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

Отчет по лабораторной работе №5

Выполнил
В.Д.Головкина,
студент группы АС66
Проверил
А. А. Крощенко,
доц. кафедры ИИТ,
« __ » _____2025 г.

Брест 2025

Цель работы: изучить и выполнить моделирование прогнозирующей нелинейной ИНС.

Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию. Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

$$y = a\cos(bx) + c\sin(dx) \ .$$

Варианты заданий приведены в следующей таблице:

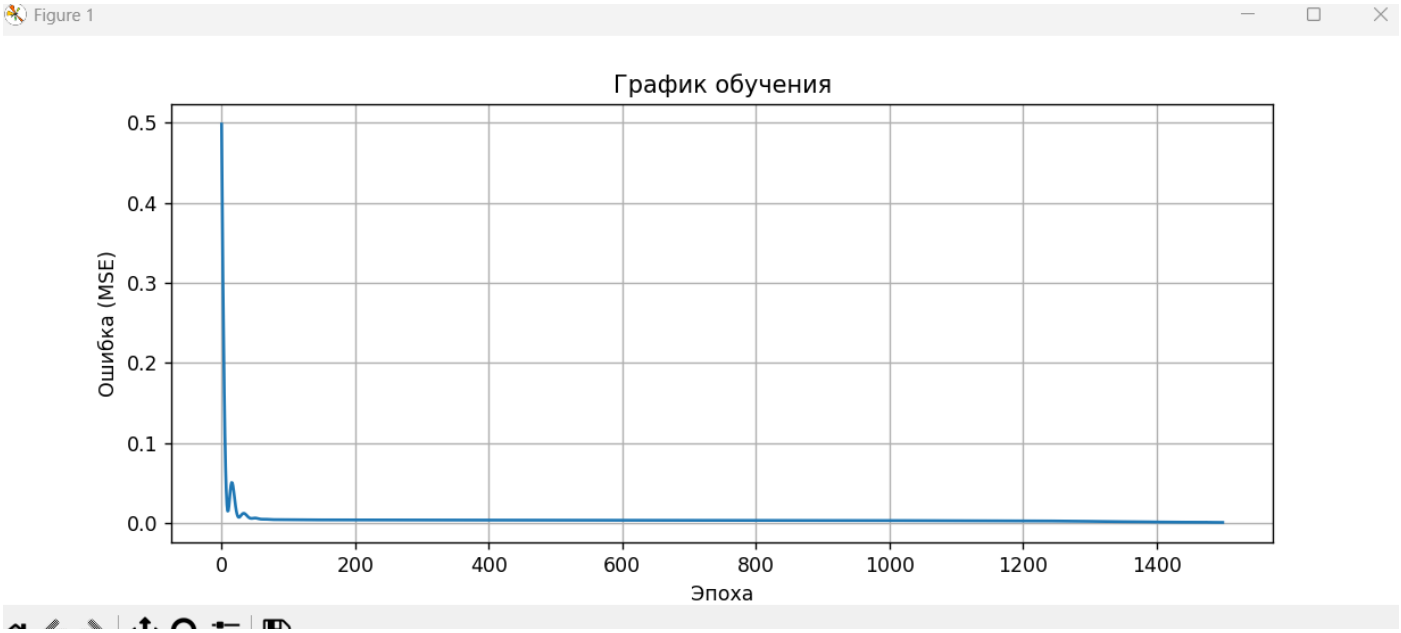
№ варианта	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое
1	0.1	0.1	0.05	0.1	6	2

