Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

`

Лабораторная работа №2

По дисциплине: «МиАПР»

## Тема: «Линейная искусственная нейронная сеть. Адаптивный шаг обучения.»

Выполнил:

Студент 2 курса

Группы ПО-7(2)

Угляница И.Н

Проверил:

Крощенко А.А.

Брест 2021

Вариант 10

**Цель работы:** изучить обучение и функционирование линейной ИНС при решении задач прогнозирования.

**Задание:** Написать на любом ЯВУ программу моделирования прогнозирующей линейной ИНС. Для тестирования использовать функцию:

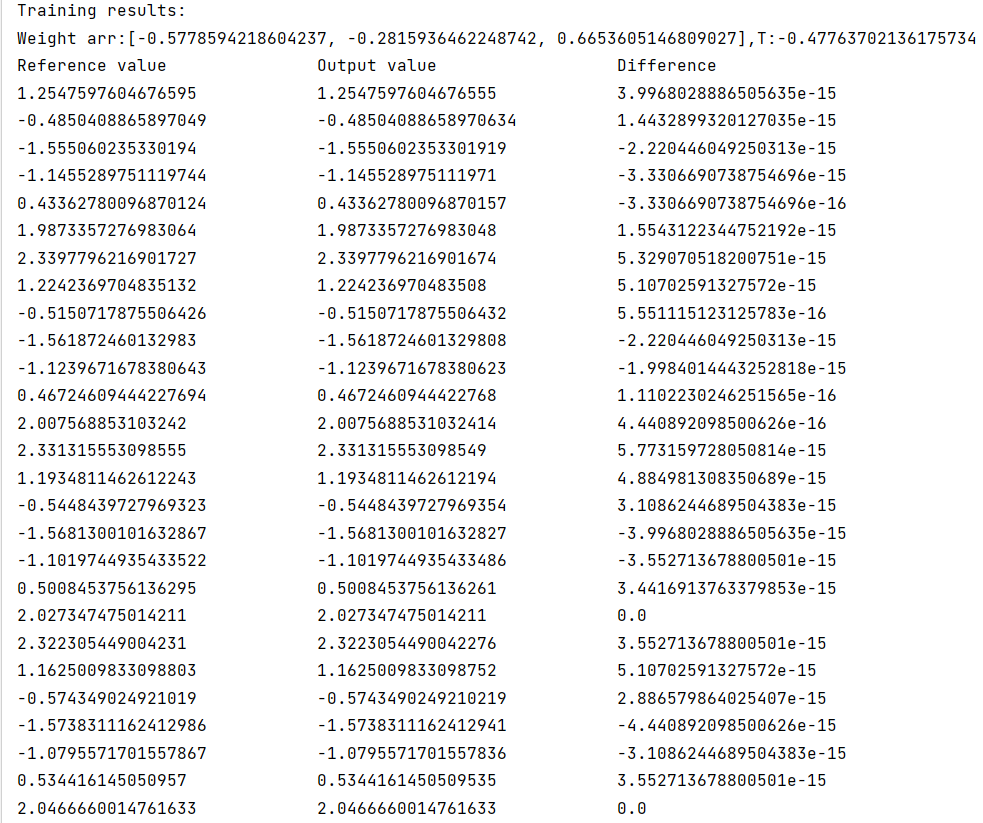
y = a\*sin(bx) + d

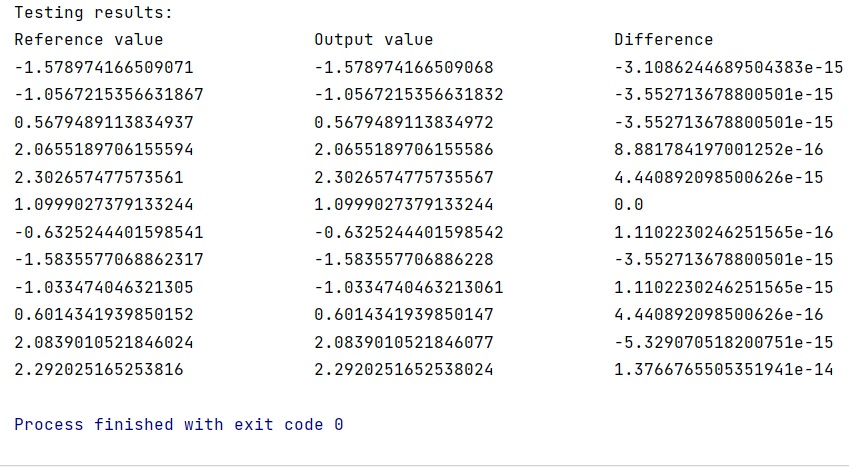
а, b, d = 2, 9, 0.4, кол-во входов ИНС = 3.

