МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

Специальность АС-66

Выполнил
А. С. Рогожин,
студент группы АС-66

Проверил
А. А. Крощенко,
доцент кафедры ИИТ,
«___» _____2025 г.

Брест 2025

Цель: выполнить моделирование прогнозирующей РНС.
Вариант 10:

Задание:
1. По вариантам предыдущей лабораторной работы реализовать предложенный вариант рекуррентной нейронной сети. Сравнить полученные результаты с ЛР 5.
Варианты заданий приведены в следующей таблице:

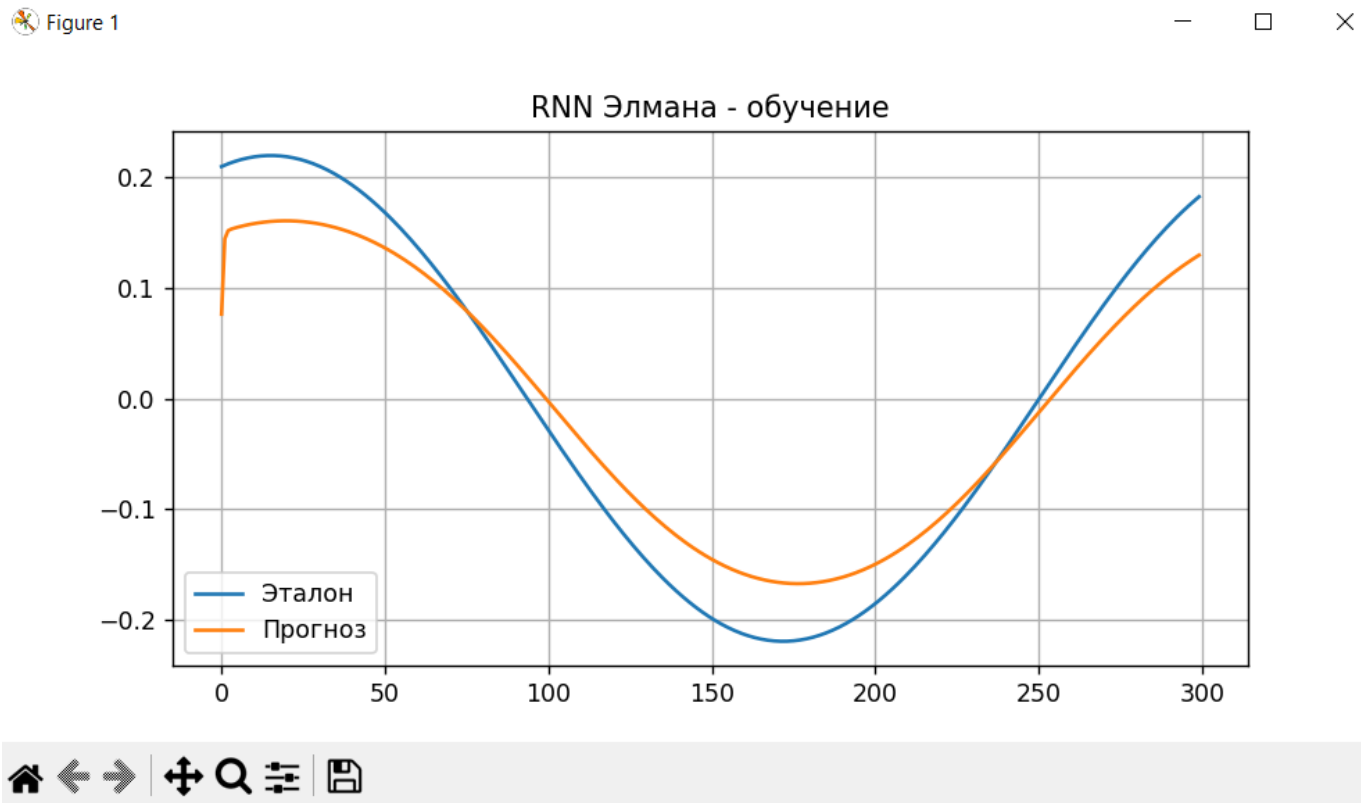
№	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое	Тип РНС
1	0.1	0.1	0.05	0.1	6	2	Элмана
2	0.2	0.2	0.06	0.2	8	3	Джордана
3	0.3	0.3	0.07	0.3	10	4	Мультирекуррентная
4	0.4	0.4	0.08	0.4	6	2	Элмана
5	0.1	0.5	0.09	0.5	8	3	Джордана
6	0.2	0.6	0.05	0.6	10	4	Мультирекуррентная
7	0.3	0.1	0.06	0.1	6	2	Элмана
8	0.4	0.2	0.07	0.2	8	3	Джордана
9	0.1	0.3	0.08	0.3	10	4	Мультирекуррентная
10	0.2	0.4	0.09	0.4	6	2	Элмана
11	0.3	0.5	0.05	0.5	8	3	Джордана

