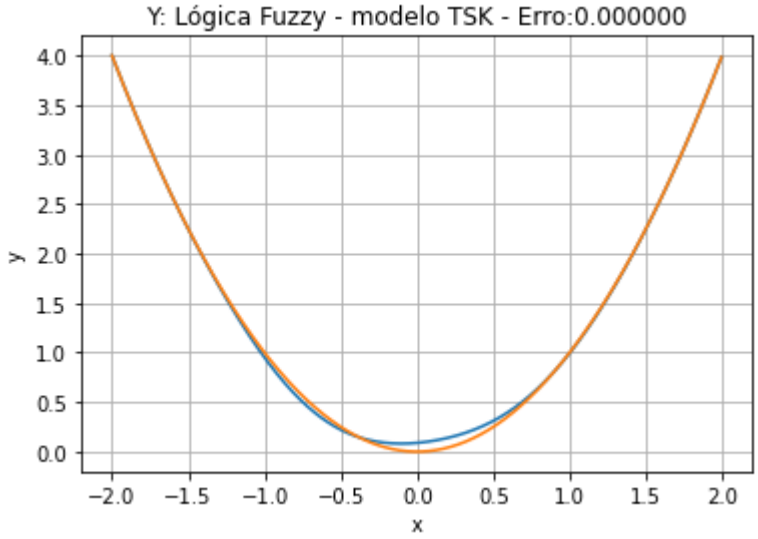
epoca: 271



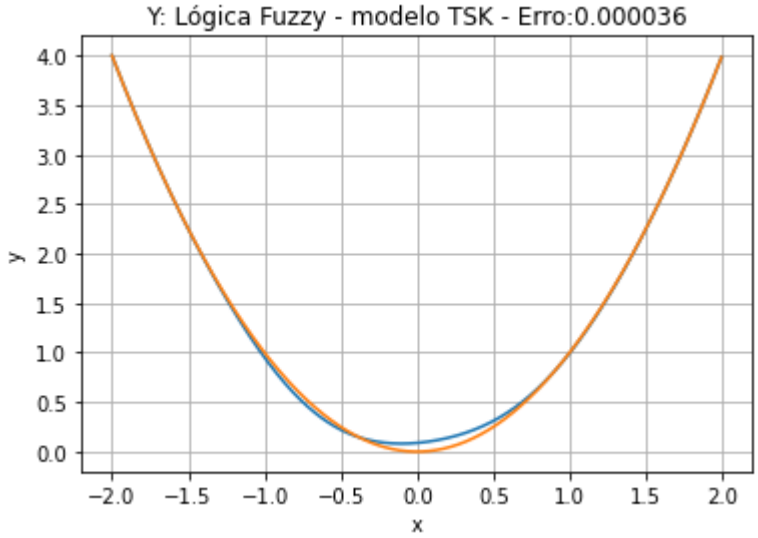
epoca: 272



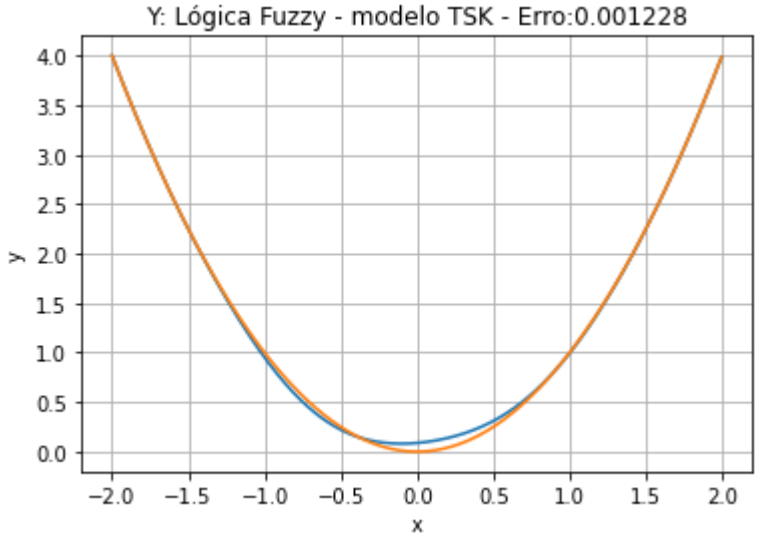
epoca: 273



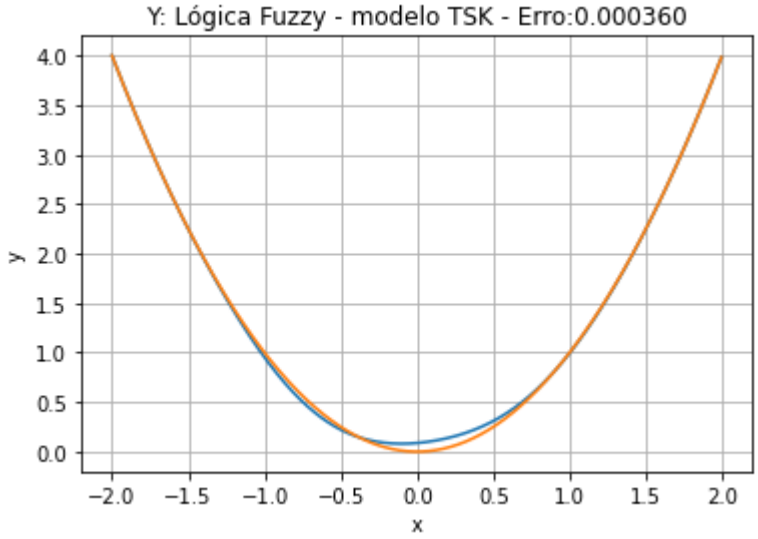
epoca: 274



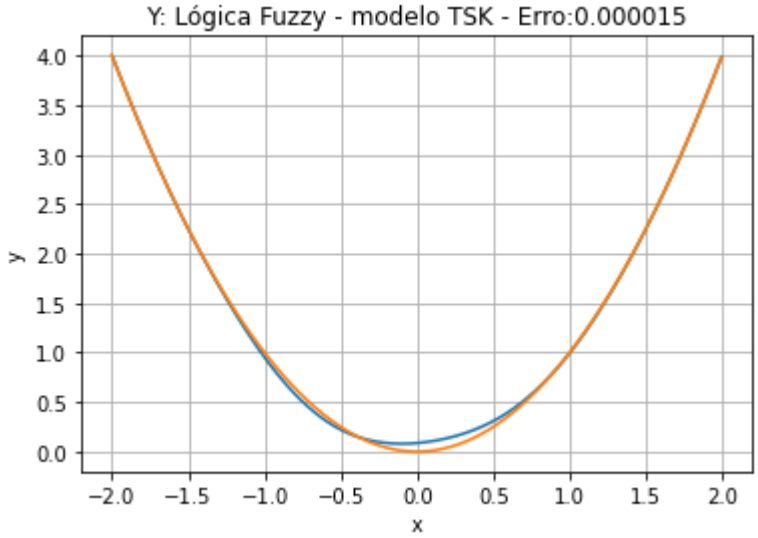
epoca: 275



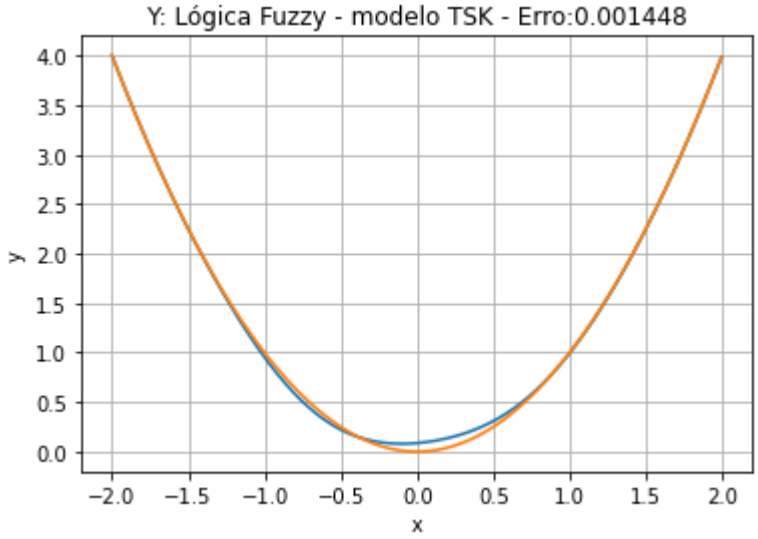
epoca: 276



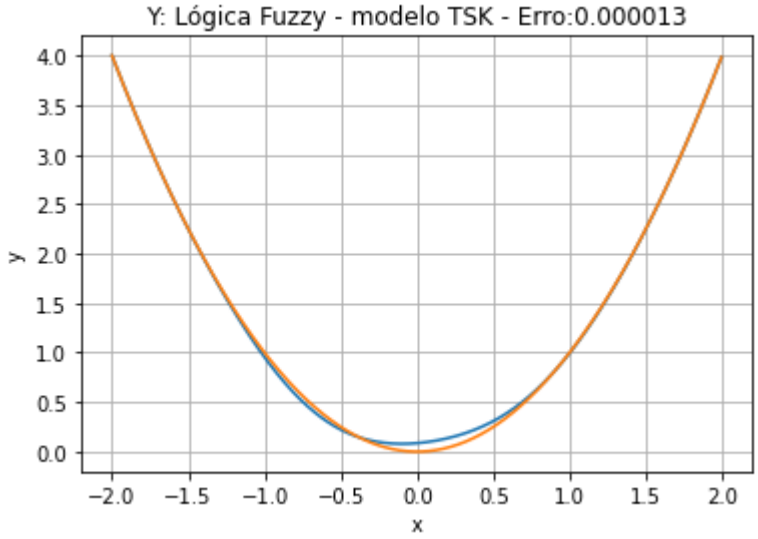
epoca: 277



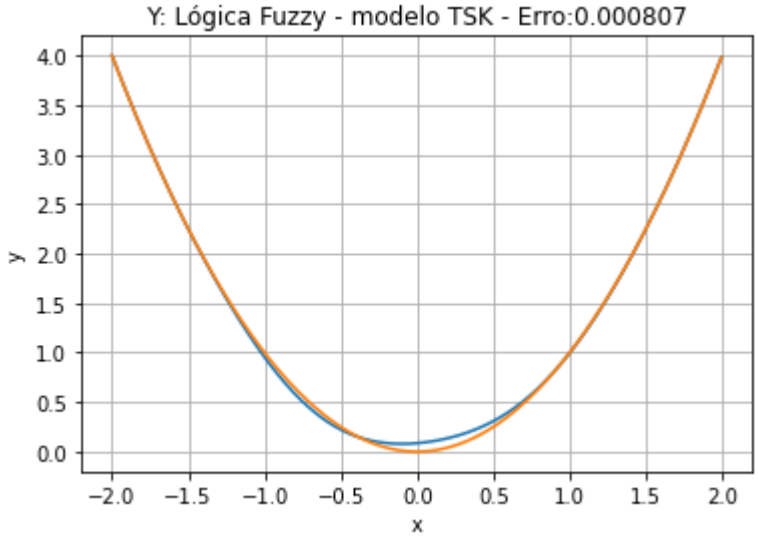
epoca: 278



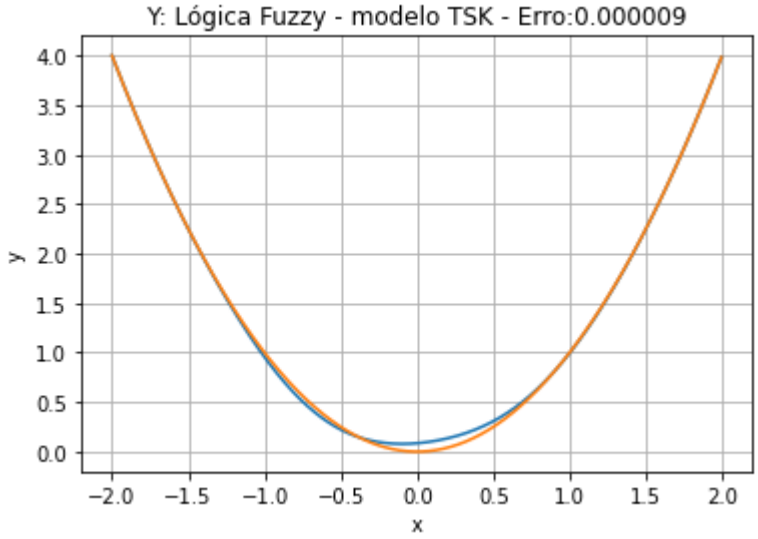
epoca: 279



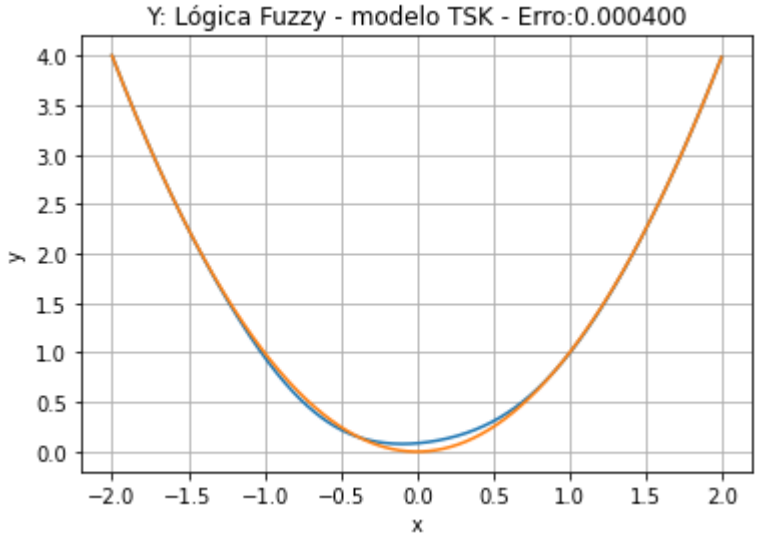
epoca: 280



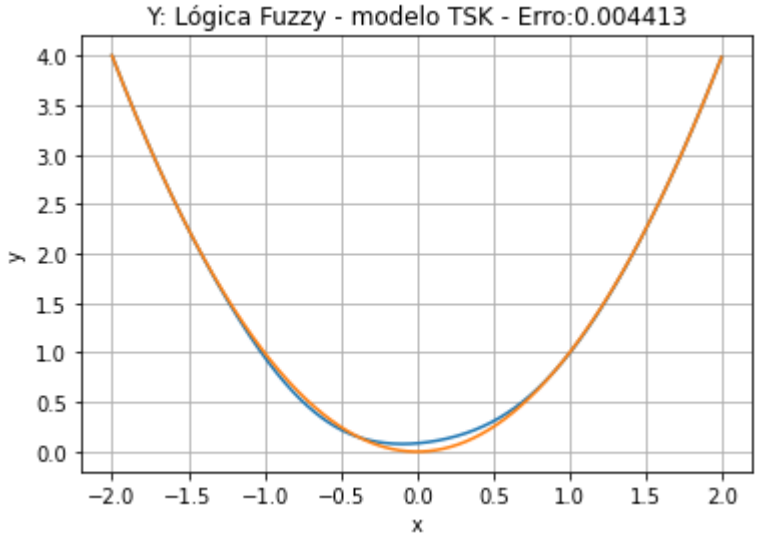
epoca: 281



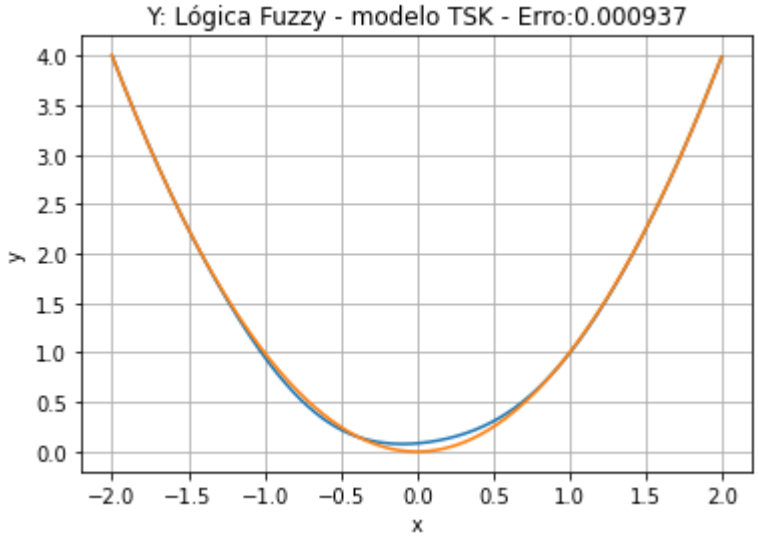
epoca: 282



