**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

**ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**Лабораторная работа №3 на тему:**

«Применение однослойной нейронной сети с линейной функцией активации для прогнозирования временных рядов»

Вариант 4

**Преподаватель:**

Коннова Н.С.

**Студент**:

Куликова А.В.

**Группа:**

ИУ8-21М

**Цель работы**

Изучить возможности однослойных НС в задачах прогнозирования временных рядов методом скользящего окна (авторегрессия).

**Постановка задачи**

|  |  |  |  |
| --- | --- | --- | --- |
| № Варианта | Функция x(t) | a | b |
| 4 |  | -5 | 3 |

**Ход работы**

В ходе исследования был проведен ряд испытаний, на основе которых были получены оптимальные значения НС для предсказания графика функции.

Было проведено исследования зависимости размера ошибки от ширины окна. Как видно из рисунка 1, оптимальная ширина окна – 10.

После нахождения оптимальной ширины окна и нормы обучения, была обучена нейронная сеть по данным параметрам. Результат обучения представлен на рисунке 1. Зависимость ошибки от количества итераций указан на рисунке 2. Начиная с 200 итерации – размер ошибки уменьшается незначительно. Таким образом найдет оптимальный норм критериев.

Результат программы представленной в приложении А:

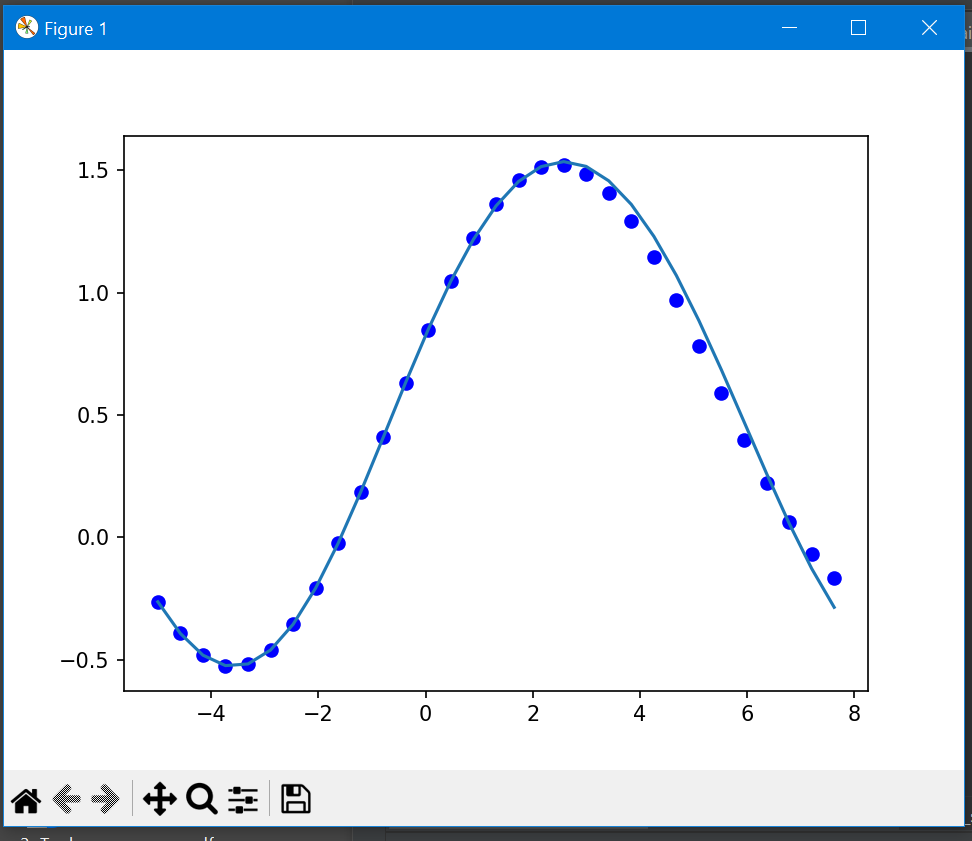


Рисунок 1 - Результат работы НС

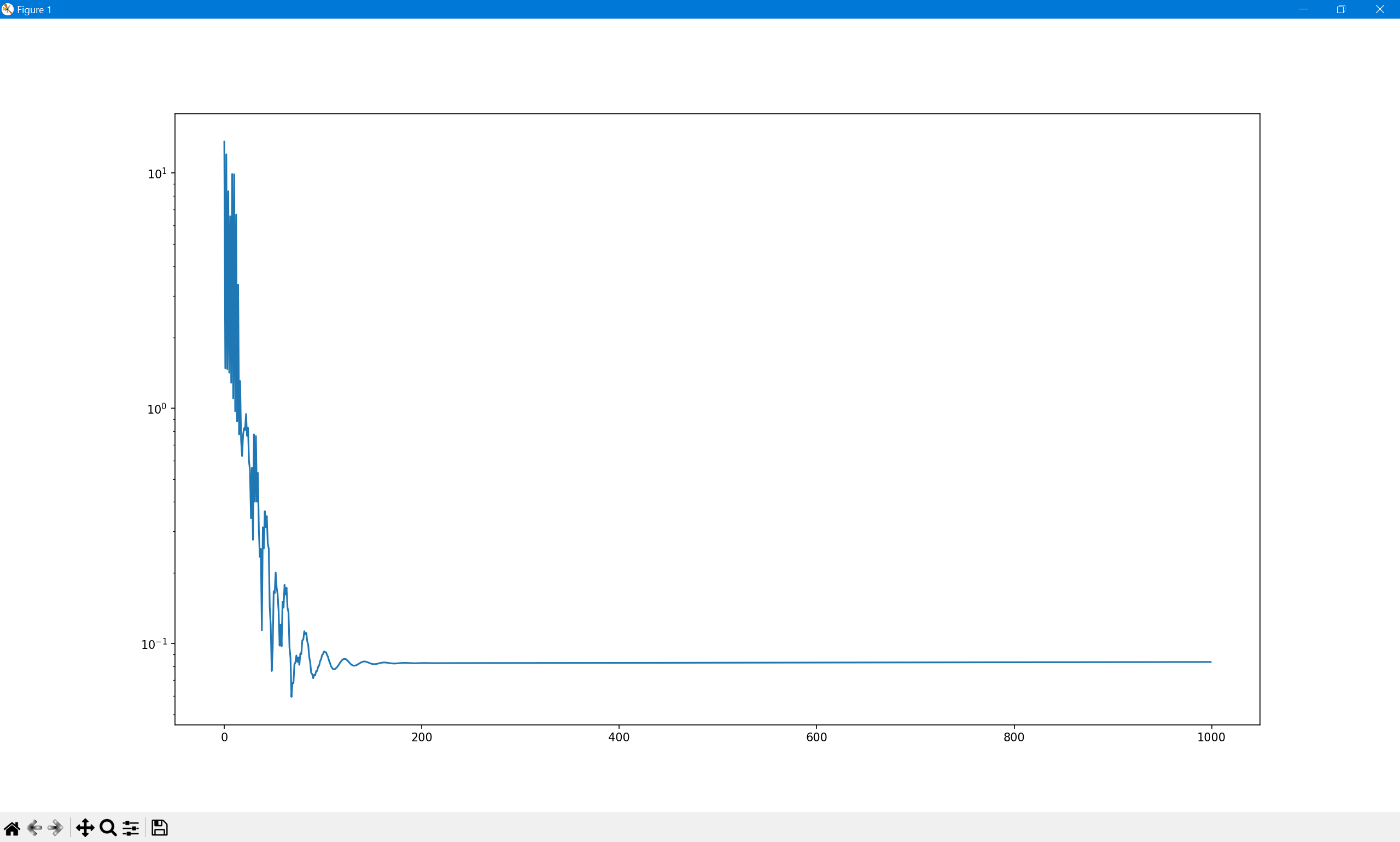


Рисунок 2 - Отношение ошибки к количествам эпох обучения

Создана нейронная сеть со следующими параметрами:

