Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
Кафедра интеллектуальных информационных технологий

# Отчет по лабораторной работе №5

Выполнил:
Студент 3-го курса
Группы АС-66
Горобец А.В.
Проверил
Крощенко А.А.

Брест 2025

Цель работы: Выполнить моделирование прогнозирующей нелинейной ИНС.

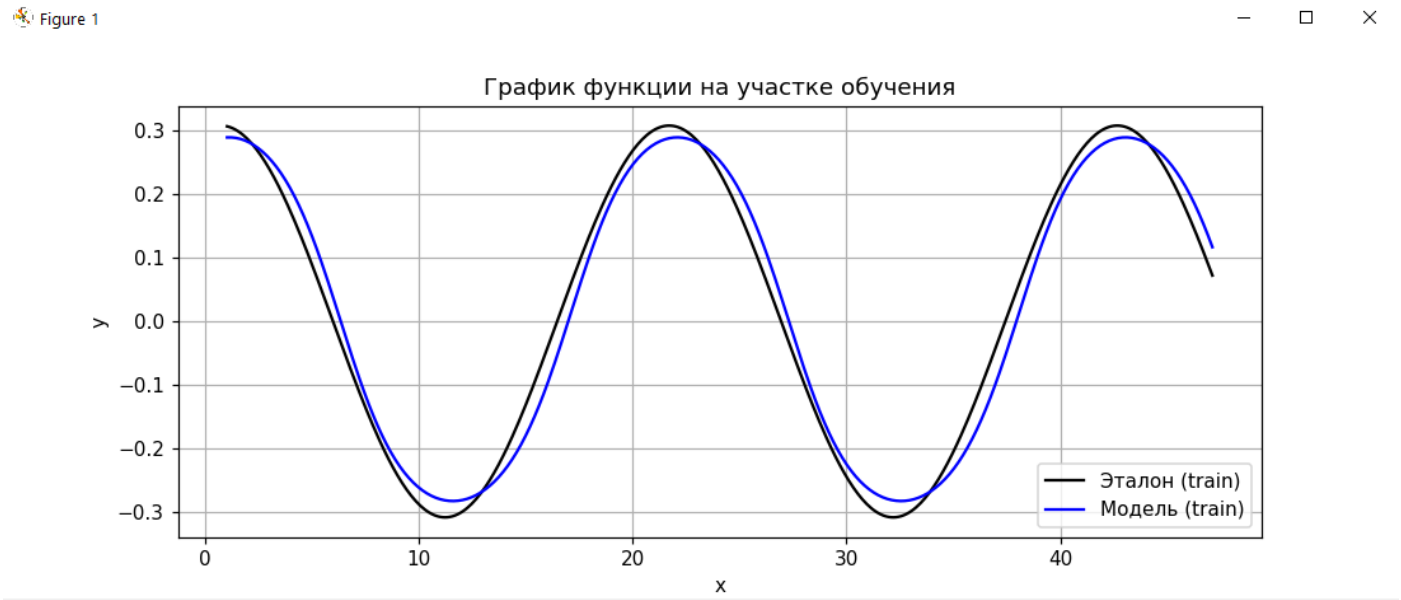
Вариант 3

| № вари-анта | a | b | c | d | Кол-во входов ИНС | Кол-во НЭ в скрытом слое |
|---|---|---|---|---|---|---|
| 3 | 0.3 | 0.3 | 0.07 | 0.3 | 10 | 4 |

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

a, b, c, d = 0.3, 0.3, 0.07, 0.3
L = 10
H = 4
epochs = 2000
lr = 0.05
test_frac = 0.25

N = 600
x = np.linspace(0, 20*np.pi, N)
y = a * np.cos(b * x) + c * np.sin(d * x)

split_idx = int(N * (1 - test_frac))
x_train, x_test = x[:split_idx], x[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

def make_dataset(y_series, L):
    X, Y = [], []
    for i in range(L, len(y_series)):
        X.append(y_series[i-L:i])
        Y.append(y_series[i])
    return np.array(X), np.array(Y).reshape(-1, 1)

X_train, Y_train = make_dataset(y_train, L)
X_test, Y_test = make_dataset(y_test, L)

x_train_targets = x_train[L:]
x_test_targets = x_test[L:]

mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X_train = (X_train - mean) / std
X_test = (X_test - mean) / std

def sigmoid(z):
```

```
    z_clip = np.clip(z, -50, 50)
    return 1.0 / (1.0 + np.exp(-z_clip))

def sigmoid_deriv(a_sigmoid):
    return a_sigmoid * (1.0 - a_sigmoid)

rng = np.random.default_rng(42)
W1 = rng.normal(0.0, np.sqrt(1.0 / L), size=(L, H))
b1 = np.zeros((1, H))
W2 = rng.normal(0.0, np.sqrt(1.0 / H), size=(H, 1))
b2 = np.zeros((1, 1))

def forward(X, W1, b1, W2, b2):
    z1 = X @ W1 + b1
    a1 = sigmoid(z1)
    z2 = a1 @ W2 + b2
    return z2, (X, z1, a1, z2)

def backward(W1, b1, W2, b2, cache, y_pred, Y, lr):
    X, z1, a1, z2 = cache
    N = X.shape[0]
    err = y_pred - Y
    grad_out = 2.0 * err / N

    dW2 = a1.T @ grad_out
    db2 = np.sum(grad_out, axis=0, keepdims=True)

    da1 = grad_out @ W2.T
    dz1 = da1 * sigmoid_deriv(a1)
    dW1 = X.T @ dz1
    db1 = np.sum(dz1, axis=0, keepdims=True)

    W1 -= lr * dW1
    b1 -= lr * db1
    W2 -= lr * dW2
    b2 -= lr * db2

    loss = np.mean(err**2)
    return W1, b1, W2, b2, loss

loss_history = []
for epoch in range(1, epochs+1):
    y_pred, cache = forward(X_train, W1, b1, W2, b2)
    W1, b1, W2, b2, loss = backward(W1, b1, W2, b2, cache, y_pred, Y_train, lr)
    loss_history.append(loss)
    if epoch % 500 == 0 or epoch == 1:
        print(f"Epoch {epoch:4d} | MSE={loss:.6f}")
```

```
def predict(X, W1, b1, W2, b2):
    z1 = X @ W1 + b1
    a1 = sigmoid(z1)
    return a1 @ W2 + b2

train_pred = predict(X_train, W1, b1, W2, b2)
test_pred = predict(X_test, W1, b1, W2, b2)

train_df = pd.DataFrame({
    "Эталонное значение": Y_train.flatten(),
    "Полученное значение": train_pred.flatten(),
    "Отклонение": (train_pred - Y_train).flatten()
})
test_df = pd.DataFrame({
    "Эталонное значение": Y_test.flatten(),
    "Полученное значение": test_pred.flatten(),
    "Отклонение": (test_pred - Y_test).flatten()
})

print("\nТаблица обучения (первые 10 строк):")
print(train_df.head(10))
print("\nТаблица прогноза (первые 10 строк):")
print(test_df.head(10))

plt.figure(figsize=(10,4))
plt.plot(x_train_targets, Y_train, label="Эталон (train)", color="black")
plt.plot(x_train_targets, train_pred, label="Модель (train)", color="blue")
plt.title("График функции на участке обучения")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(8,4))
plt.plot(loss_history, color="red")
plt.title("Изменение ошибки по эпохам")
plt.xlabel("Эпоха")
plt.ylabel("MSE")
plt.grid(True)
plt.show()
```





Вывод: выполнил моделирование прогнозирующей нелинейной ИНС. Генерировал обучающие и тестовые данных используя заданную функцию.