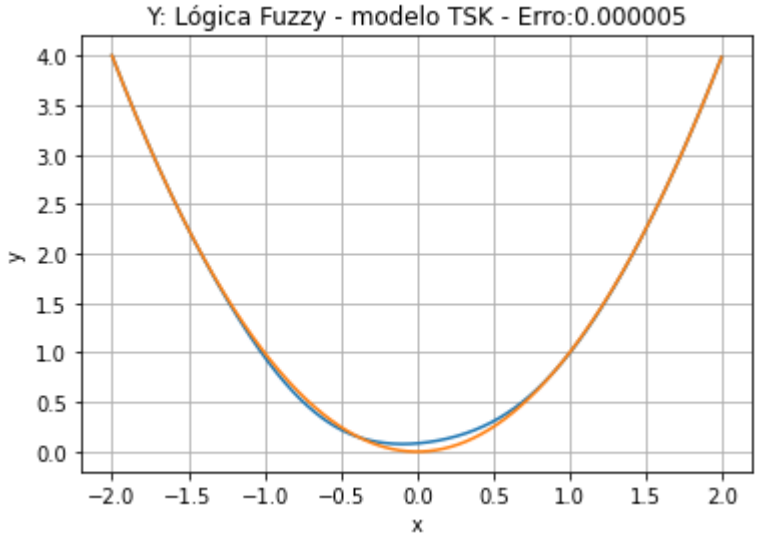
epoca: 283



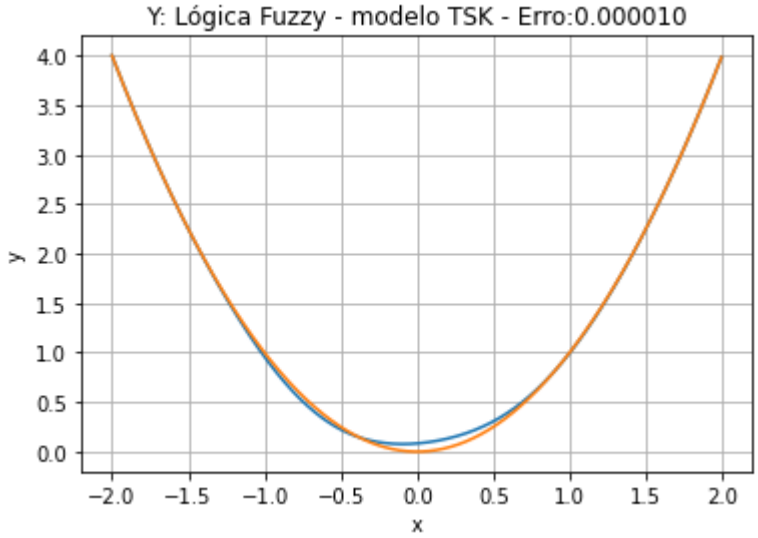
epoca: 284



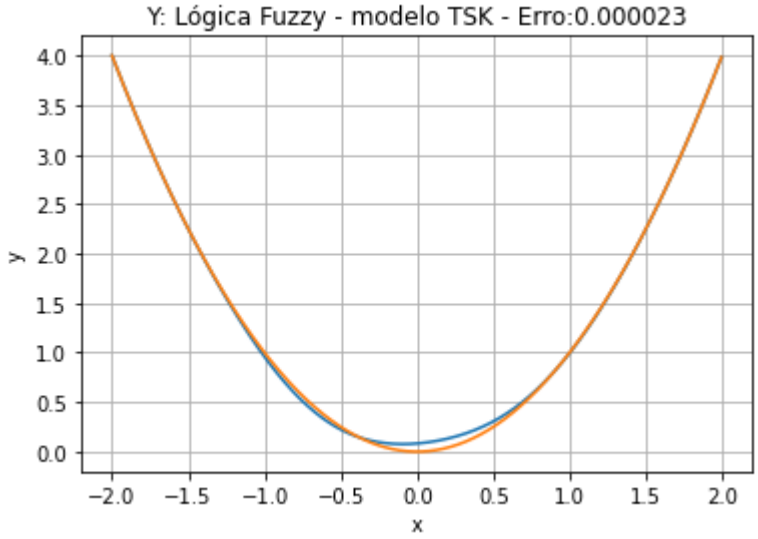
epoca: 285



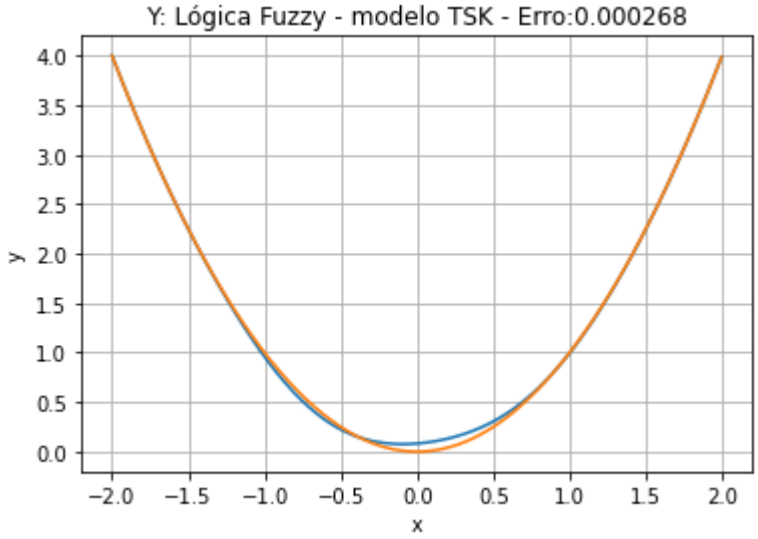
epoca: 286



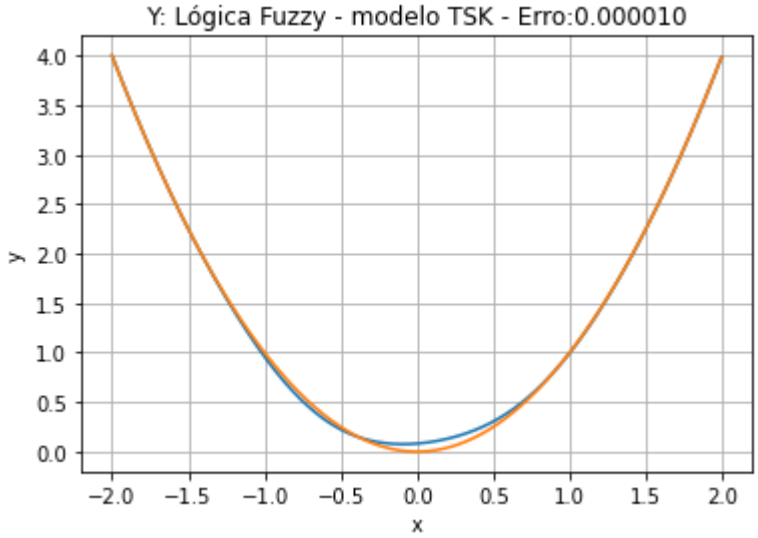
epoca: 287



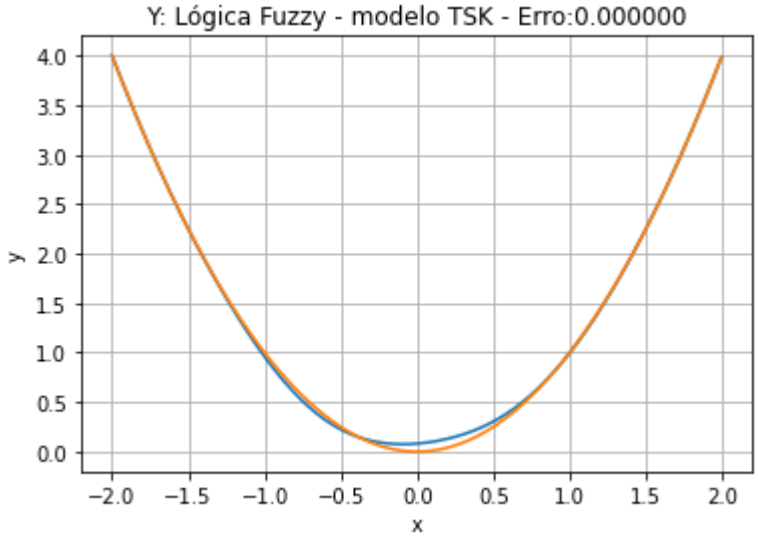
epoca: 288



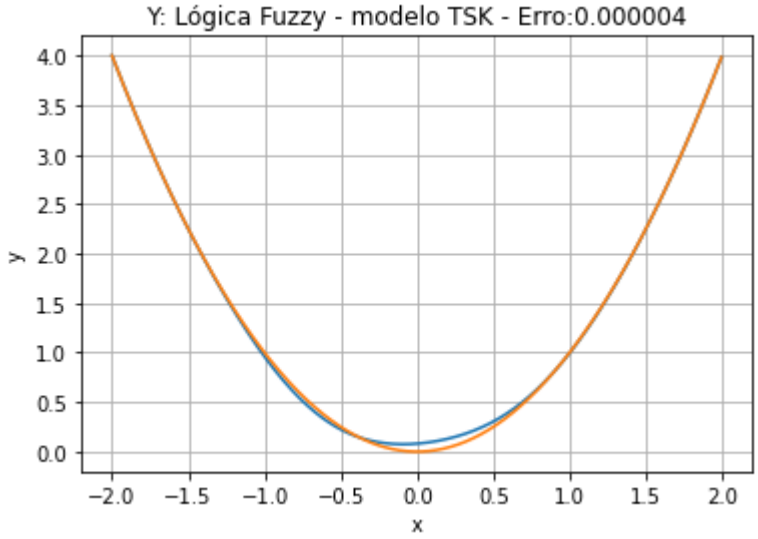
epoca: 289



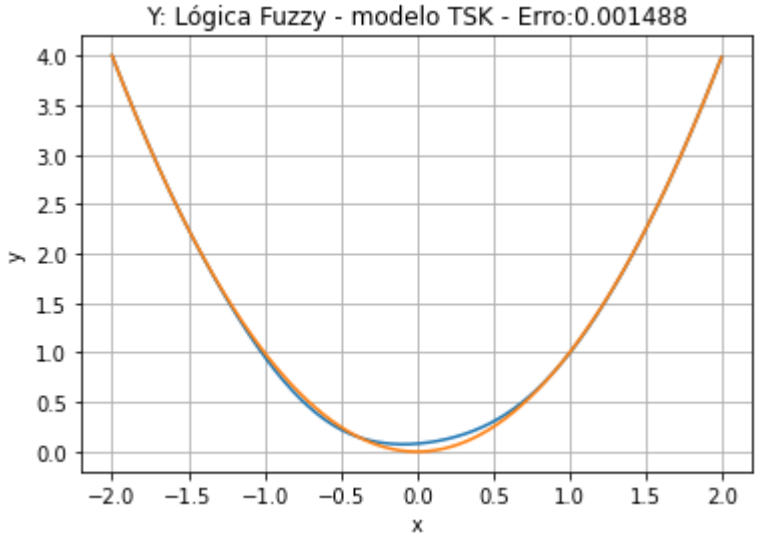
epoca: 290



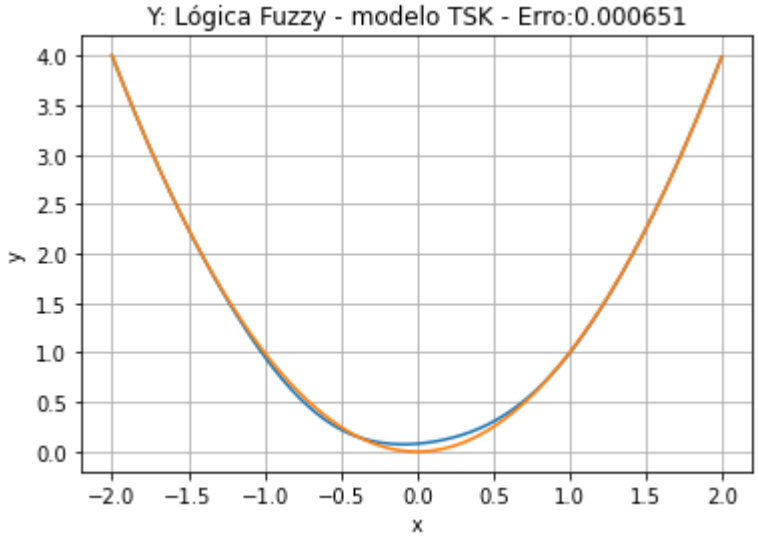
epoca: 291



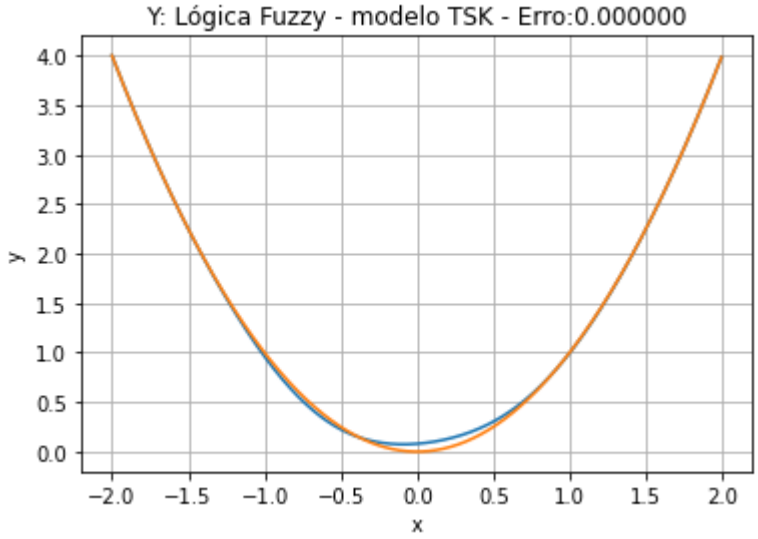
epoca: 292



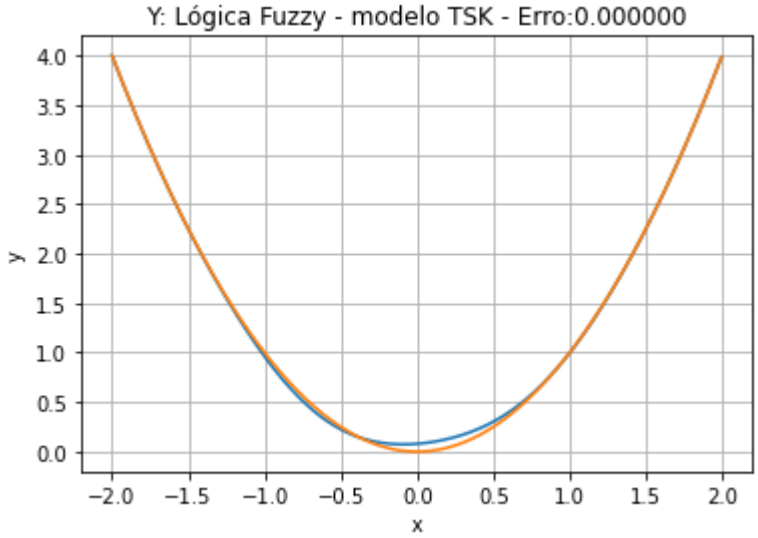
epoca: 293



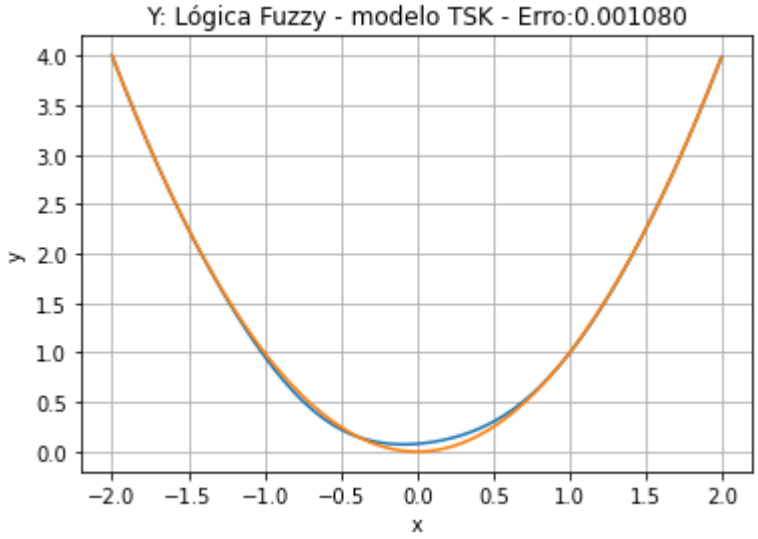
epoca: 294



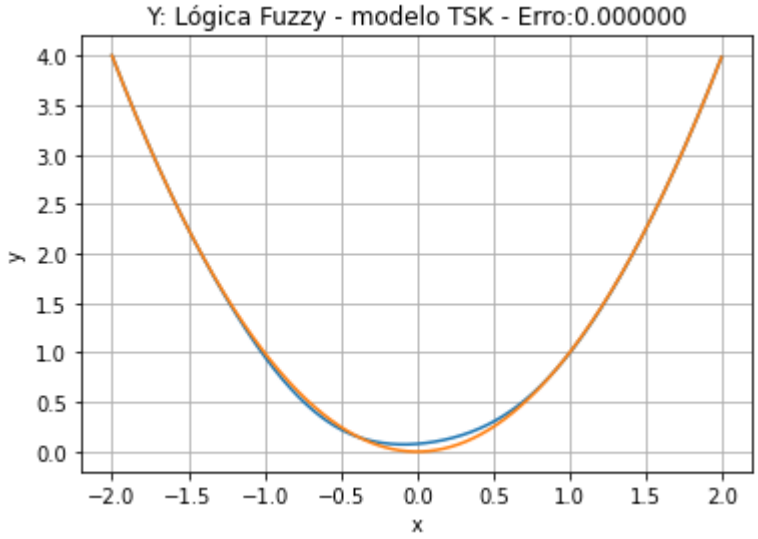