Вектор весов: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

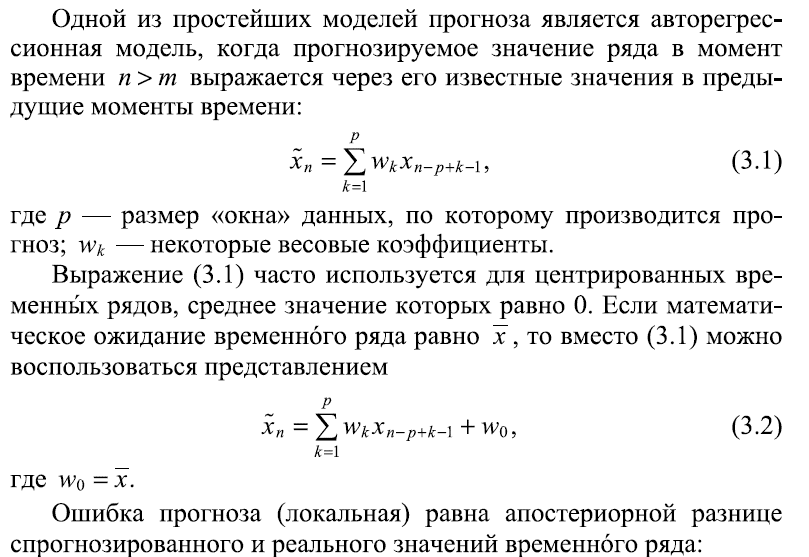
Размер конечной ошибки: 0.084

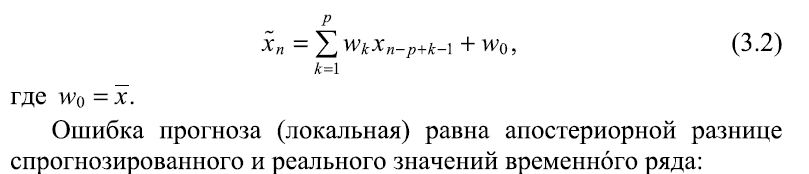
**Выводы**

В ходе лабораторной работы было доказано, что ошибка имеет свойство изменяться в зависимости от количества итераций, ширины окна, а также нормы обучения. Помимо этого, выведены наилучшее входные данные для НС.

**Контрольные вопросы**

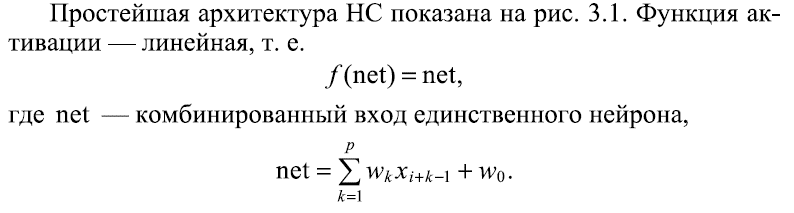
**1.** **В чем состоит принцип прогнозирования на основе авторегрессии?**

