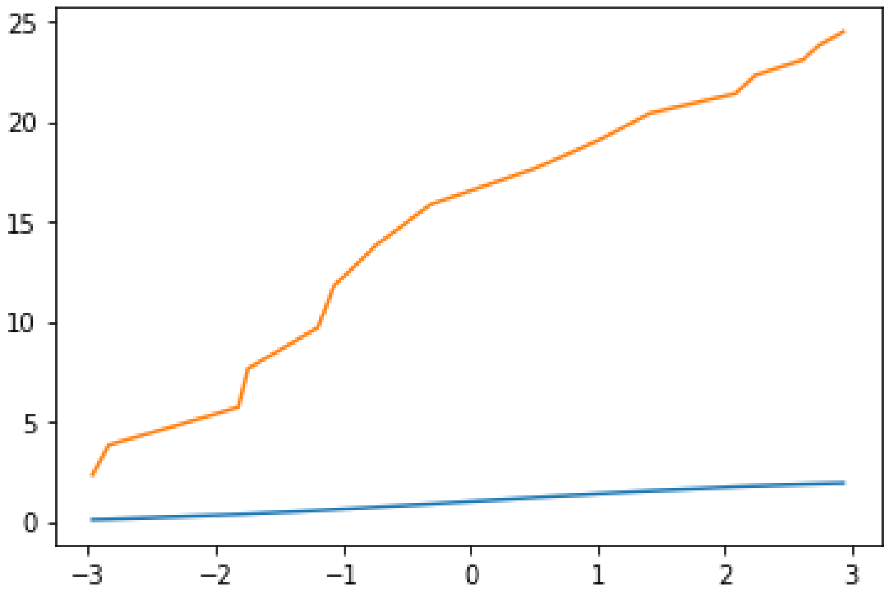
Κεφάλαιο **13**

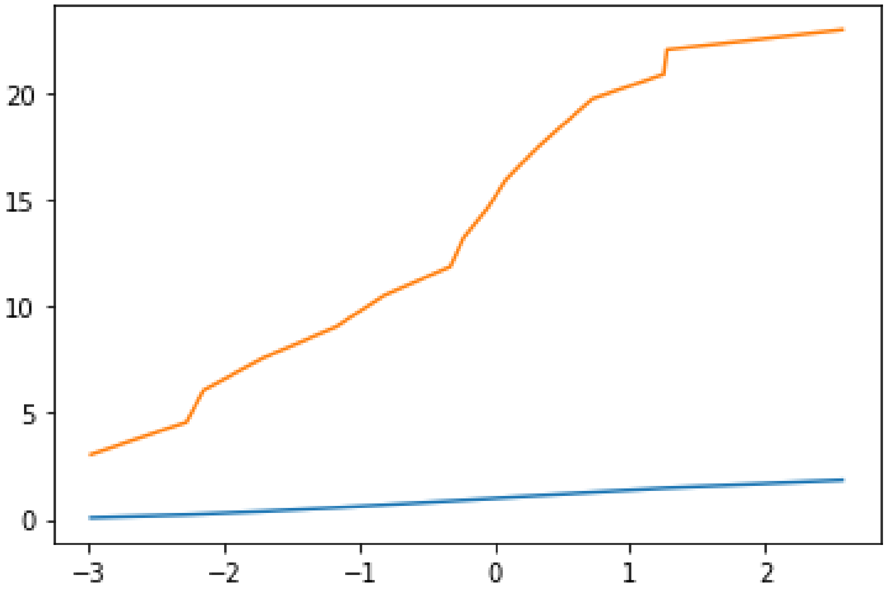
Πρόβλημα 13



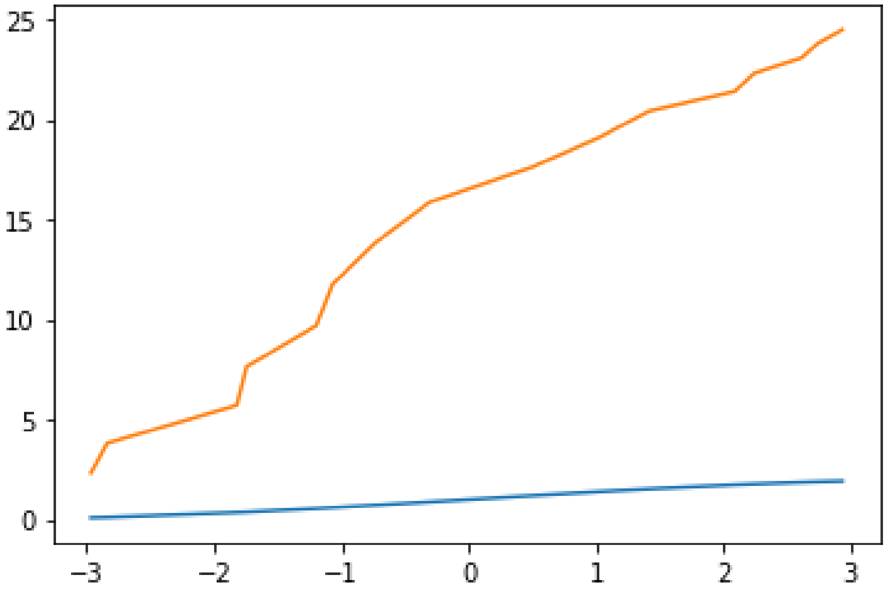
Εικόνα 13.1: *Network Response S=4*

13.1 Python Code

```
import matplotlib.pyplot as plt

from math import sin, pi, exp, sqrt
from random import uniform

# Initialize input vectors

p = [uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),
     uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),
     uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),uniform(-3.0, 3.0),
     uniform(-3.0, 3.0)]
p.sort()
learning_rate = 0.01

def g_function(p):
```



Εικόνα 13.2: *Network Response S=8*



Εικόνα 13.3: *Network Response S=16*

```
    return 1+sin(p*(pi/8))

def radbas(n):
    return exp(-n*n)

def purelin(n):
    return n

def purelin_der(n):
    return 1

def radbas_der(n):
```

```python
    return -2*n*exp(-n*n)

S = 4

# Initialize weights and biases

for k in range(3):
    print(f"For S = {S}")
    w1 = []
    b1 = []
    w2 = []
    for i in range(S):
        w1.append(uniform(0, 0.5))
        b1.append(uniform(0, 0.5))
        w2.append(uniform(0, 0.5))
    b2 = uniform(0, 0.5)

    # Start training

    while True:
        sum_sq_error = 0
        for i in range(10):
            n1 = []
            a1 = []