epoca: 295



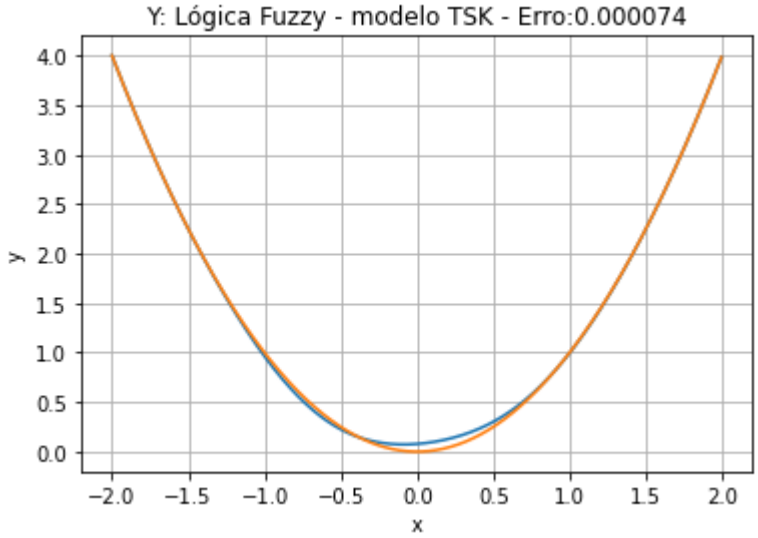
epoca: 296



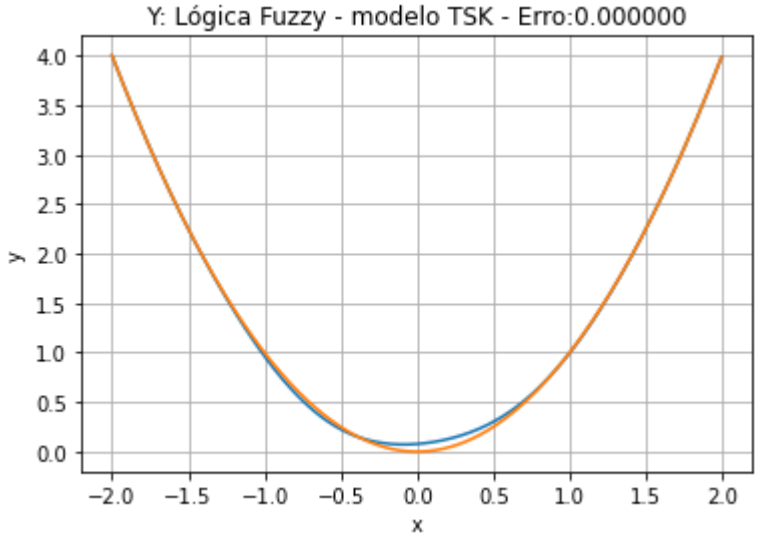
epoca: 297



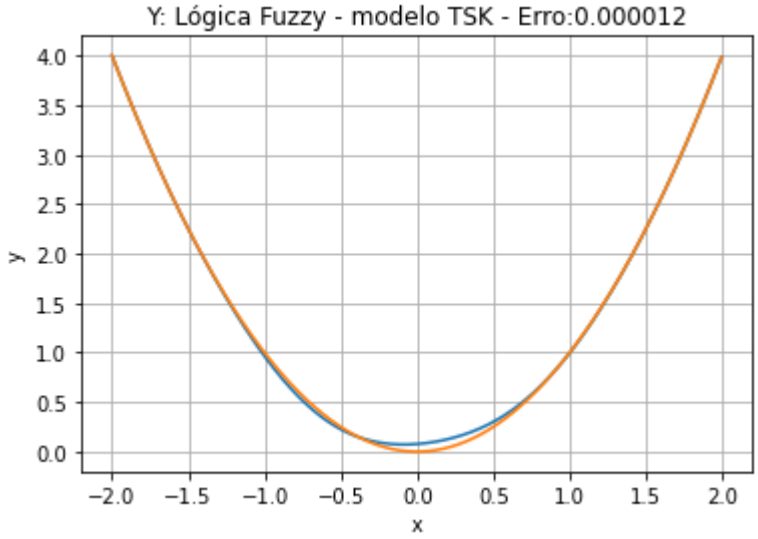
epoca: 298



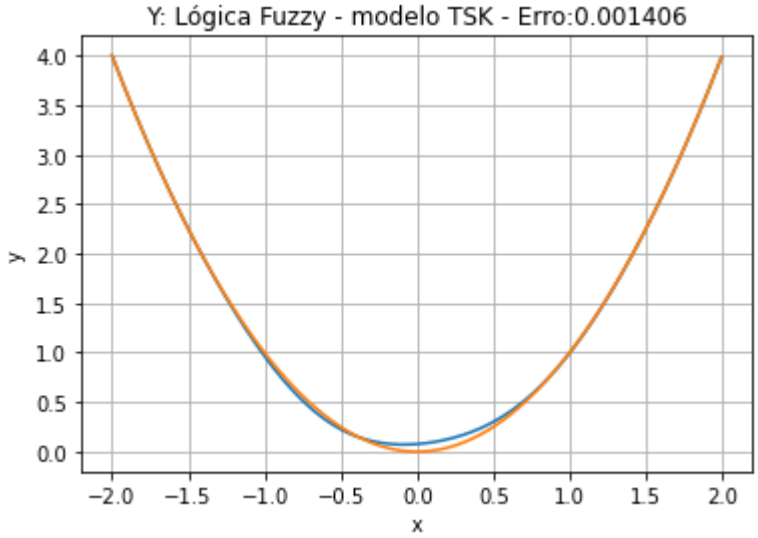
epoca: 299



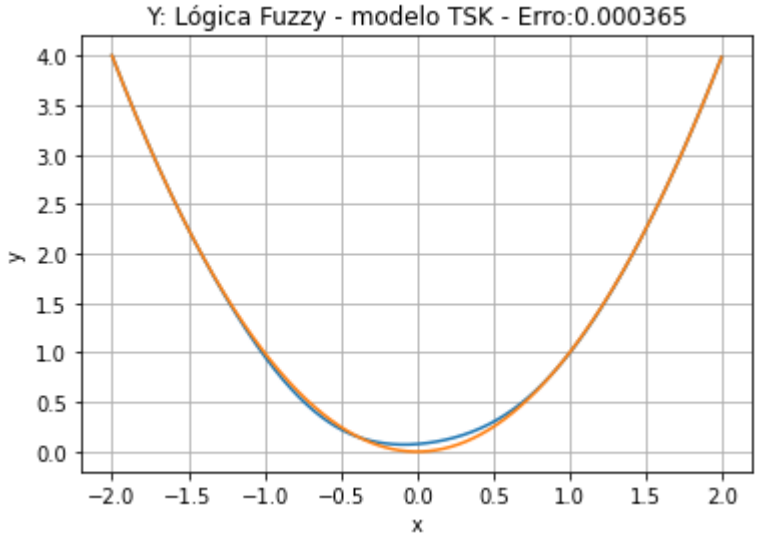

```
epoca: 300
```



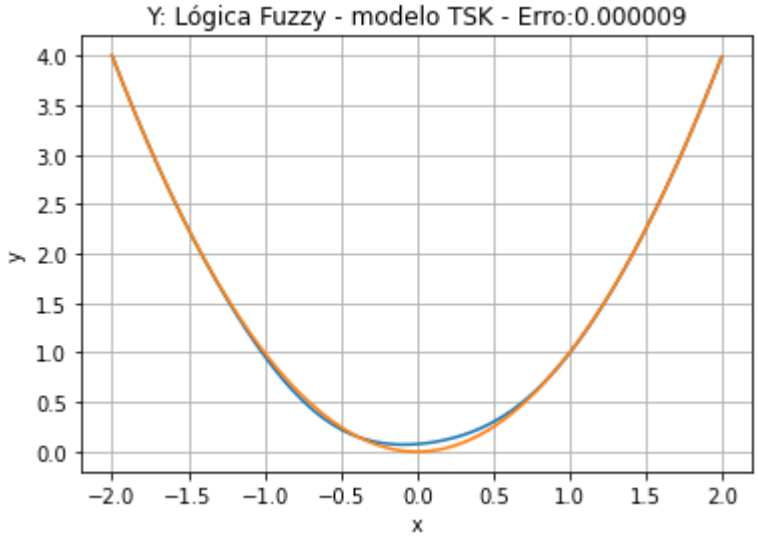

```
epoca: 301
```



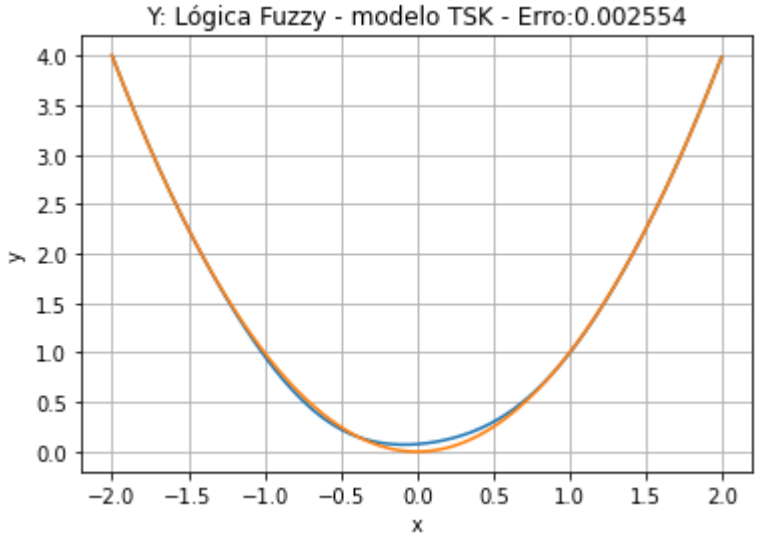

```
epoca: 302
```



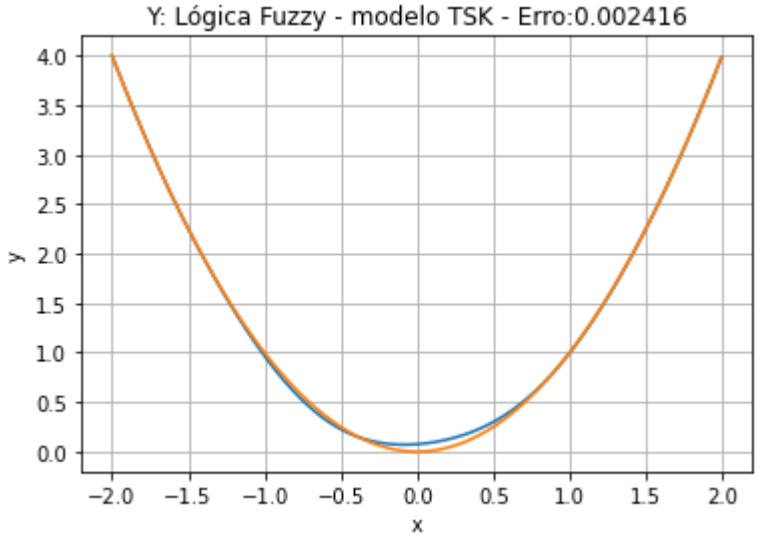
epoca: 303



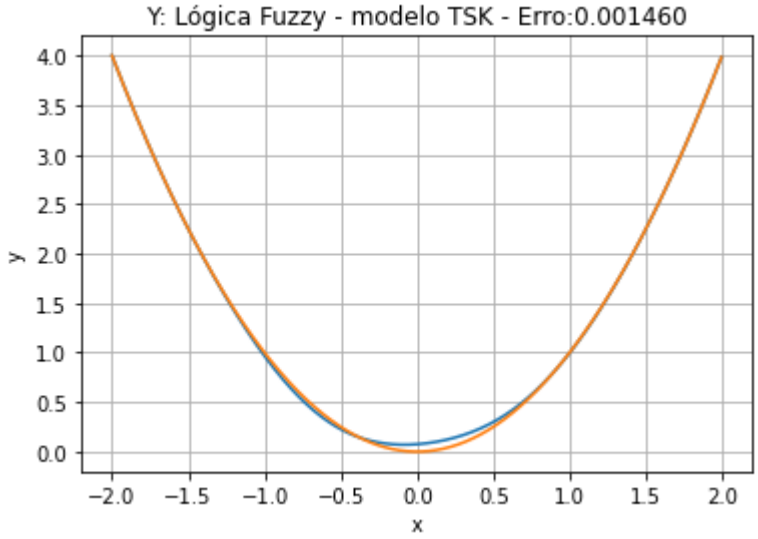
epoca: 304



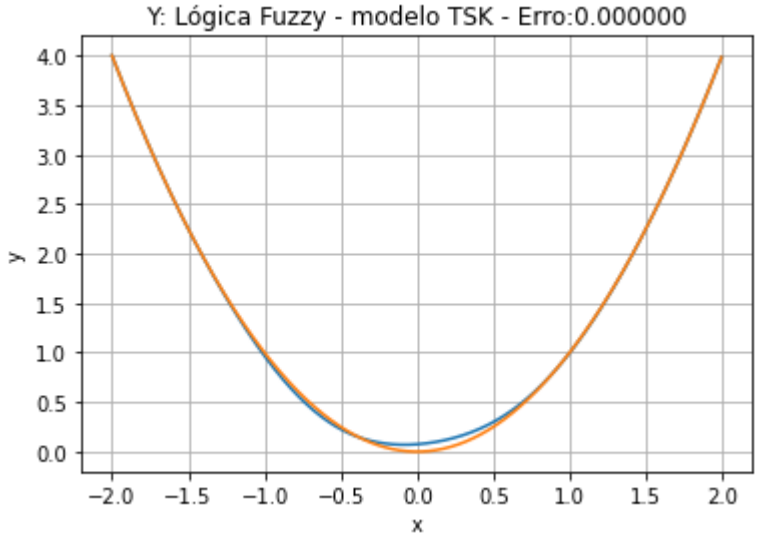
epoca: 305



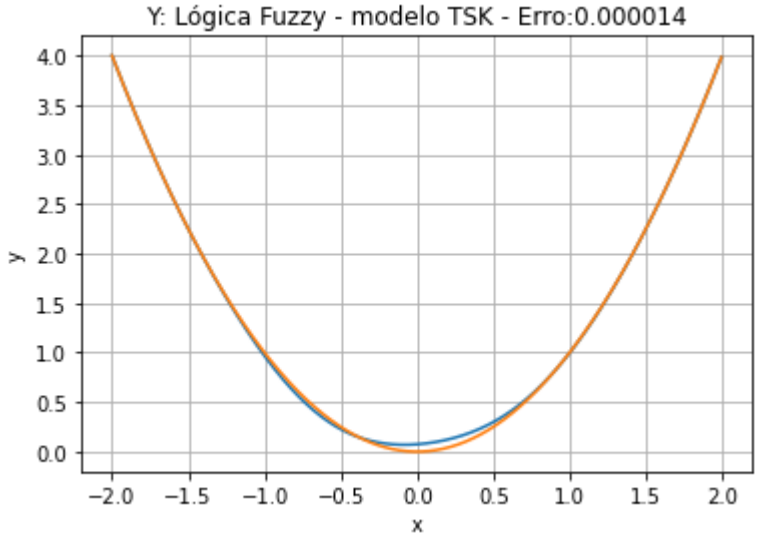

```
epoca: 306
```