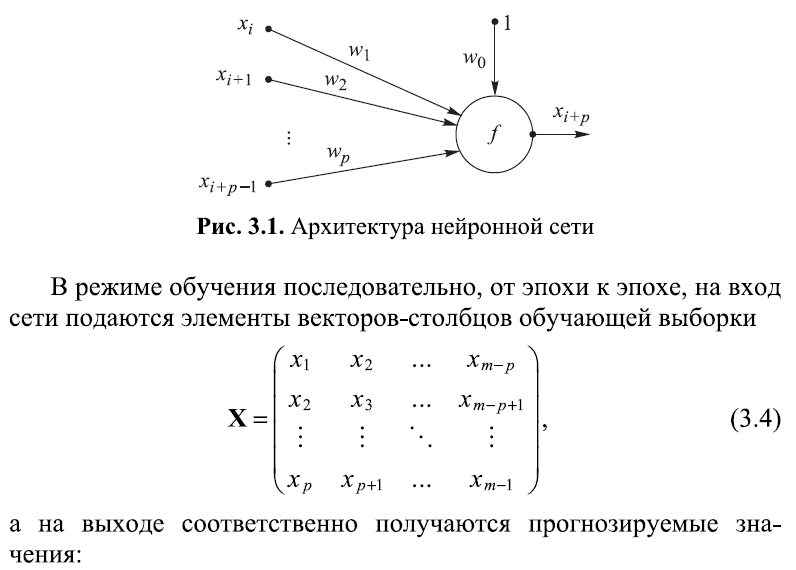


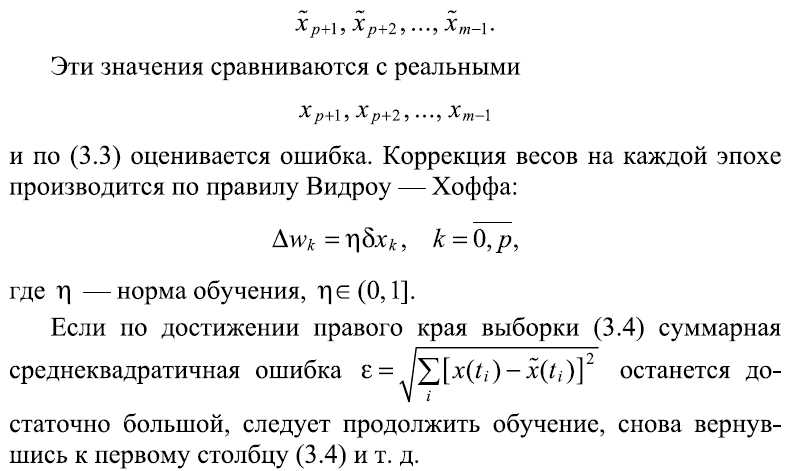




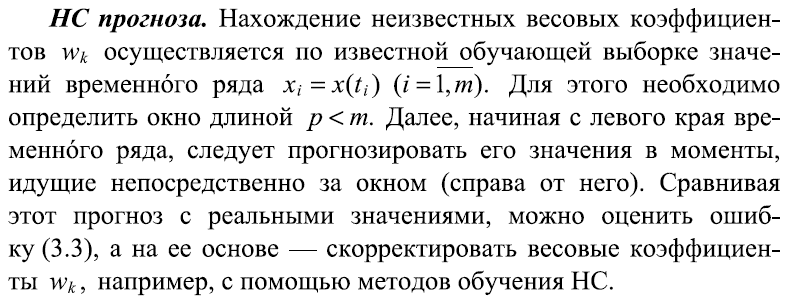
**2. Объясните методику обучения НС прогноза.**







**3. Поясните принцип функционирования НС прогноза.**

. 

**Приложение А**

Листинг программы:

import math  
import matplotlib.pyplot as plt  
  
class Network:  
  
 *"""Инициализация нейронной сети с заданными параметрами."""* def \_\_init\_\_(self, synapses: list, sample: list, N, p, end, a, b):  
 self.synapses = synapses  
 self.sample = sample  
 self.errors = []  
 self.N = N  
 self.p = p  
 self.end = end  
 self.a = a  
 self.b = b  
 print("Создана нейронная сеть со следующими параметрами: ")  
 print("Вектор весов: ", self.synapses)  
  
 """ Рассчитывает выходной сигнал нейронной сети для заданных входных данных. """  
 def iteration(self, inputs: list):  
 net = 0  
 for i in range(len(self.synapses)):  
 net += self.synapses[i] \* inputs[i]  
 return net  
  
