Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «МиАПР»

Тема: «Линейная искусственная нейронная сеть. Адаптивынй шаг обучения»

Выполнил:

Студент 2 курса

Группы ПО-7(2)

Панкратов Р.С.

Проверил:

Крощенко А.А.

2021

Вариант 3

**Цель работы:**

изучить обучение и функционирование линейной ИНС при решении

задач прогнозирования с применением адаптивного шага.

**Задание:**

Модифицировать программу из лабораторной работы №1, используя правило адаптивного шага обучения. Произвести исследование получившейся модели ИНС на задачах прогнозирования, согласно варианту лабораторной работы № 1

Задание из лабораторной работы № 1:

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

y = a\*sin(b\*x) + d

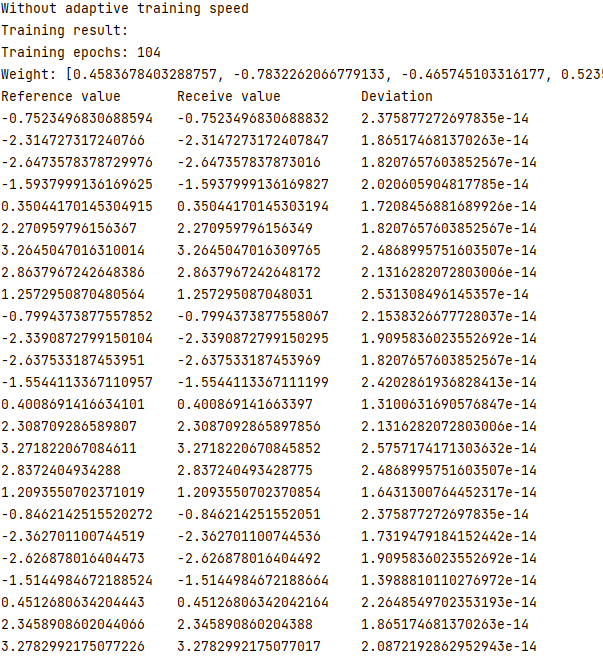
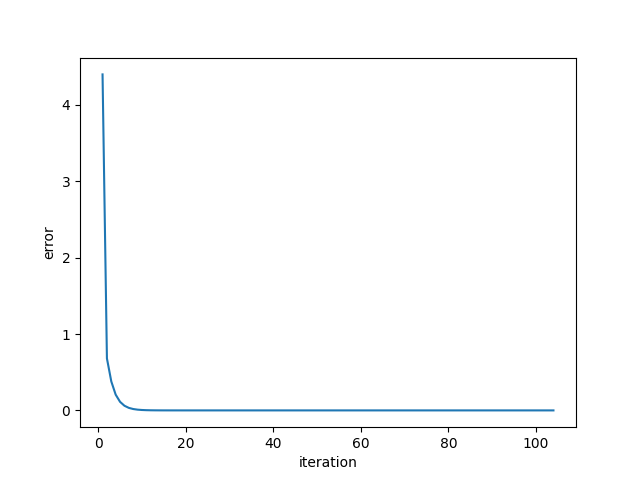
a = 3, b = 7, d = 0.3, кол-во входов ИНС = 5.

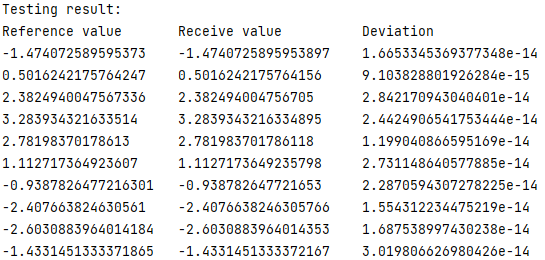
**Код программы:**

import math  
import random  
from typing import Tuple, List, Sequence  
from dataclasses import dataclass  
  
import matplotlib.pyplot as plt  
  
MIN\_ERROR = 1.0e-27 *# Минимальная ошибка для остановки обучени*TRAINING\_SPEED = 0.01 *# Скорость обучения нейронной сети*@dataclass(frozen=True)  
class TrainingResult:  
 w: List[float]  
 T: float  
 data\_for\_drawing: Tuple[List[int], List[float]]  
 epochs: int  
  