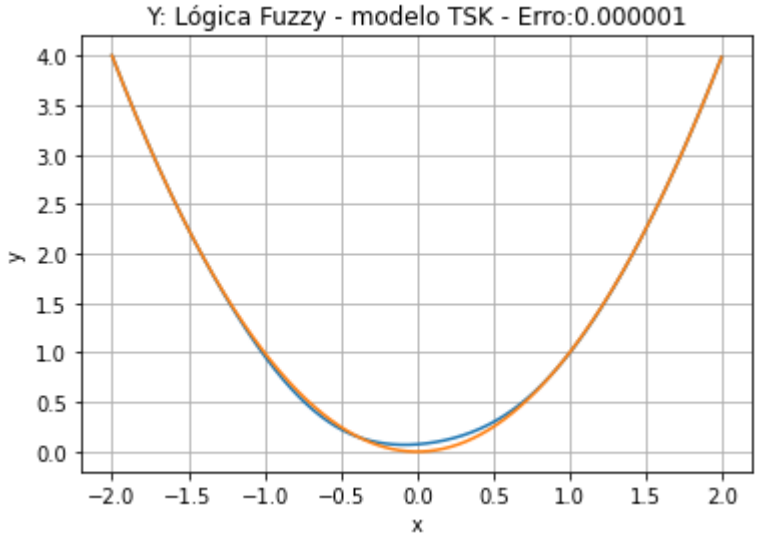

```
epoca: 307
```



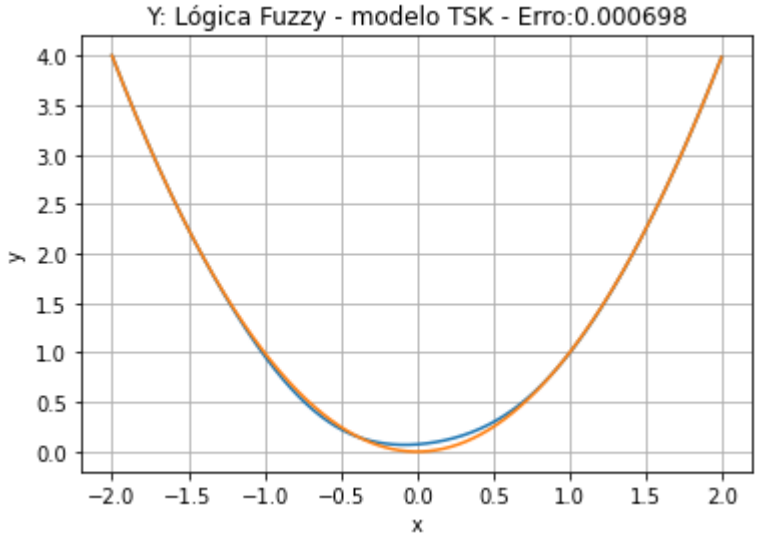

```
epoca: 308
```



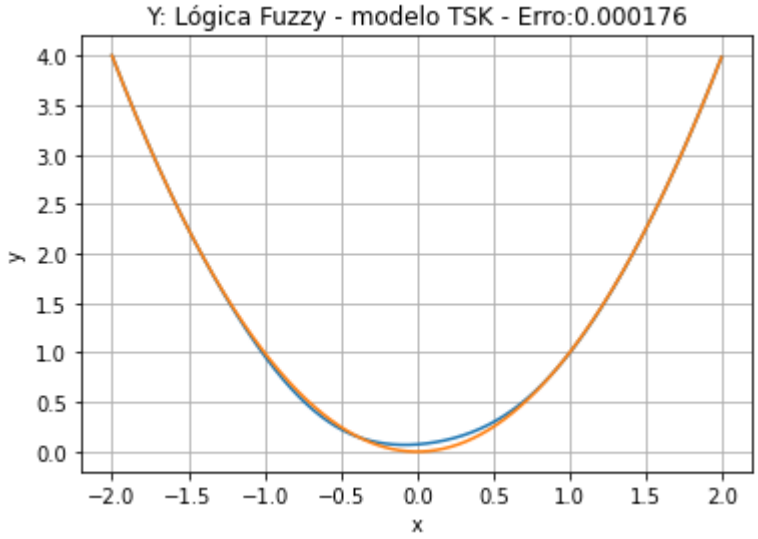
epoca: 309



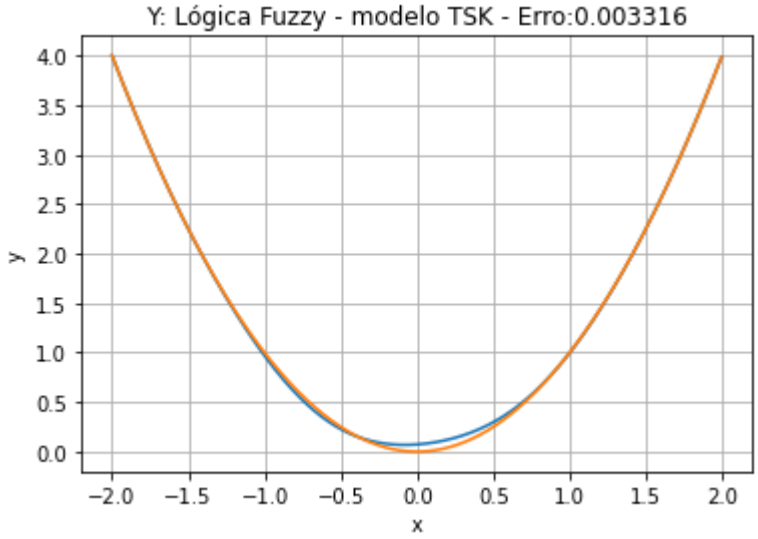
epoca: 310



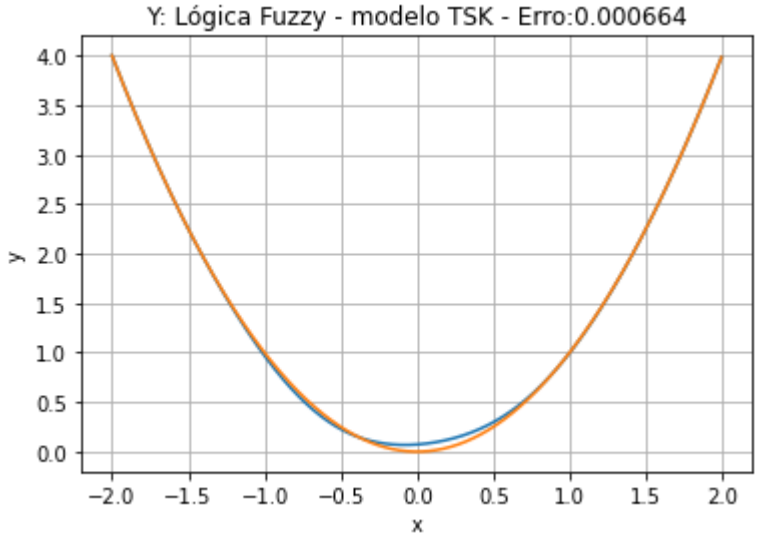
epoca: 311



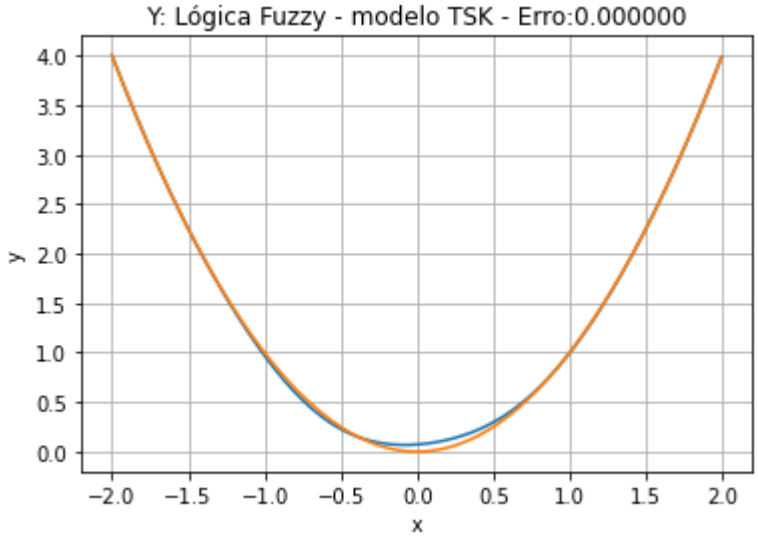
epoca: 312



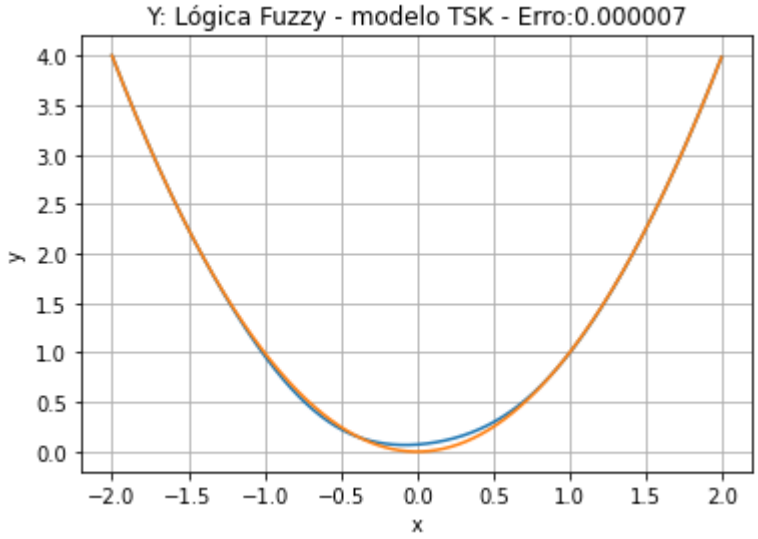
epoca: 313



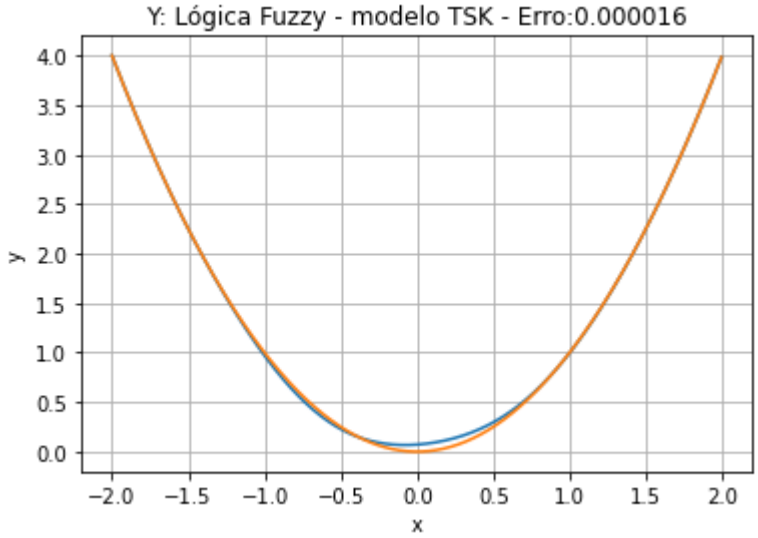
epoca: 314



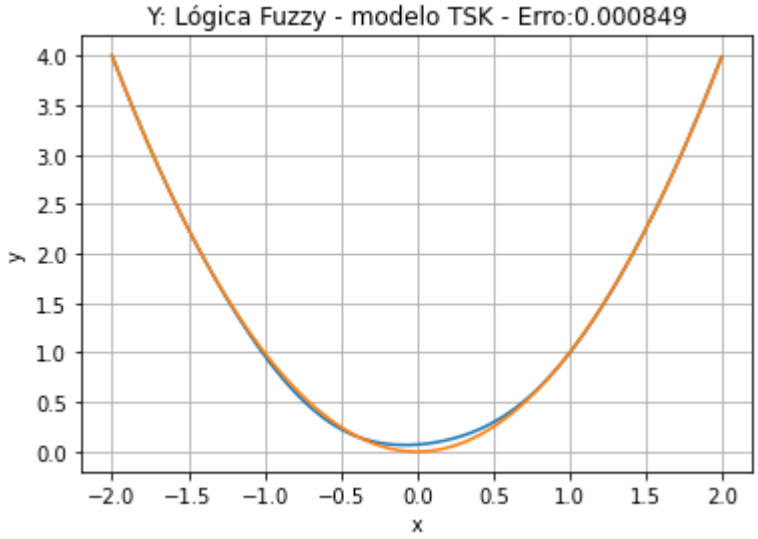
epoca: 315



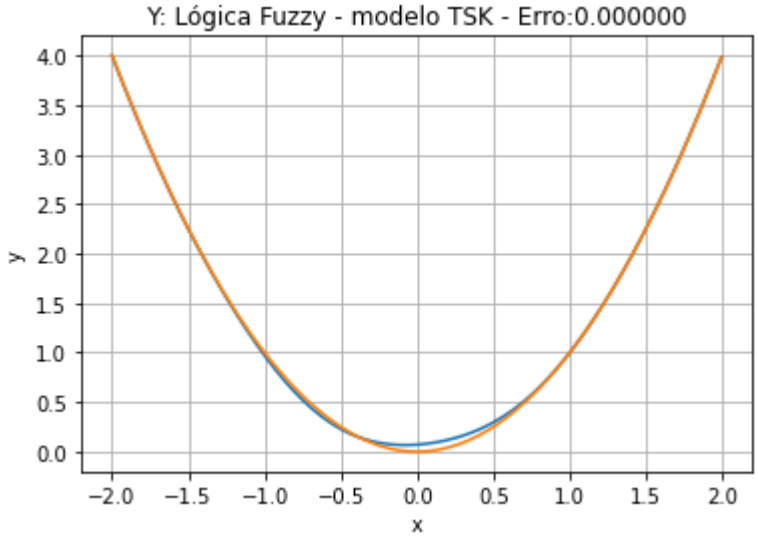
epoca: 316



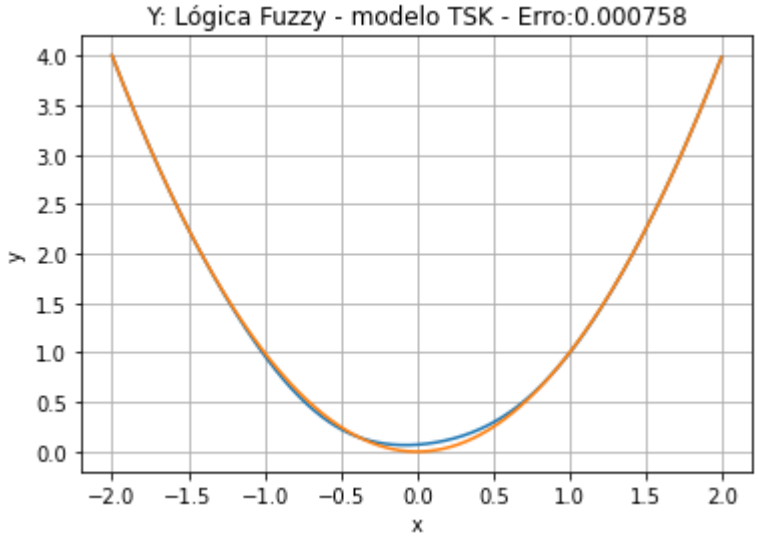
epoca: 317



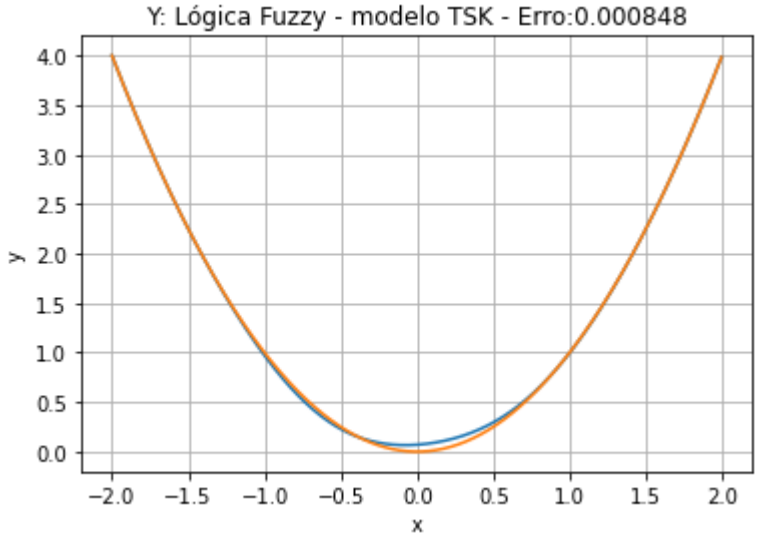
epoca: 318