            n2 = b2
            for j in range(S):
                n = sqrt((p[i]-w1[j])*(p[i]-w1[j]))+b1[j]
                n1.append(n)
                a = radbas(n)
                a1.append(a)
                n2 += a * w2[j]
            a2 = purelin(n2)

            # Calculate error

            e = g_function(p[i])-a2
            sum_sq_error = sum_sq_error + e*e

            # Calculate sensitivities and recalculate weghts and biases

            s2 = -2*purelin_der(n2)*(e)
            s1 = []
            for j in range(S):
                s1.append(radbas_der(n1[j])*w1[j]*s2)

                w2[j] -= learning_rate*s2*a1[j]
            b2 -= learning_rate*s2

            for j in range(S):
                w1[j] -= learning_rate*s1[j]*p[i]
                b1[j] -= learning_rate*s1[j]

        # Check sum square error threshold

        if sum_sq_error <= 1.2:
            break

    # Classify vectors

    result = []
    g = []
    print( f"points: {p}")
    for i in range(16):
        for j in range(S):
            n = p[i]*w1[j]+b1[j]
            a = radbas(n)
            n2 += a * w2[j]
        a2 = purelin(n2)
        result.append(a2)
        g.append(g_function(p[i]))

    # Design plot

    plt.plot(p, g)
    plt.plot(p, result)
    plt.show()

    S = S * 2
```