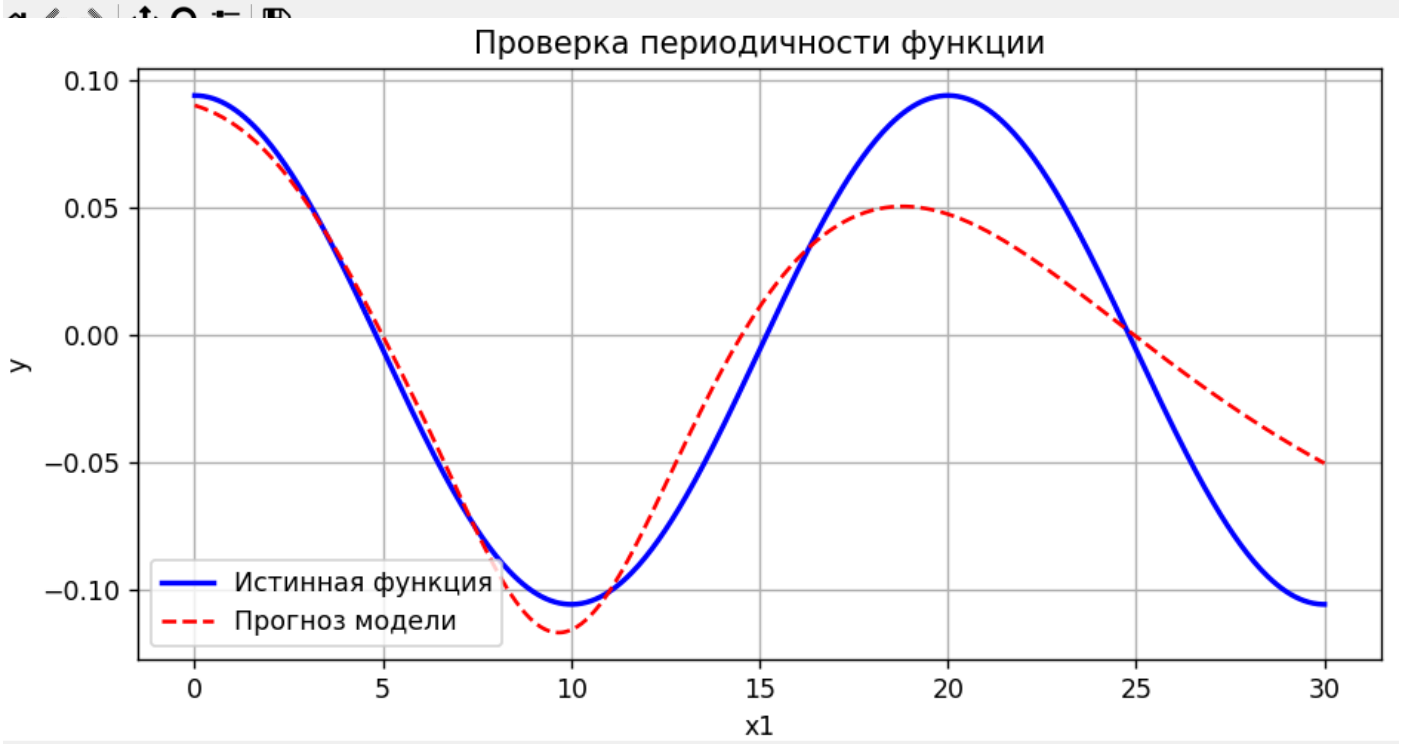
```
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

a, b, c, d = 0.1, 0.1, 0.05, 0.1
n_inputs = 2
n_hidden = 12
n_samples = 1000

def target_function(X):
    x1, x2 = X[:, 0], X[:, 1]
    return a * np.cos(np.pi * b * x1) + c * np.sin(np.pi * d * x2)

np.random.seed(42)
periods_x1 = 3
periods_x2 = 2

X = np.zeros((n_samples, n_inputs))
X[:, 0] = np.random.uniform(0, periods_x1 / b, n_samples)
X[:, 1] = np.random.uniform(0, periods_x2 / d, n_samples)
y = target_function(X)

split = int(0.8 * n_samples)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test, dtype=torch.float32).view(-1, 1)

class SimpleMLP(nn.Module):
    def __init__(self):
```