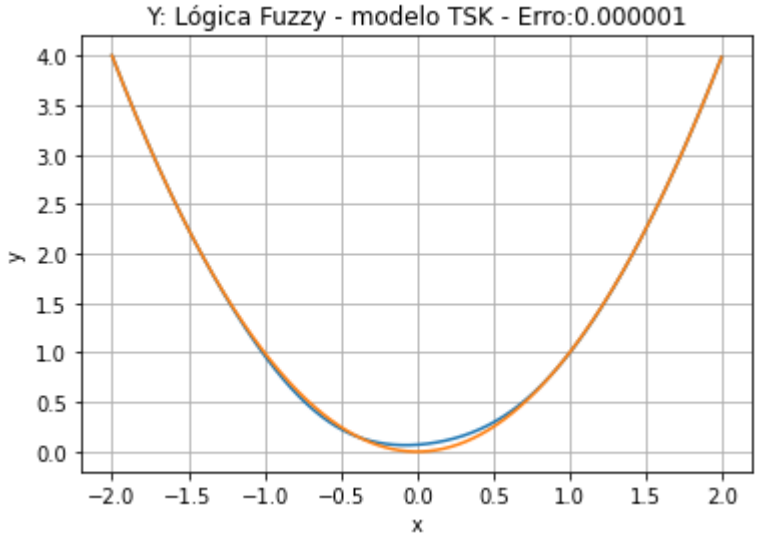
epoca: 319



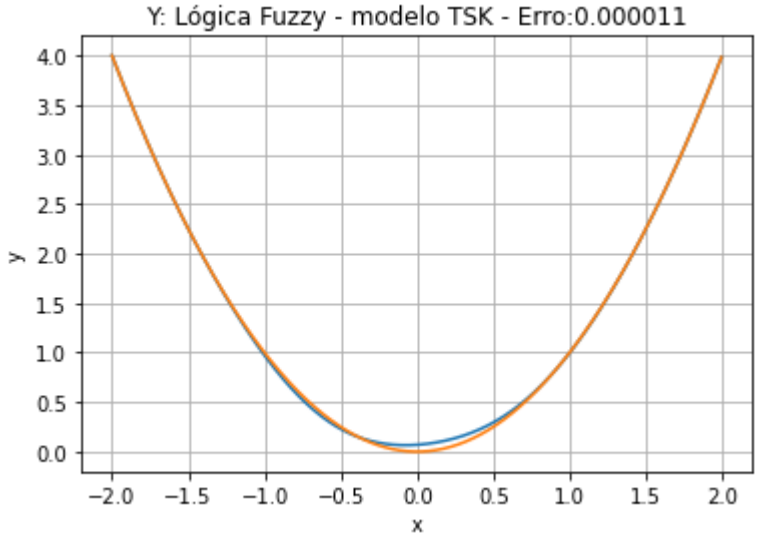
epoca: 320



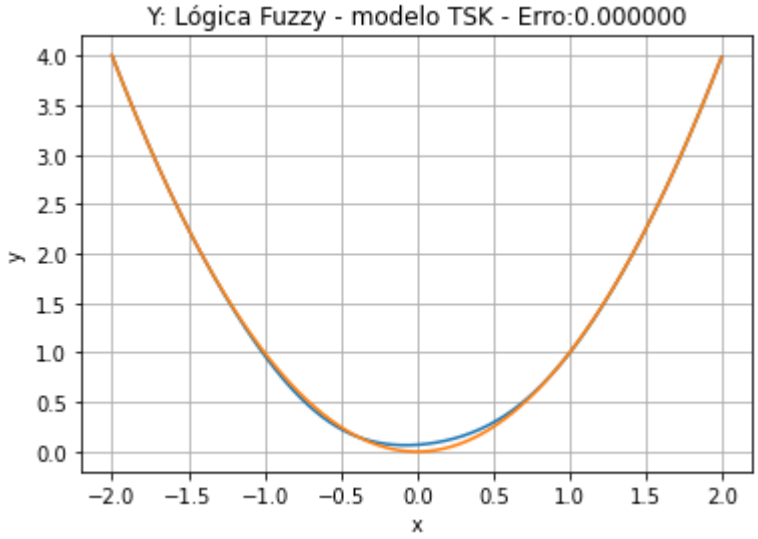
epoca: 321



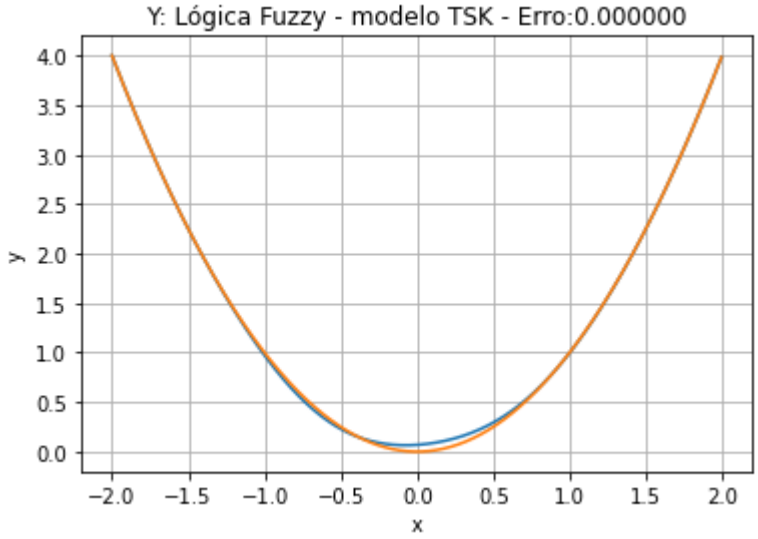
epoca: 322



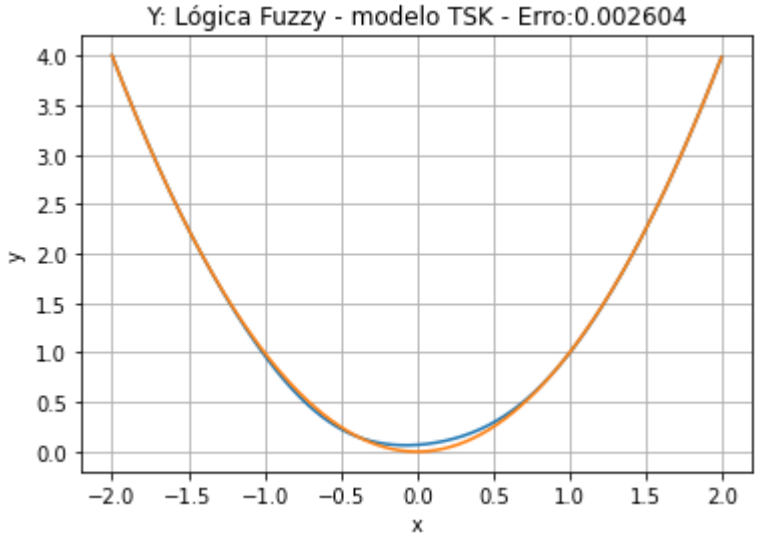
epoca: 323



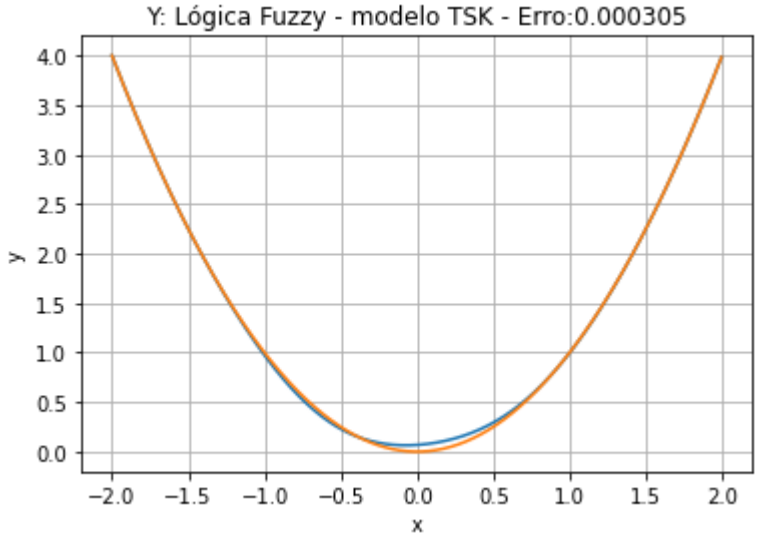
epoca: 324



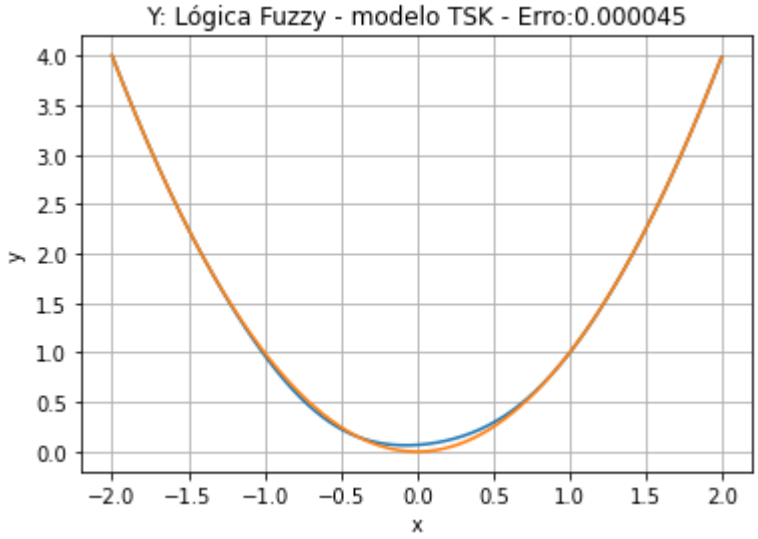
epoca: 325



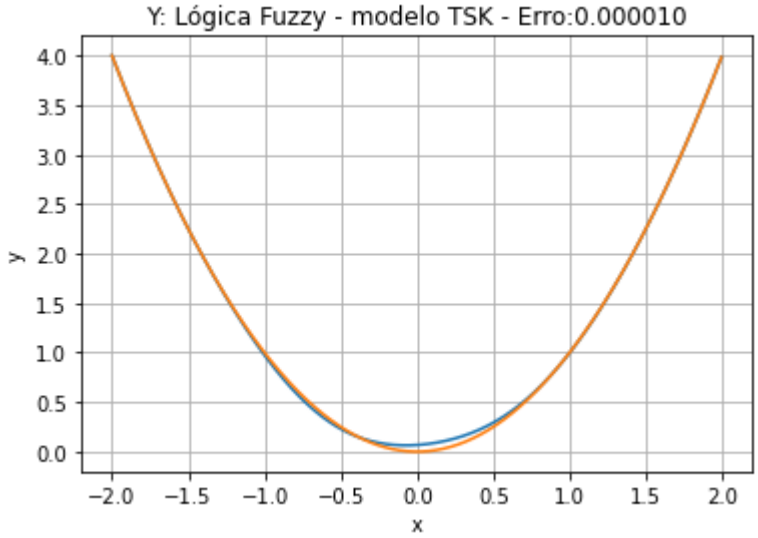
epoca: 326



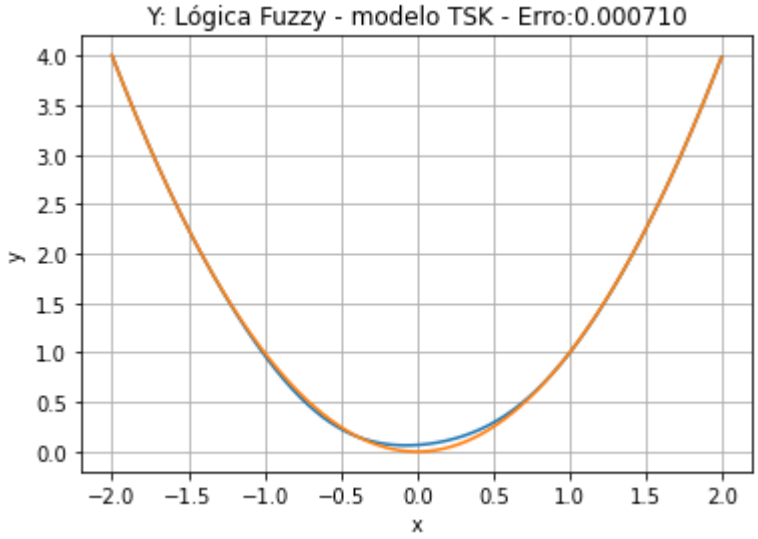
epoca: 327



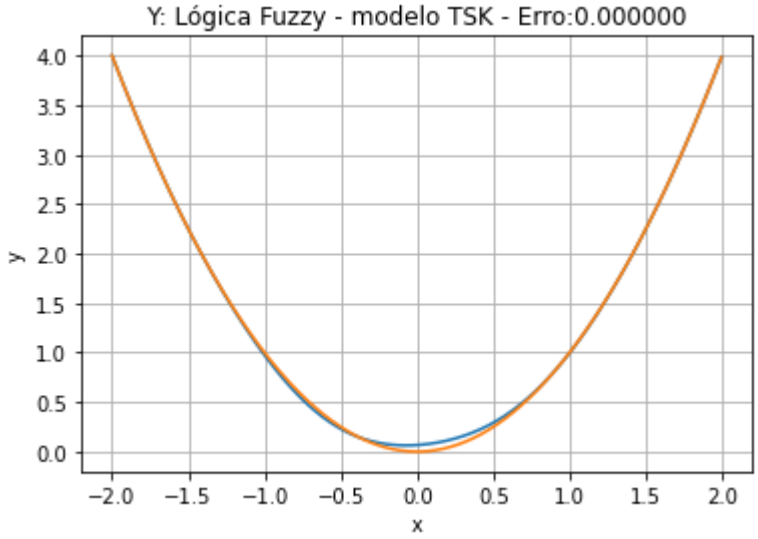
epoca: 328



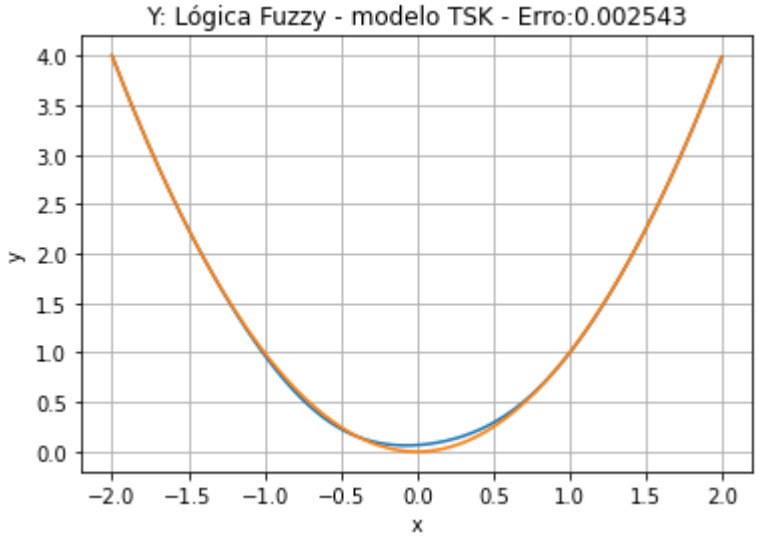
epoca: 329



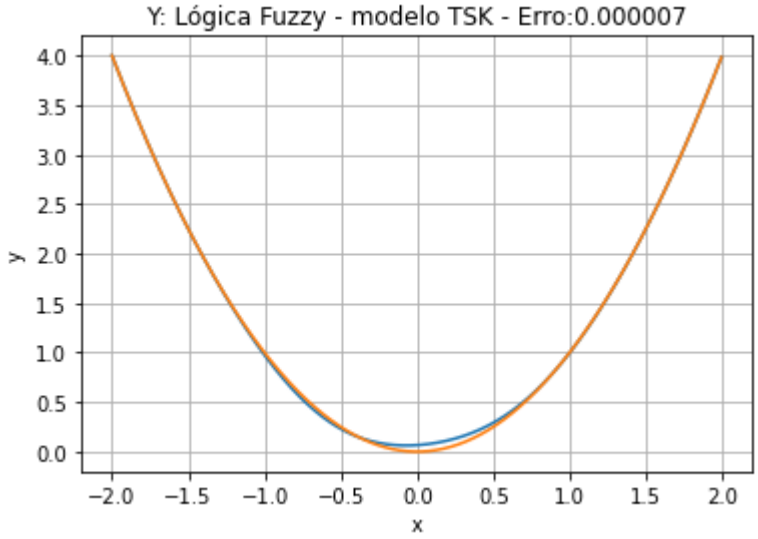
epoca: 330



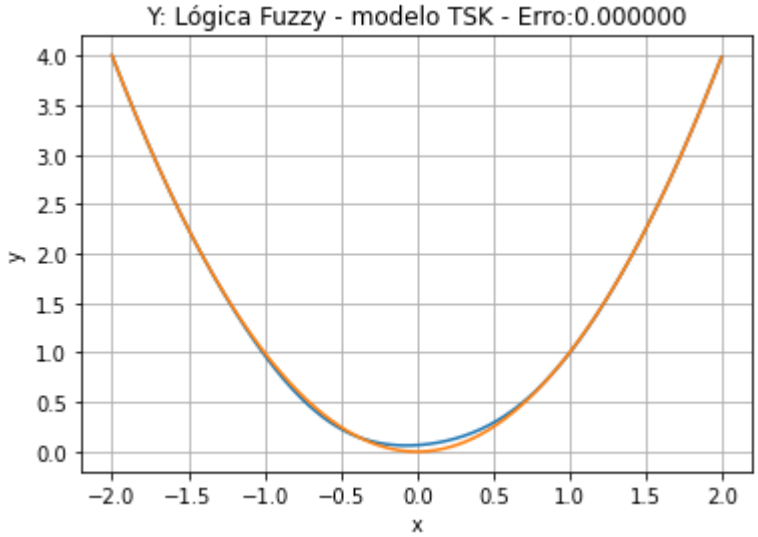