**Код программы:**

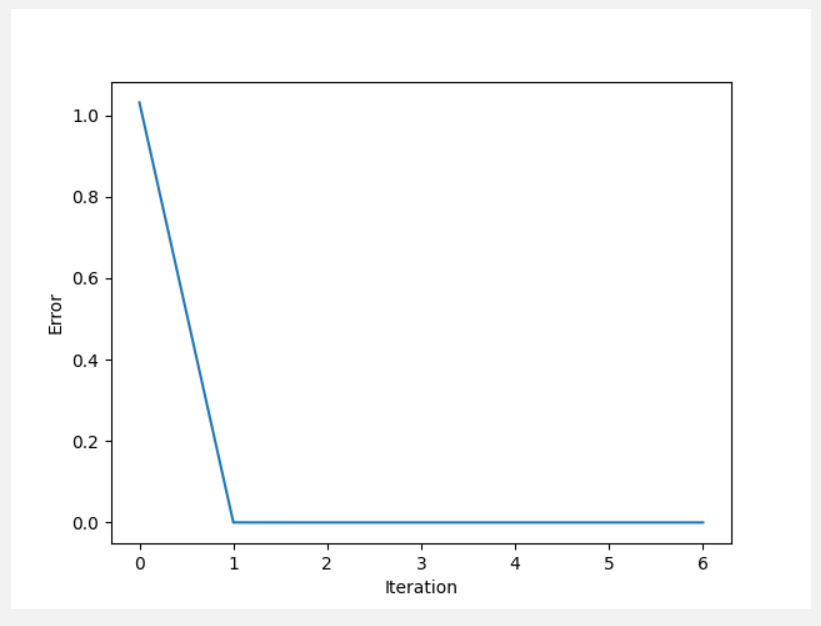
import math  
import random  
import matplotlib.pyplot as plt  
  
  
def func(x):  
 a, b, d = 2, 9, 0.4  
 return a \* math.sin(b \* x) + d  
  
  
def output\_value(  
 weight\_arr, reference\_arr, T, shift, amount\_inputs  
) -> float: *# Формула 1.2  
 # Формула 1.2 нахождение выходного значения* value = 0  
 for j in range(amount\_inputs):  
 value += weight\_arr[j] \* reference\_arr[j + shift]  
 return value - T  
  
  
def calc\_w(weight\_arr, t\_alpha, reference\_arr, value, shift, amount\_inputs, T):  
 *# Формула 1.7 обновление весов* for j in range(len(weight\_arr)):  
 weight\_arr[j] -= (  
 t\_alpha  
 \* (value - reference\_arr[amount\_inputs + shift])  
 \* reference\_arr[shift + j]  
 )  
def adaptive\_alpha(reference\_arr,amount\_inputs,shift):  
 temp = 0  
 for i in range(amount\_inputs):  
 temp += reference\_arr[i+shift]\*\*2  
 return 1 / (1 + temp)  
  
  
  
def main():  
 train\_epoch = 30 *# количество значений для обучения* prediction\_epoch = 15 *# количество значений для прогнозирования* amount\_inputs = 3 *# количество входных нейронов* t\_alpha = 0 *# скорость обучения* step = 0.1 *# шаг табулирования* min\_еrror = 1.0e-29  
 weight\_arr = [  
 random.uniform(0, 1) for i in range(amount\_inputs)  
 ] *# массив рандомных весов* T = random.uniform(0, 1) *# порог* reference\_arr = [  
 func(i \* step) for i in range(train\_epoch)  
 ] *# эталонные значения для обучения* error = 1  
 plot\_arr = []  
  
 while error >min\_еrror:  
 error = 0  
 for i in range(train\_epoch - amount\_inputs):  
 value = output\_value(weight\_arr, reference\_arr, T, i, amount\_inputs)  
 t\_alpha = adaptive\_alpha(reference\_arr, amount\_inputs, i) *# вычисление адаптивного шага обучения*  
 calc\_w(weight\_arr, t\_alpha, reference\_arr, value, i, amount\_inputs, T)  
 T += t\_alpha \* (  
 value - reference\_arr[amount\_inputs + i]  
 ) *# формула 1.8 обновление порога* error += (value - reference\_arr[i + amount\_inputs]) \*\* 2  
 error /= train\_epoch - amount\_inputs  
 plot\_arr.append(error)  
 plt.plot(plot\_arr)  
 plt.xlabel("Iteration")  
 plt.ylabel("Error")  
 plt.show()  
  
 print("Training results:")  
 print(f"Weight arr:{weight\_arr},T:{T}")  
 print("{:30}{:30}{:30}".format("Reference value", "Output value", "Difference"))  
  
 for i in range(train\_epoch - amount\_inputs):  
 value = output\_value(weight\_arr, reference\_arr, T, i, amount\_inputs)  
 print(  
 "{:<30}{:<30}{:<30}".format(  
 reference\_arr[amount\_inputs + i],  
 value,  
 reference\_arr[amount\_inputs + i] - value,  
 )  
 )  
 reference\_arr\_t = [  
 func(i \* step) for i in range(train\_epoch, train\_epoch + prediction\_epoch)  
 ] *# массив эталонных значений для прогнозирования* print("Testing results:")  
 print("{:30}{:30}{:30}".format("Reference value", "Output value", "Difference"))  
 for i in range(prediction\_epoch - amount\_inputs):  
 value = output\_value(weight\_arr, reference\_arr\_t, T, i, amount\_inputs)  
 print(  
 "{:<30}{:<30}{:<30}".format(  
 reference\_arr\_t[amount\_inputs + i],  
 value,  
 reference\_arr\_t[amount\_inputs + i] - value,  
 )  
 )  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Результат:**





**График изменения ошибки:**



**Вывод:** Исходя из графика и полученных значений можно сделать вывод, что благодаря адаптивному шагу обучения нейронная сеть быстрее обучается, а также ее выходные значения получаются более точными.