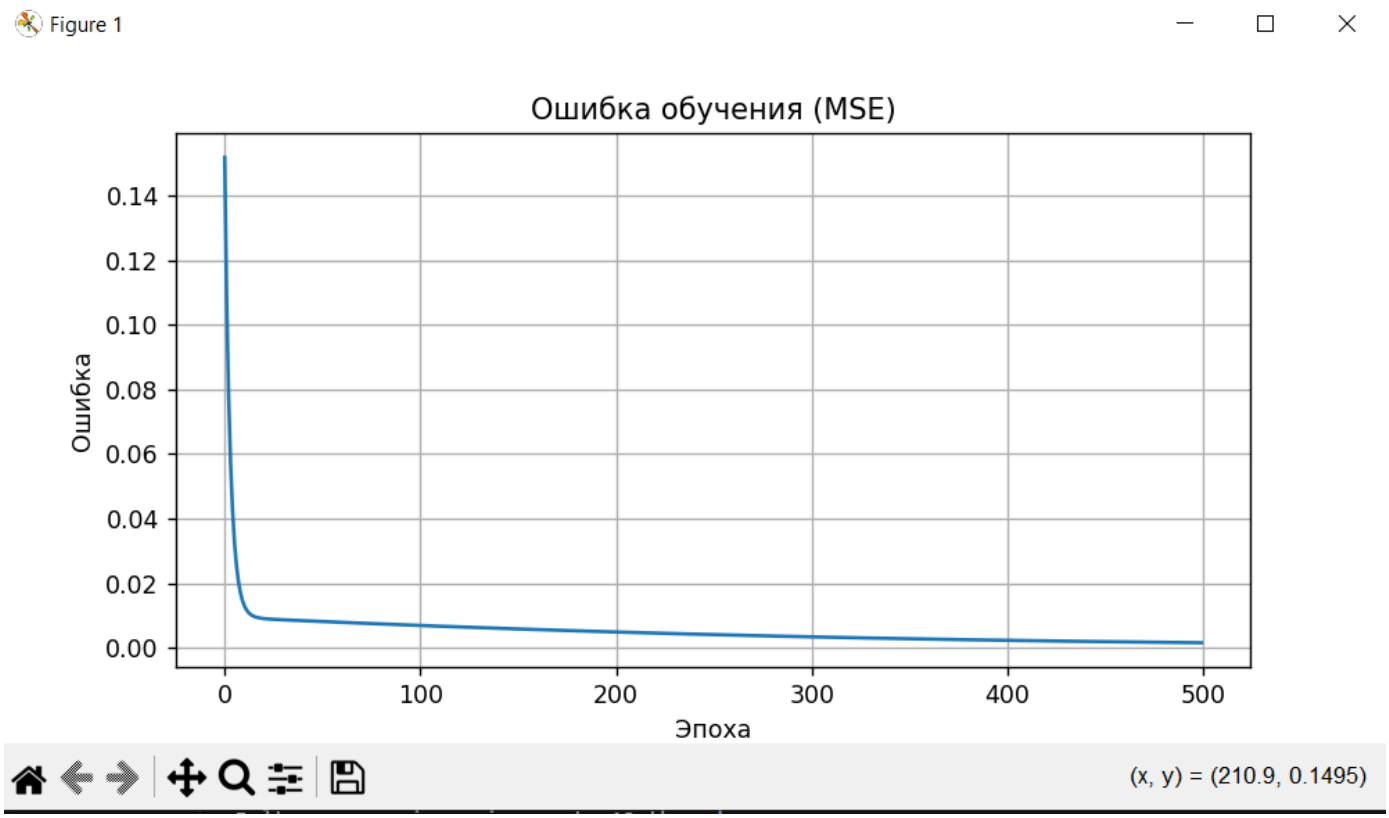
В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

```
import numpy as np
import matplotlib.pyplot as plt

a, b, c, d = 0.2, 0.4, 0.09, 0.4

def func(x):
    return a * np.cos(b * x) + c * np.sin(d * x)

x = np.linspace(0, 20, 400)
y = func(x)

window = 6
hidden = 2

X, Y = [], []
for i in range(len(y) - window):
    X.append(y[i:i+window])
    Y.append(y[i+window])
X = np.array(X)
Y = np.array(Y)

train_size = 300
X_train, X_test = X[:train_size], X[train_size:]
Y_train, Y_test = Y[:train_size], Y[train_size:]

np.random.seed(0)
Wx = np.random.randn(window, hidden) * 0.5
```

```
Wh = np.random.randn(hidden, hidden) * 0.5
b1 = np.zeros(hidden)

Wy = np.random.randn(hidden) * 0.5
b2 = 0.0

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

lr = 0.05
epochs = 500
errors = []

for epoch in range(epochs):
    h_prev = np.zeros(hidden)
    y_preds = []
    hs = []

    for t in range(len(X_train)):
        x_t = X_train[t]
        h = sigmoid(x_t @ Wx + h_prev @ Wh + b1)
        y_pred = h @ Wy + b2
        y_preds.append(y_pred)
        hs.append(h)
        h_prev = h

    y_preds = np.array(y_preds)
    err = y_preds - Y_train
    mse = np.mean(err ** 2)
    errors.append(mse)

    dWx = np.zeros_like(Wx)
    dWh = np.zeros_like(Wh)
    db1 = np.zeros_like(b1)
    dWy = np.zeros_like(Wy)
    db2 = 0.0
    dh_next = np.zeros(hidden)

    for t in reversed(range(len(X_train))):
        h = hs[t]
        h_prev = hs[t-1] if t > 0 else np.zeros(hidden)
        dy = err[t]

        dWy += h * dy * (2 / len(X_train))
        db2 += dy * 2 / len(X_train)

        dh = Wy * dy * 2 / len(X_train) + dh_next
        dz = dh * h * (1 - h)

        dWx += np.outer(X_train[t], dz)
        dWh += np.outer(h_prev, dz)
        db1 += dz

        dh_next = Wh @ dz

    Wx -= lr * dWx
    Wh -= lr * dWh
    b1 -= lr * db1
    Wy -= lr * dWy
    b2 -= lr * db2

def rnn_predict(X):
    h_prev = np.zeros(hidden)
    y_preds = []
    for t in range(len(X)):
        h = sigmoid(X[t] @ Wx + h_prev @ Wh + b1)
        y_pred = h @ Wy + b2
        y_preds.append(y_pred)
        h_prev = h
    return np.array(y_preds)

train_pred = rnn_predict(X_train)
```

```
test_pred = rnn_predict(X_test)

plt.figure(figsize=(8,4))
plt.plot(Y_train, label="Эталон")
plt.plot(train_pred, label="Прогноз")
plt.title("RNN Элмана - обучение")
plt.legend()
plt.grid()
plt.show()