epoca: 331



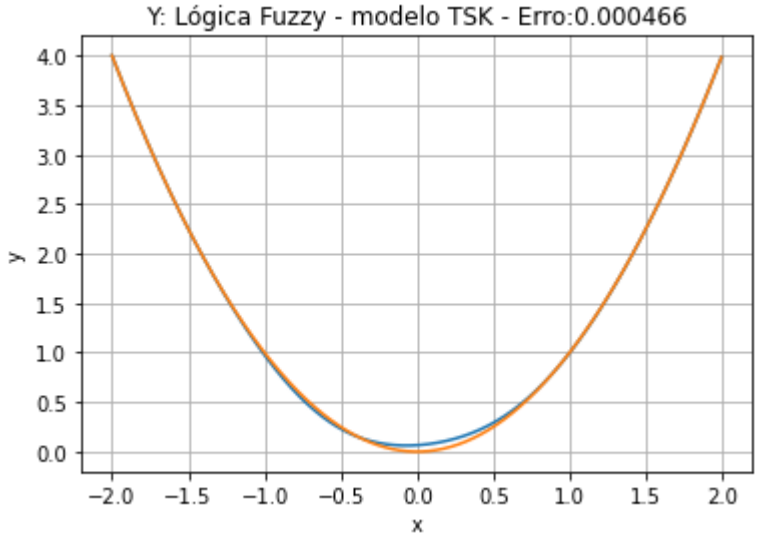
epoca: 332



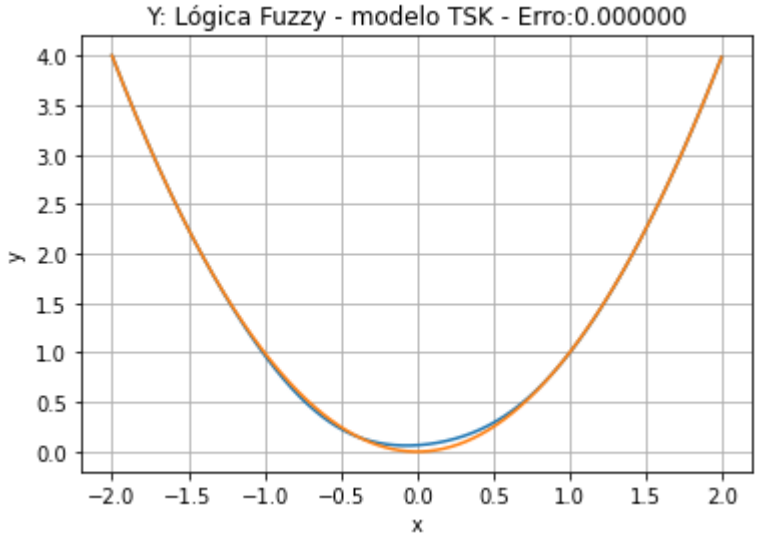
epoca: 333



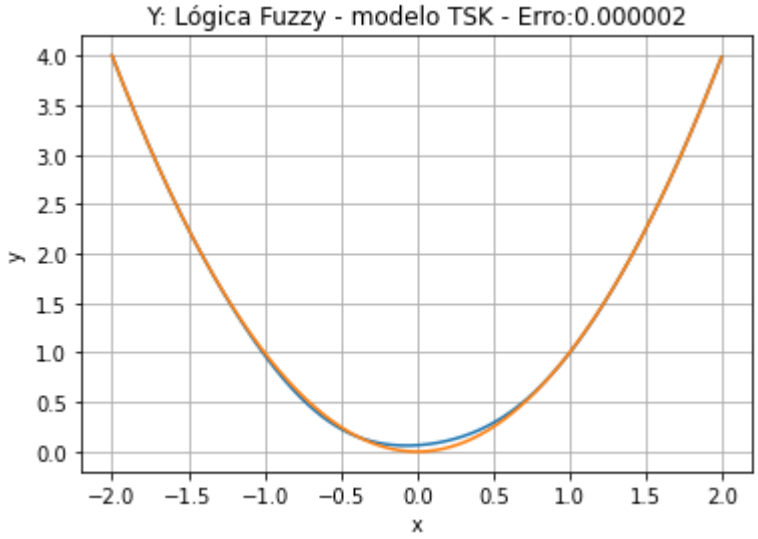
epoca: 334



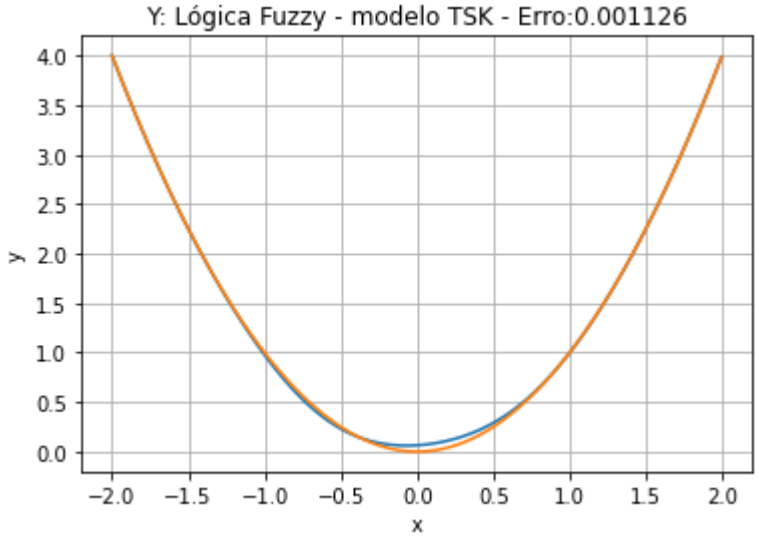
epoca: 335



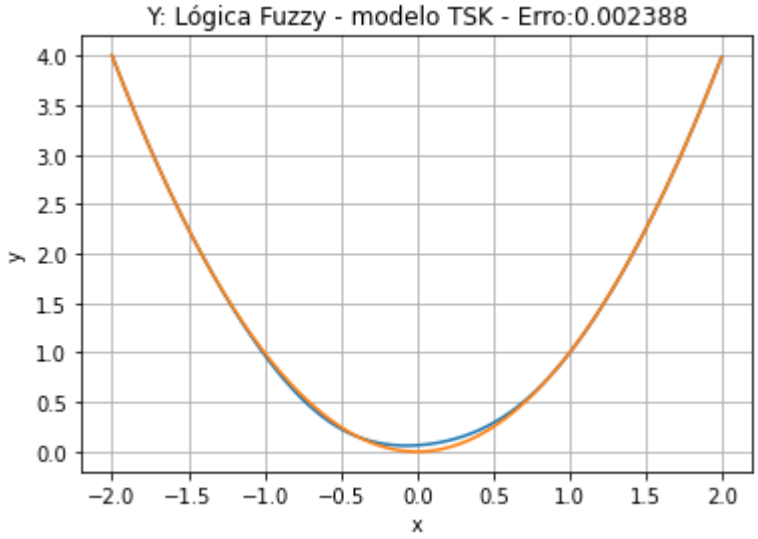
epoca: 336



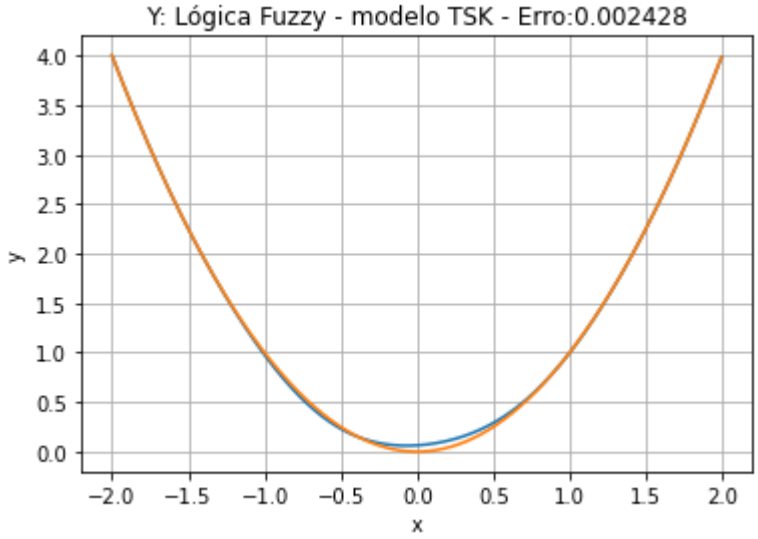
epoca: 337



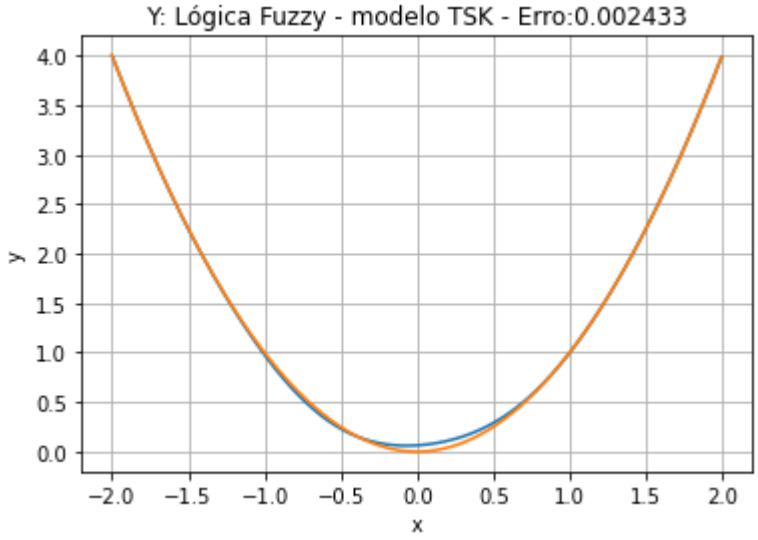
epoca: 338



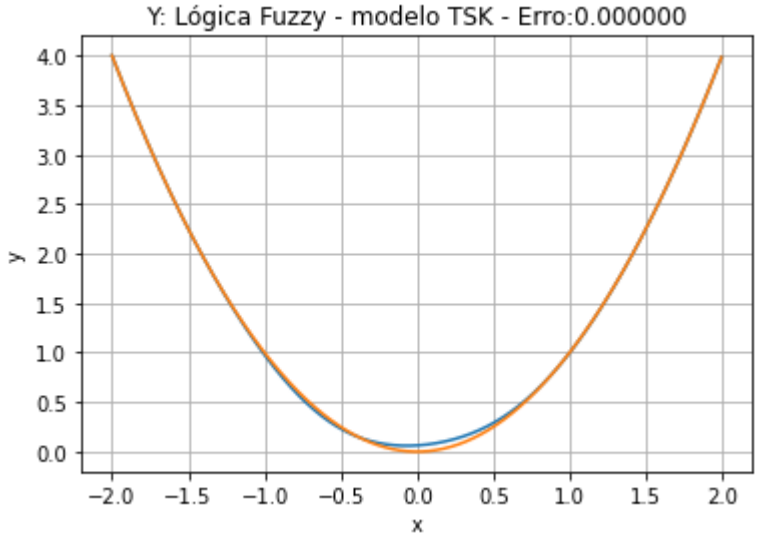
epoca: 339



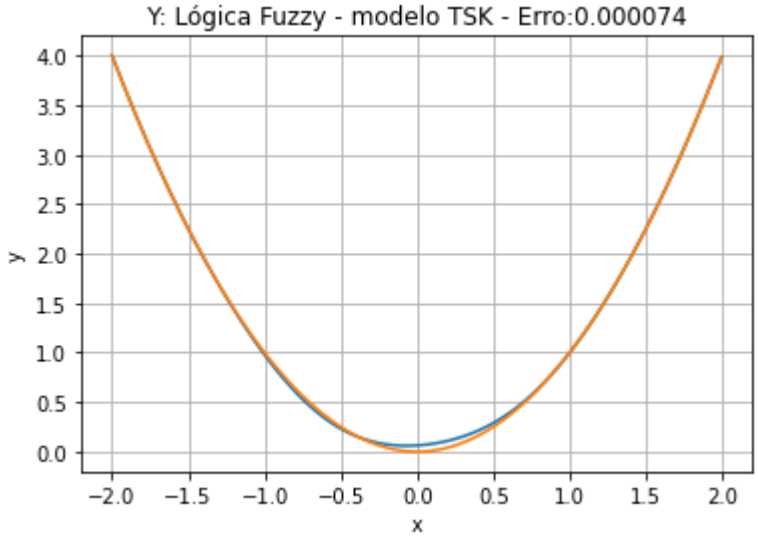
epoca: 340



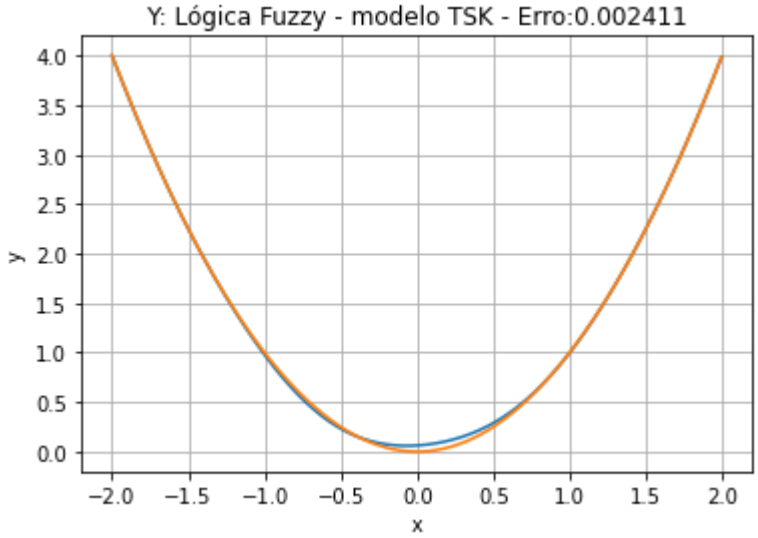
epoca: 341



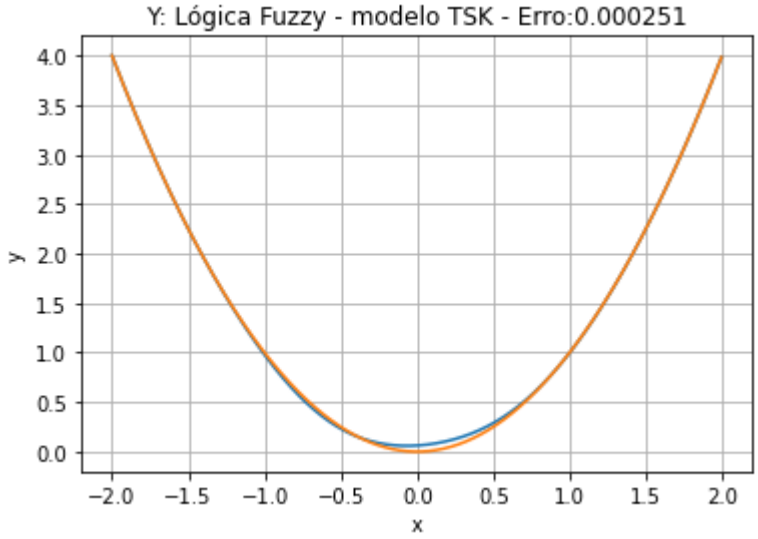
epoca: 342



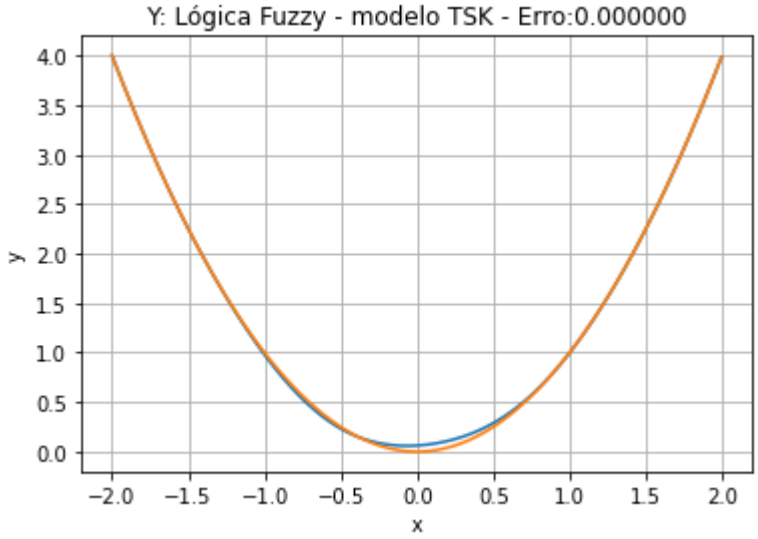
epoca: 343