 """ Прогнозирует следующие значения на заданное количество шагов. """  
 def predict(self, count\_steps: int, previous: list):  
 out = []  
 for i in range(count\_steps):  
 previous = previous[1:] + [self.iteration([1] + previous)] # сдвигаем окно и добавляем предсказание  
 out.append(previous[-1])  
 return out  
  
 """ Обучает нейронную сеть на основе предоставленных обучающих данных. """  
 def learning(self, p, norm\_learning):  
 error = 0  
 for i in range(len(self.sample)-1):  
 err = self.sample[i+1][-1] - self.iteration(self.sample[i])  
 for j in range(p+1):  
 self.synapses[j] += norm\_learning \* err \* self.sample[i][j]  
 error += (err\*\*2)  
 self.errors.append(self.get\_err())  
 # self.errors.append(math.sqrt(error))  
  
 def get\_err(self):  
 predicted = self.predict(self.N-self.p+self.end, [func(self.a + (self.b-self.a)/(self.N-1) \* i) for i in range(self.p)])  
 ideal = [func(self.a + (self.b-self.a)/(self.N-1) \* i) for i in range(self.p, self.N+self.end)]  
 # ideal = ideal[-len(predicted):]  
 # print(ideal)  
 # print(predicted)  
 return math.sqrt(sum([(ideal[i] - predicted[i])\*\*2 for i in range(len(predicted))])/self.p)  
  
def func(x):  
 return (float)(0.5 \* math.exp(0.5 \* math.cos(0.5 \* x)) + math.sin(0.5 \* x))  
  
""" обучающая выборка.создаются списки Y(x) со сдвигом на 1 каждый """  
def create\_sample(m, p, X, out=False):  
 sample = []  
 # создаем выборку  
 for i in range(m-p):  
 sample.append([1]) # Добавляем единицу в качестве первого элемента каждого образца  
 for j in range(p):  
 sample[i].append(X[j+i]) # Добавляем p элементов из входного массива X в образец  
 # вывод выборки при необходимости в читаемом виде  
 if out:  
 for i in range(m-p):  
 print(sample[i]) # Выводим образцы, если параметр out равен True  
 return sample # Возвращаем сформированную выборку  
  
def main():  
 a = -5  
 b = 3  
 p = 10 # Размер окна  
 NORM\_LEARNING = 0.5 # Норма обучения  
 ITERATIONS = 1000 # Количество итераций обучения  
  
 N = 20 # Количество шагов  
 end = 2 \* b - a # Конец прогнозируемой функции  
 T = end  
 step = (b-a)/(N-1) # Расстояние между t  
 X = [func(a + step \* i) for i in range(N)] # Список прогнозируемых точек  
  
 Net = Network([0] \* (p + 1), create\_sample(N, p, X), N, p, T, a, b) # Создаем НС  
 for i in range(ITERATIONS):  
 Net.learning(p, NORM\_LEARNING) # Обучаем НС  
 print("Размер конечной ошибки: {}".format(round(Net.errors[-1], 3))) # Выводим размер конечной ошибки  
  
 x = [(a + step \* i) for i in range(N+T)]  
 y = Net.predict(N+T-p, [func(a + step \* i) for i in range(p)]) # Предсказываем значения  
  
 # Добавляем значения из окна для красивого вывода  
 y = [func(a + step \* i) for i in range(p)] + y  
  
 plt.plot(x, y, 'bo') # Выводим предсказанные точки  
 y = [func(a + step \* i) for i in range(N+T)]  
 plt.plot(x, y) # Выводим исходную функцию  
 plt.show() # Отображаем график  
  
 plt.semilogy([i for i in range(len(Net.errors))], Net.errors) # Строим график логарифмической ошибки  
 plt.show() # Отображаем график  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()