plt.figure(figsize=(8,4))
plt.plot(errors)
plt.title("Ошибка обучения (MSE)")
plt.xlabel("Эпоха")
plt.ylabel("Ошибка")
plt.grid()
plt.show()

def print_table(name, Y_true, Y_pred, limit=20):
    print("\n-----", name, "-----")
    print("Эталон\t\tПрогноз\t\tОтклонение")
    for t, p in list(zip(Y_true, Y_pred))[:limit]:
        print(f"{t:.6f}\t{p:.6f}\t{t - p:.6f}")

print_table("ОБУЧЕНИЕ", Y_train, train_pred)
print_table("ПРОГНОЗ", Y_test, test_pred)
```






```
PS D:\Uni\OMO\ml_as66\reports> & C:/Python/

----- ОБУЧЕНИЕ -----
Эталон          Прогноз         Отклонение
0.209355        0.076047        0.133308
0.210624        0.144106        0.066517
0.211807        0.151453        0.060354
0.212905        0.152938        0.059967
0.213918        0.153849        0.060069
0.214844        0.154660        0.060184
0.215685        0.155414        0.060270
0.216438        0.156114        0.060324
0.217105        0.156761        0.060344
0.217684        0.157354        0.060331
0.218176        0.157893        0.060283
0.218580        0.158380        0.060200
0.218896        0.158813        0.060083
0.219125        0.159193        0.059931
0.219265        0.159520        0.059745
0.219317        0.159794        0.059523
0.219281        0.160015        0.059266
0.219156        0.160183        0.058974
0.218944        0.160298        0.058646
0.218643        0.160359        0.058284
```

```
----- ПРОГНОЗ -----
Эталон          Прогноз         Отклонение
0.184560        0.056611        0.127949
0.186899        0.125609        0.061290
0.189162        0.133975        0.055187
0.191349        0.136233        0.055116
0.193459        0.137873        0.055586
0.195491        0.139407        0.056084
0.197445        0.140883        0.056562
0.199320        0.142304        0.057016
0.201114        0.143671        0.057443
0.202828        0.144983        0.057844
0.204460        0.146242        0.058218
0.206009        0.147446        0.058563
0.207476        0.148596        0.058880
0.208860        0.149692        0.059167
0.210159        0.150735        0.059425
0.211374        0.151723        0.059651
0.212505        0.152658        0.059847
0.213549        0.153538        0.060011
0.214508        0.154365        0.060143
0.215381        0.155139        0.060242
PS D:\Uni\OMO\ml_as66\reports>
```

Вывод: выполнили моделирование прогнозирующей РНС для прогнозирования функции a*cosbx+c*sindx.