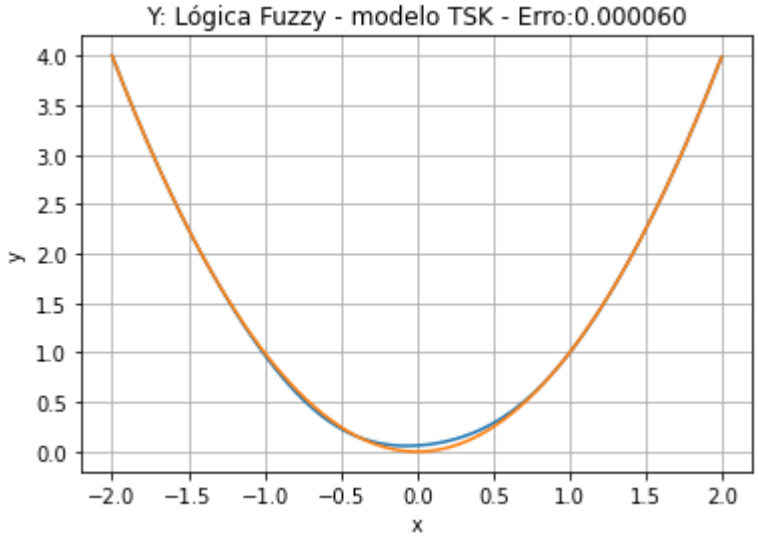
epoca: 344



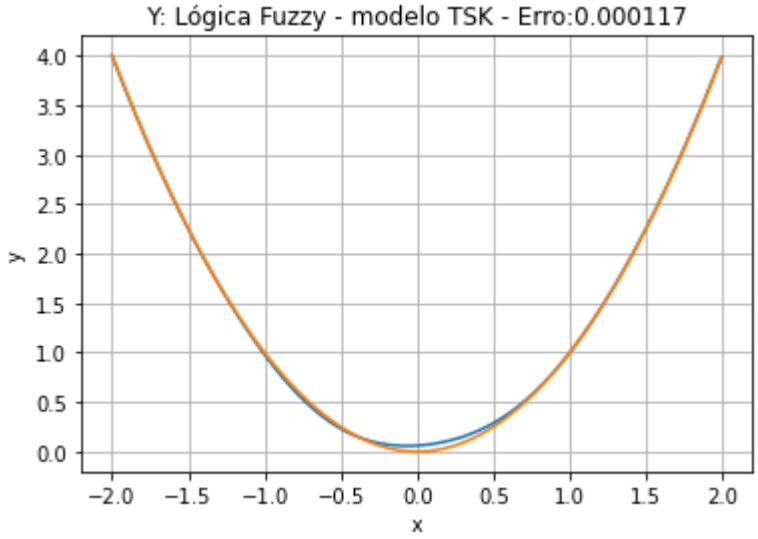
epoca: 345



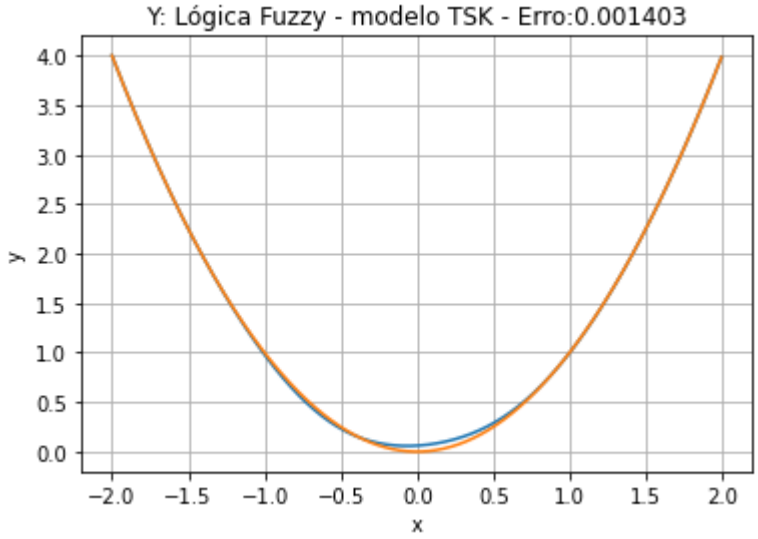
epoca: 346



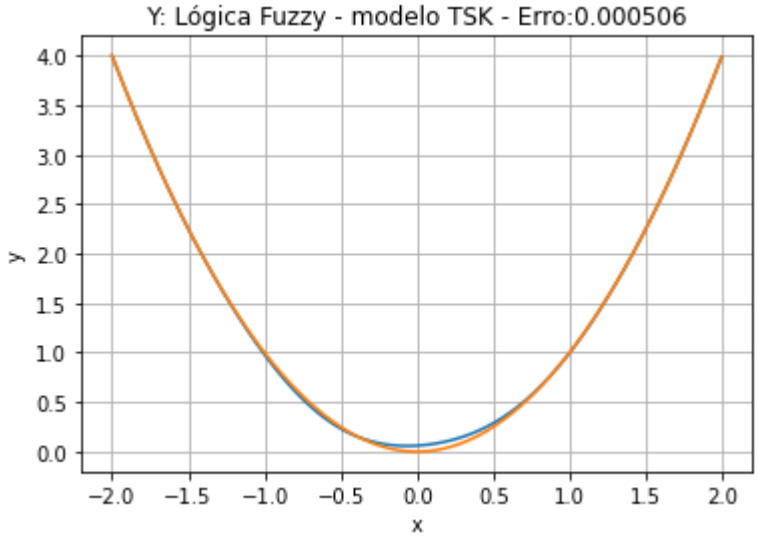
epoca: 347



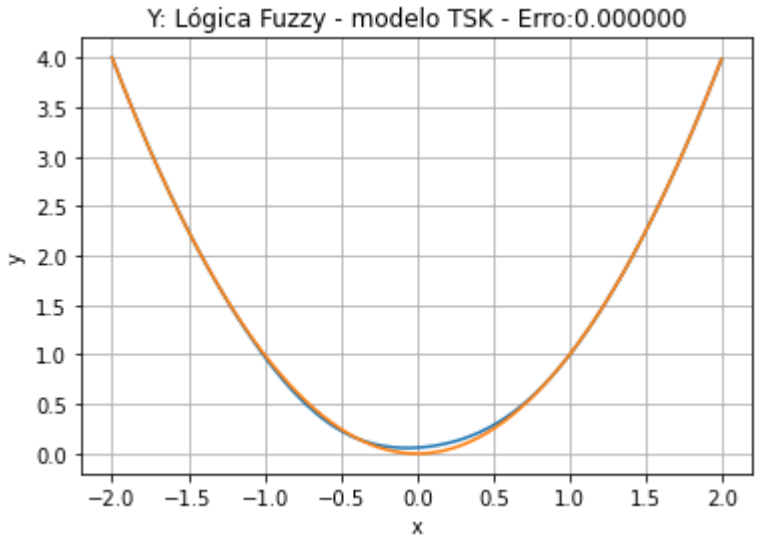
epoca: 348



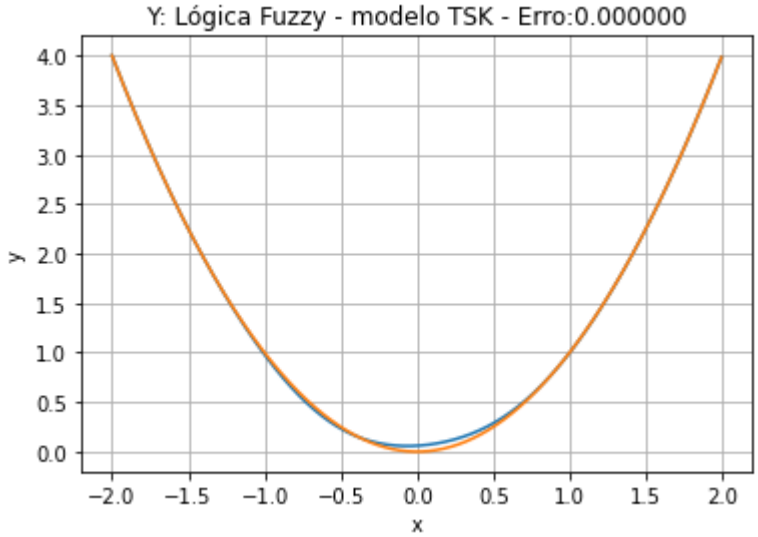
epoca: 349



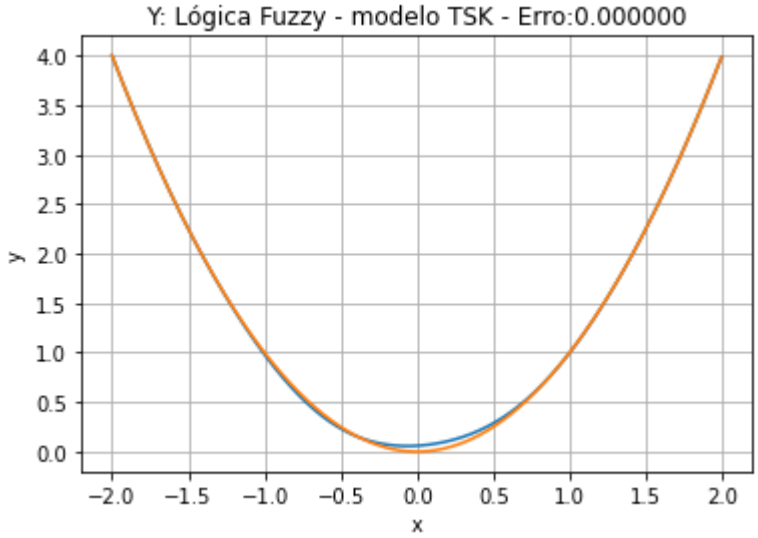
epoca: 350



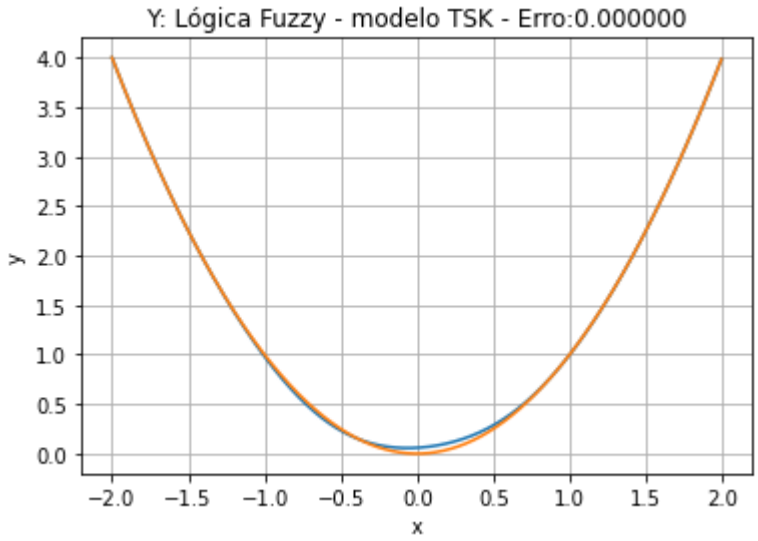
epoca: 351



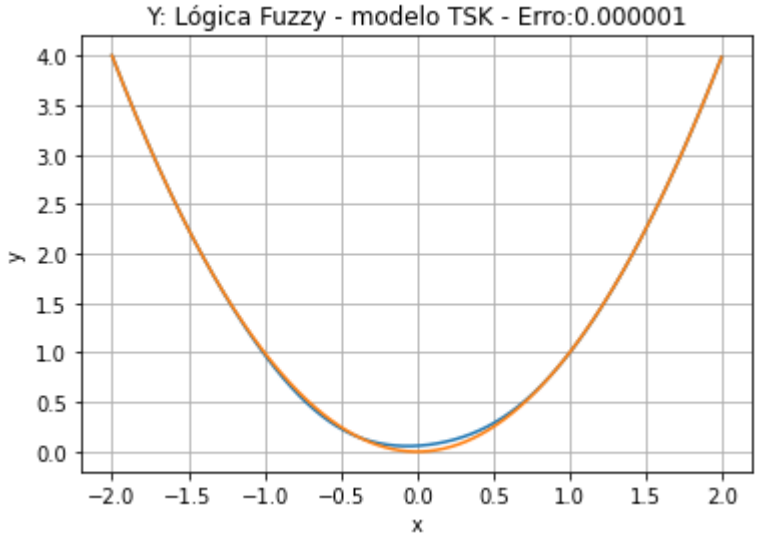
epoca: 352



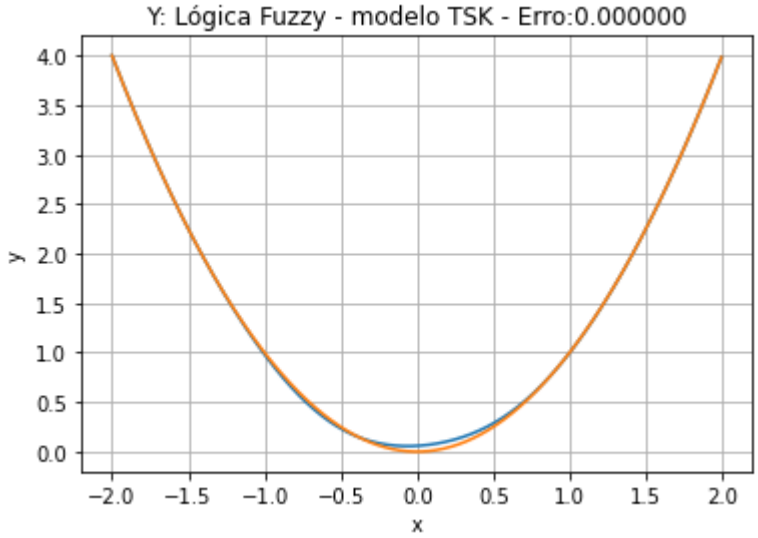
epoca: 353



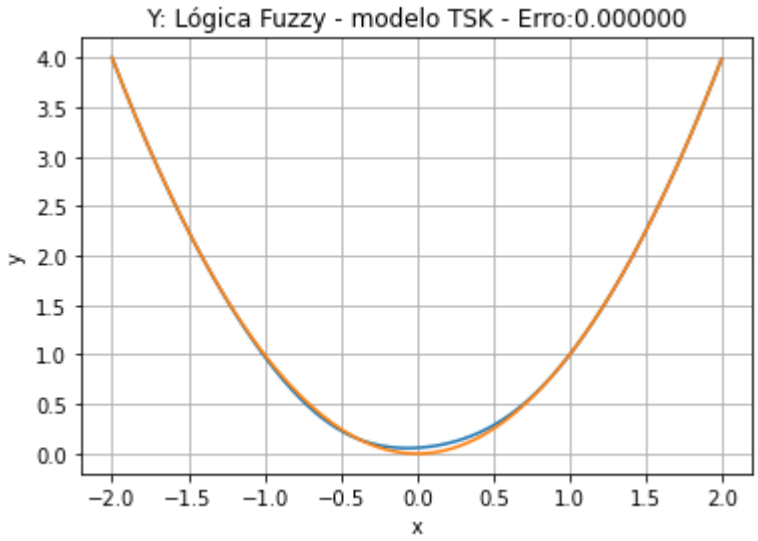
epoca: 354



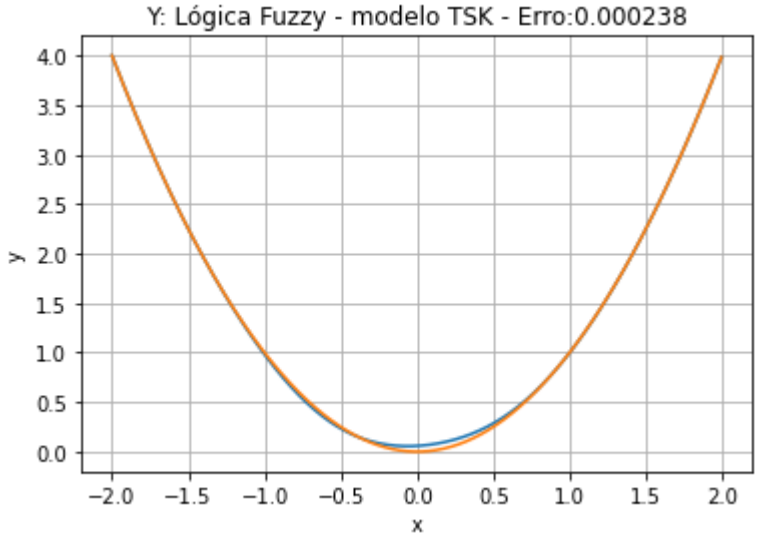
epoca: 355