  
def func(x: float, a: int, b: int, d: float) -> float:  
 return a \* math.sin(b \* x) + d  
  
  
def calculate\_output(  
 nn\_inputs: int, w: Sequence[float], t: float, training\_outputs: Sequence[float], offset: int  
) -> float:  
 output = 0  
 for j in range(nn\_inputs):  
 output += w[j] \* training\_outputs[j + offset]  
 return output - t  
  
  
def training\_nn(  
 nn\_inputs: int, training\_outputs: Sequence[float], is\_adaptive\_training\_speed: bool = False  
) -> TrainingResult:  
 w = [random.uniform(0, 1) for \_ in range(nn\_inputs)] *# Веса* T = random.uniform(0.5, 1) *# Порог* data\_for\_drawing = ([], []) *# Данные для рисования графика* training\_speed = TRAINING\_SPEED  
  
 error, iteration, epochs = 1, 0, 0  
  
 while error > MIN\_ERROR:  
 error = 0  
  
 for i in range(len(training\_outputs) - nn\_inputs):  
 if is\_adaptive\_training\_speed:  
 *# Получение адаптивного шага обучения, лаб 2 формула(10)* temp = 0  
 for j in range(nn\_inputs):  
 temp += training\_outputs[i + j] \*\* 2  
 training\_speed = 1 / (1 + temp)  
  
 *# Получение выходного значения нейронной сети, формула(1.2)* output = calculate\_output(nn\_inputs, w, T, training\_outputs, i)  
  
 *# Обновление весов нейронной сети, формула(1.7)* for j in range(nn\_inputs):  
 ideal\_output = training\_outputs[i + nn\_inputs]  
 w[j] -= training\_speed \* (output - ideal\_output) \* training\_outputs[i + j]  
  
 *# Обновление порога нейронной сети, формула(1.8)* T += training\_speed \* (output - training\_outputs[i + nn\_inputs])  
  
 *# Обновление среднеквадратичной ошибки нейронной сети, формула(1.3)* error += (output - training\_outputs[i + nn\_inputs]) \*\* 2  
  
 error /= len(training\_outputs) - nn\_inputs  
  
 epochs += 1  
 iteration += 1  
  
 data\_for\_drawing[0].append(iteration)  
 data\_for\_drawing[1].append(error)  
  
 return TrainingResult(w, T, data\_for\_drawing, epochs)  
  
  
def draw\_graph(x\_data: List[int], y\_data: List[float]) -> None:  
 plt.plot(x\_data, y\_data)  
 plt.ylabel('error')  
 plt.xlabel('iteration')  
 plt.show()  
  
  
def print\_nn\_output(w: List[float], t: float, ideal\_outputs: List[float], nn\_inputs: int) -> None:  
 print(f"{'Reference value':<22}{'Receive value':<23}{'Deviation'}")  
  
 for i in range(len(ideal\_outputs) - nn\_inputs):  
 output = calculate\_output(nn\_inputs, w, t, ideal\_outputs, i)  
 print(f"{ideal\_outputs[i+nn\_inputs]:<20} {output: <21} {(ideal\_outputs[i+nn\_inputs] - output)}")  
  
  
def print\_nn\_work\_report(  
 training\_result: TrainingResult, nn\_inputs: int, training\_outputs: List[float], testing\_outputs: List[float]  
) -> None:  
 draw\_graph(\*training\_result.data\_for\_drawing)  
  
 print("Training result:")  
 print(f"Training epochs: {training\_result.epochs}")  
 print(f"Weight: {training\_result.w}, T: {training\_result.T}")  
 print\_nn\_output(training\_result.w, training\_result.T, training\_outputs, nn\_inputs)  
  
 print("\nTesting result:")  
 print\_nn\_output(training\_result.w, training\_result.T, testing\_outputs, nn\_inputs)  
  
  
def main():  
 a, b, d = 3, 7, 0.3  
 nn\_inputs = 5 *# Количество входных значений(входов)* training\_values, testing\_values = 30, 15 *# Количество значений функции для обучения и прогнозирования* step = 0.1 *# Шаг табуляции функции* training\_outputs = [func(i \* step, a, b, d) for i in range(training\_values)]  
 testing\_outputs = [func(i \* step, a, b, d) for i in range(training\_values, training\_values + testing\_values)]  
 try:  
 training\_result = training\_nn(nn\_inputs, training\_outputs, False)  
 except OverflowError:  
 return print("The neural network is divergent, the learning speed is too fast, reduce it")  
  
 print("Without adaptive training speed")  
 print\_nn\_work\_report(training\_result, nn\_inputs, training\_outputs, testing\_outputs)  
  
 training\_result = training\_nn(nn\_inputs, training\_outputs, True)  
 print("With adaptive training speed")  
 print\_nn\_work\_report(training\_result, nn\_inputs, training\_outputs, testing\_outputs)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

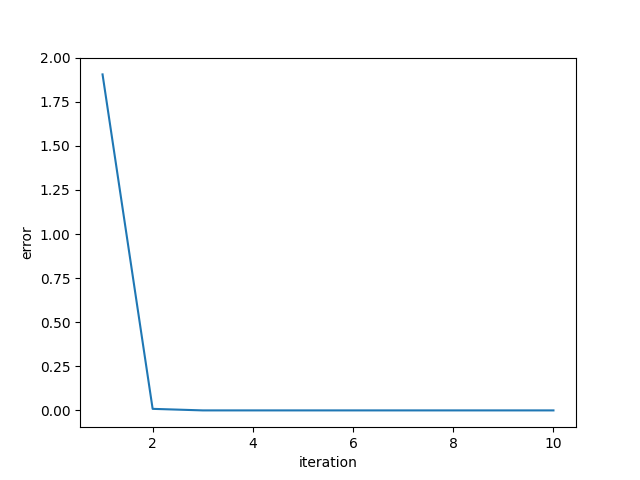
**Результат:**

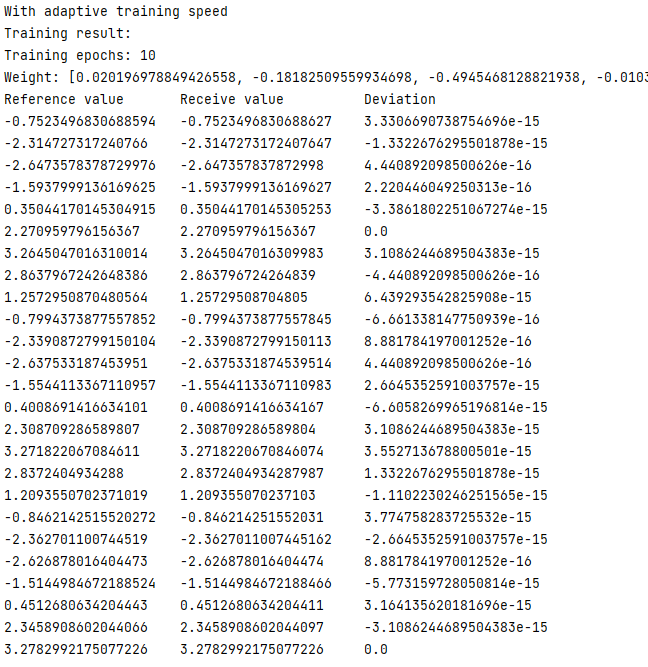
Результаты без адаптивного шага обучения:

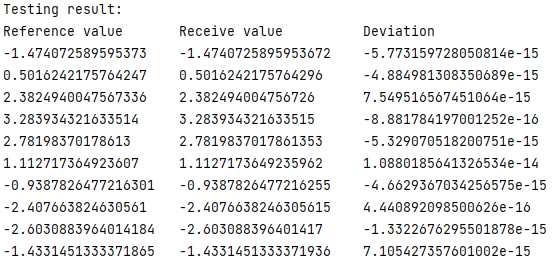




Результаты c адаптивным шагом обучения:





****

**Вывод:**

Без адаптивного шага нейронная сеть обучалась 104 эпохи, с ним 10 эпох. Адаптивный шаг обучения позволяет существенно сократить количество эпох обучения искусственной нейронной сети.