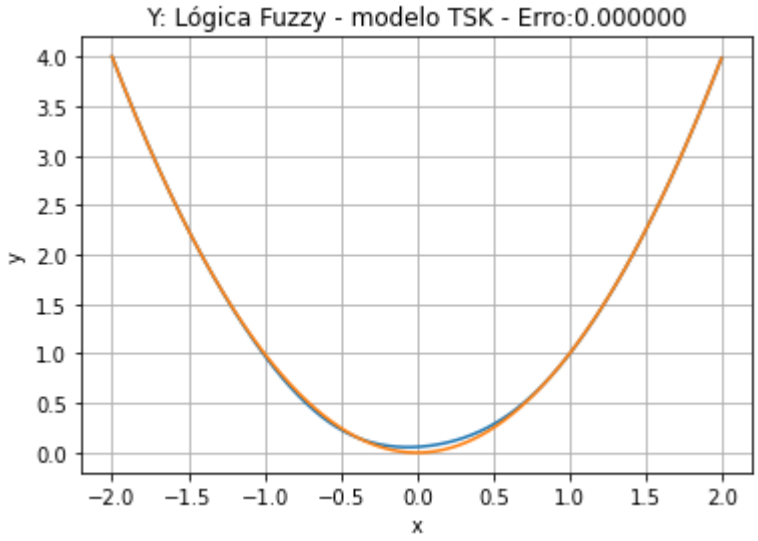
epoca: 356



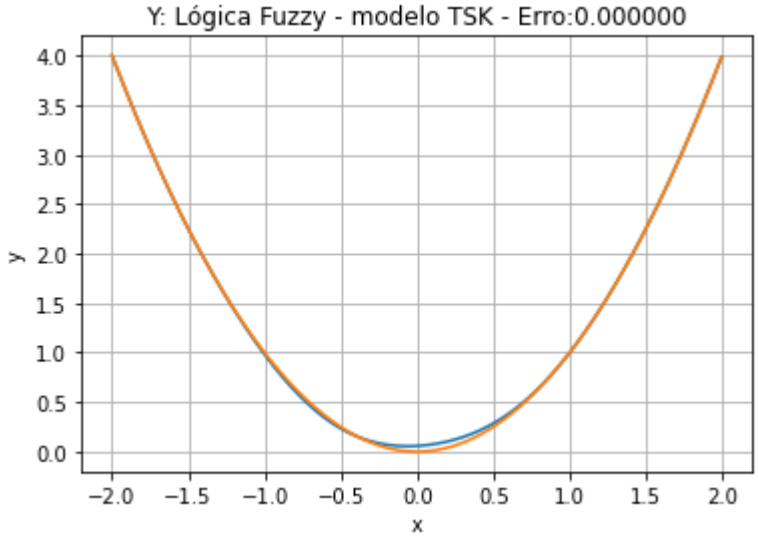
epoca: 357



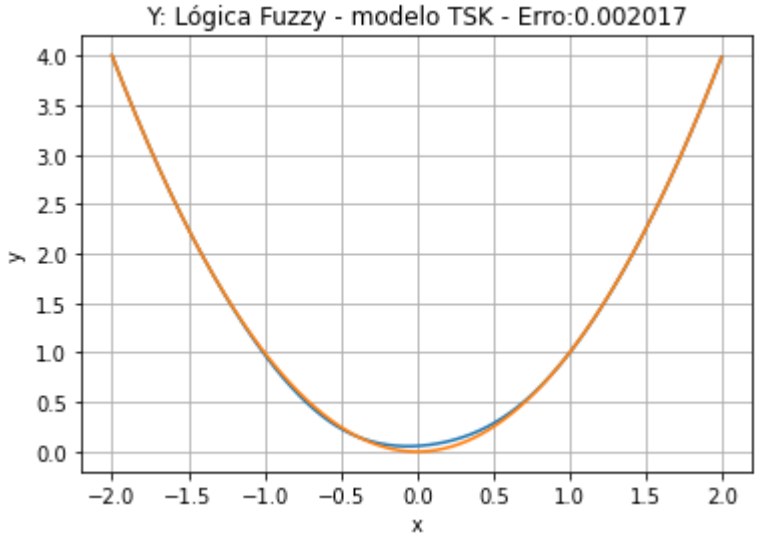
epoca: 358



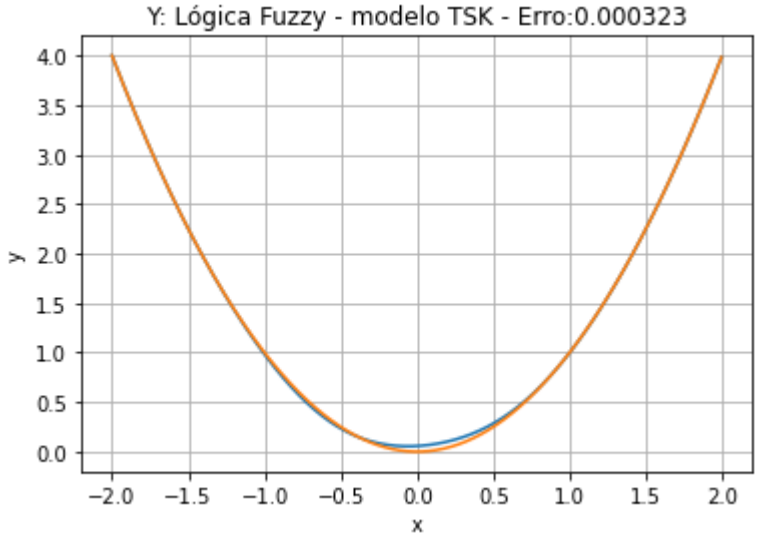
epoca: 359



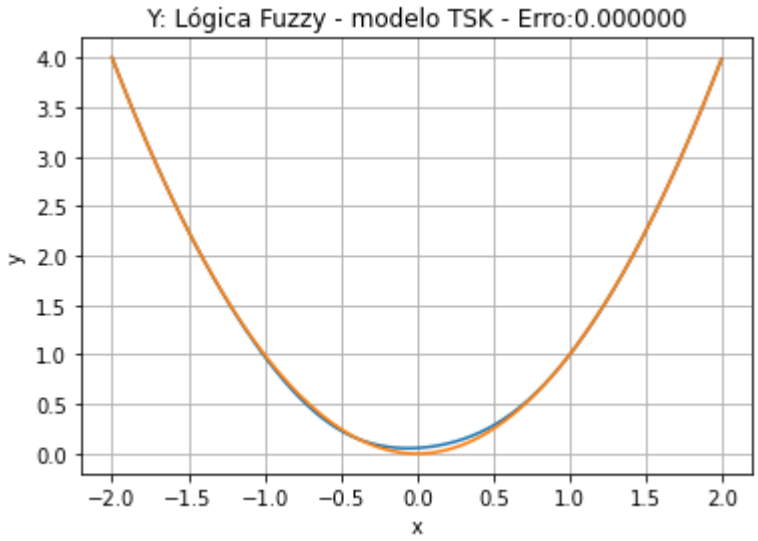
epoca: 360



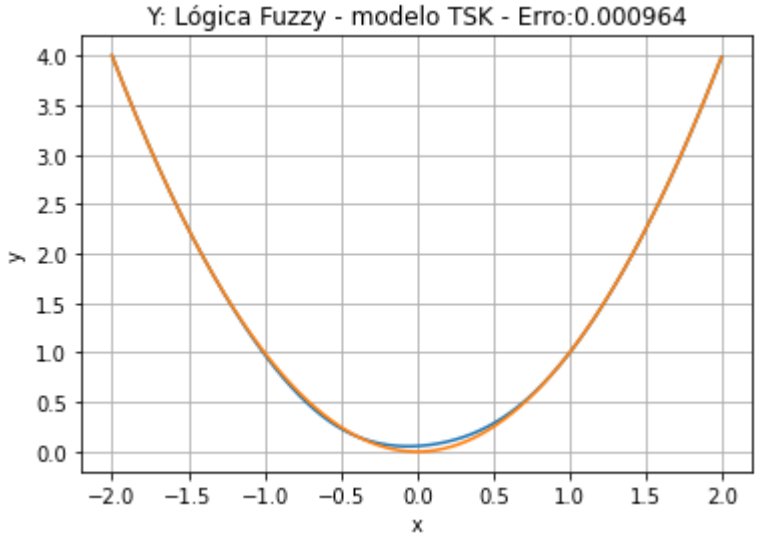
epoca: 361



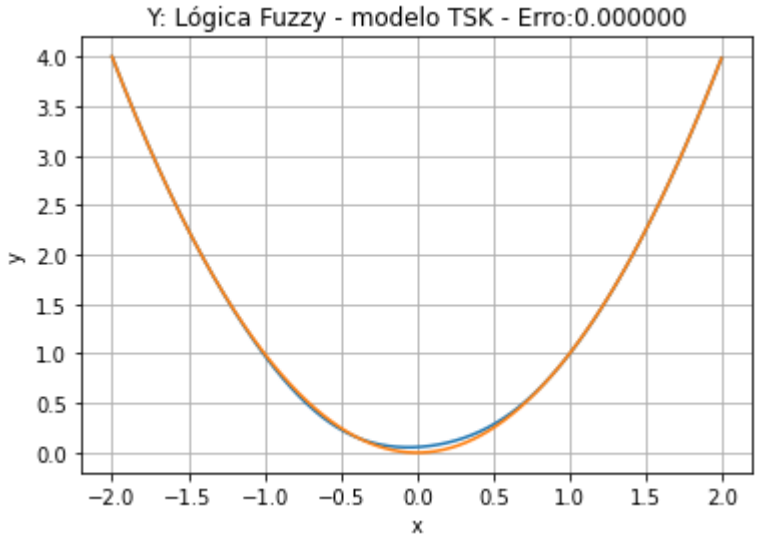
epoca: 362



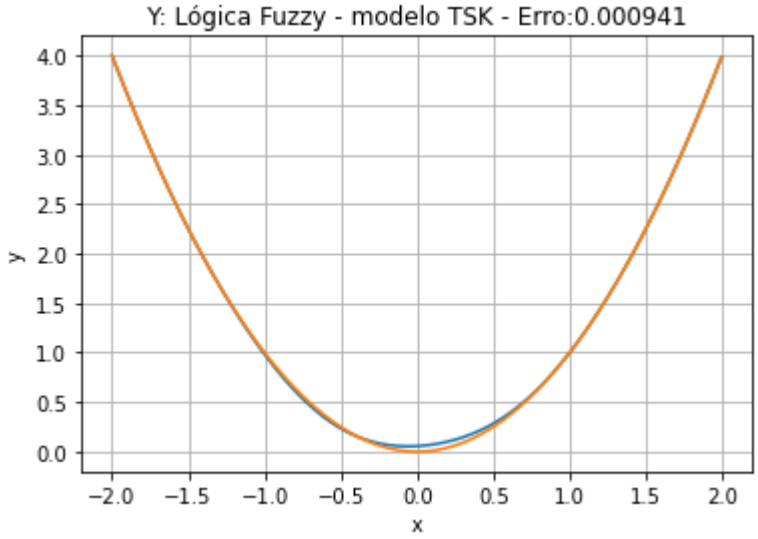
epoca: 363



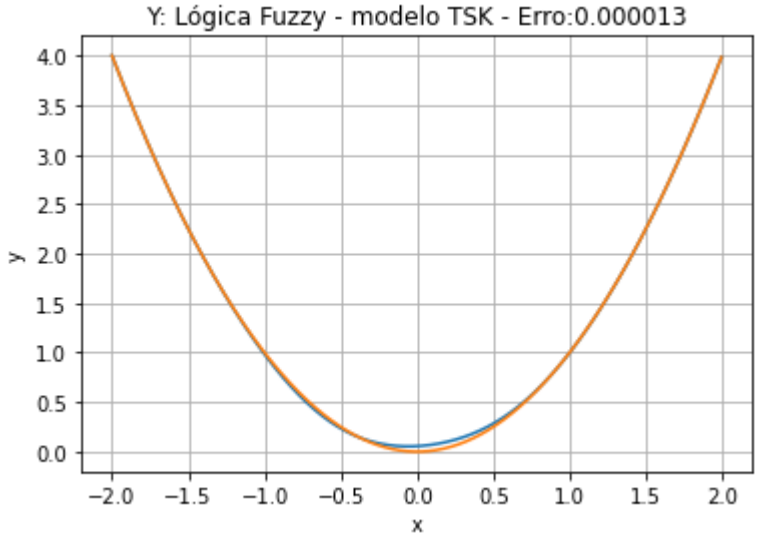
epoca: 364



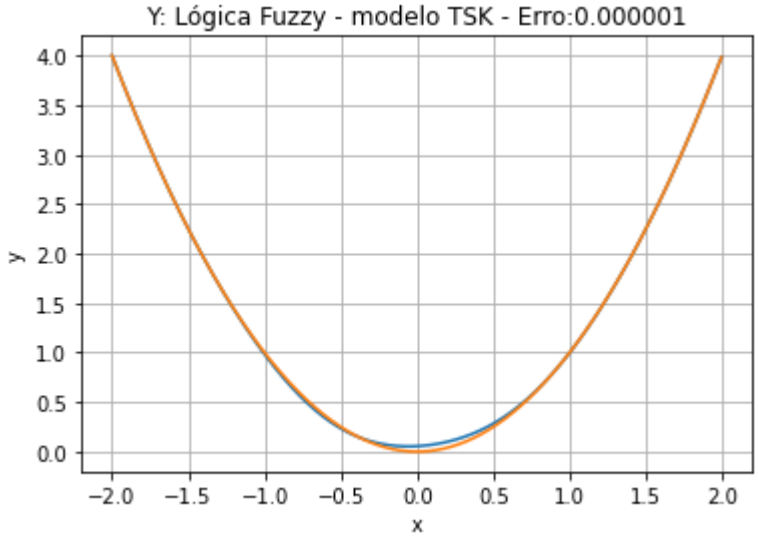
epoca: 365



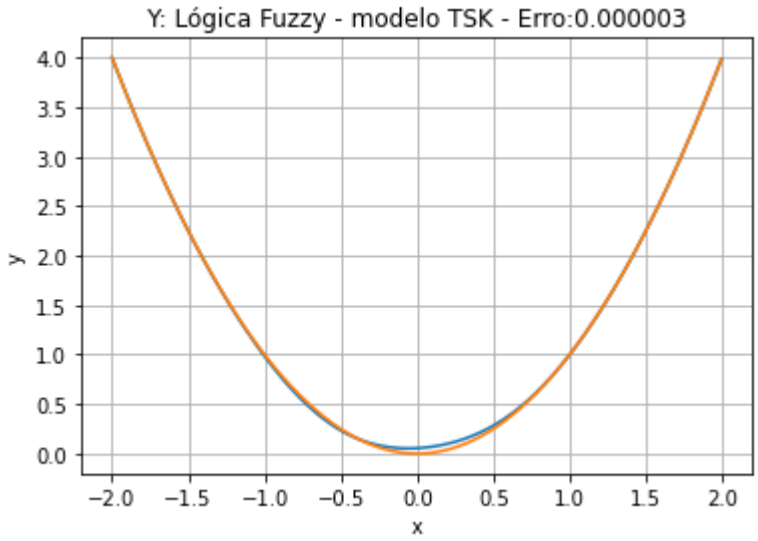
epoca: 366



epoca: 367



epoca: 368



Resultado final

In [82]:
```
PlotaGrafico(x_axis,Y,erro)
```