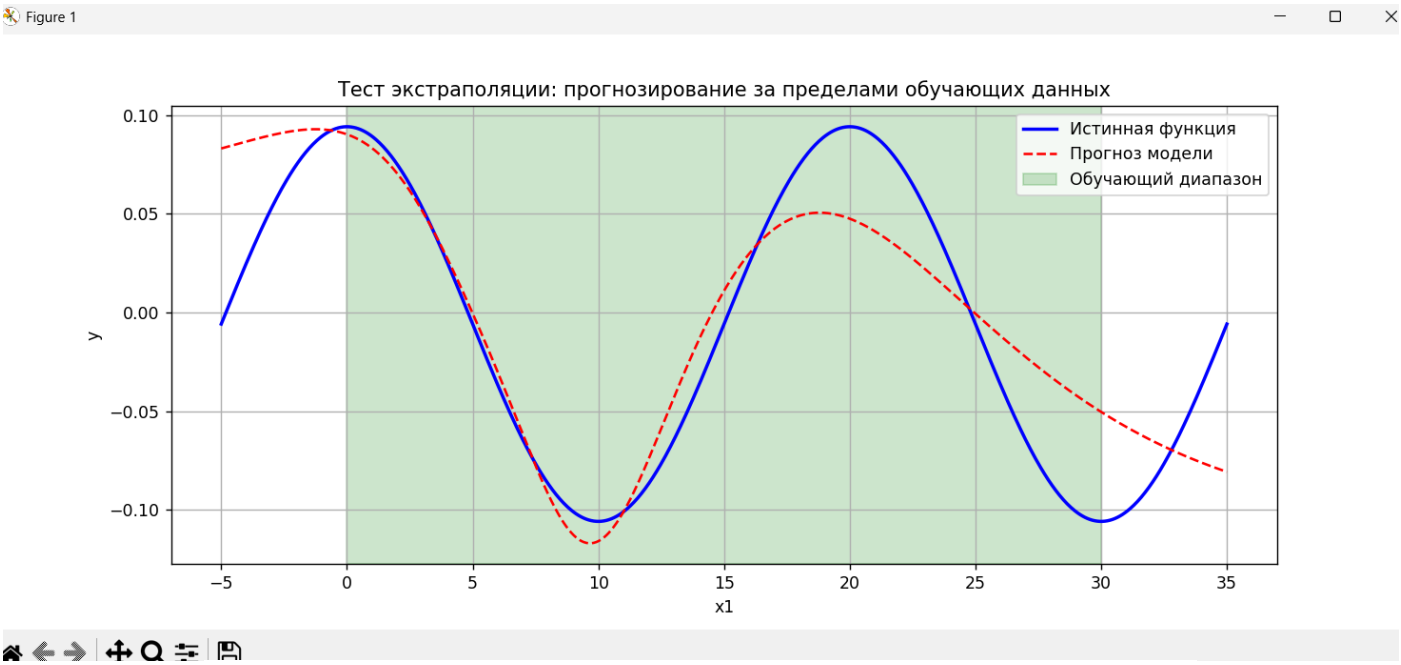
```
        super(SimpleMLP, self).__init__()
        self.hidden = nn.Linear(n_inputs, n_hidden)
        self.sigmoid = nn.Sigmoid()
        self.output = nn.Linear(n_hidden, 1)  # Линейная активация на выходе

    def forward(self, x):
        x = self.sigmoid(self.hidden(x))
        return self.output(x)

model = SimpleMLP()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
n_epochs = 1500

losses = []
print("Обучение модели...")
for epoch in range(n_epochs):
    model.train()
    y_pred = model(X_train_tensor)
    loss = criterion(y_pred, y_train_tensor)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    losses.append(loss.item())

    if epoch % 200 == 0:
        print(f"Эпоха {epoch}, Ошибка: {loss.item():.6f}")

print(f"Обучение завершено. Финальная ошибка: {losses[-1]:.6f}")

plt.figure(figsize=(10, 4))
plt.plot(losses)
plt.title('График обучения')
plt.xlabel('Эпоха')
plt.ylabel('Ошибка (MSE)')
plt.grid(True)
plt.show()

print("\n" + "="*50)
print("ТЕСТИРОВАНИЕ МОДЕЛИ")
print("="*50)

model.eval()

with torch.no_grad():
    y_train_pred = model(X_train_tensor).numpy().flatten()
    y_test_pred = model(X_test_tensor).numpy().flatten()

mse_test = mean_squared_error(y_test, y_test_pred)
mae_test = mean_absolute_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)
```

```
print(f"Метрики качества на тестовой выборке:")
print(f"MSE: {mse_test:.8f}")
print(f"MAE: {mae_test:.8f}")
print(f"R²: {r2_test:.6f}")

plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
plt.plot(y_test, 'b-', label='Эталон', linewidth=2, alpha=0.8)
plt.plot(y_test_pred, 'r--', label='Прогноз', linewidth=1.5)
plt.title('Тестовая выборка: сравнение прогноза и эталона')
plt.xlabel('Номер примера')
plt.ylabel('Значение')
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 4)
x1_test = np.linspace(0, periods_x1 / b, 500)
x2_fixed = np.median(X[:, 1])
X_periodic = np.column_stack([x1_test, np.full_like(x1_test, x2_fixed)])
X_periodic_tensor = torch.tensor(X_periodic, dtype=torch.float32)

with torch.no_grad():
    y_periodic_pred = model(X_periodic_tensor).numpy().flatten()

y_periodic_true = target_function(X_periodic)

plt.plot(x1_test, y_periodic_true, 'b-', label='Истинная функция', linewidth=2)
plt.plot(x1_test, y_periodic_pred, 'r--', label='Прогноз модели', linewidth=1.5)
plt.title('Проверка периодичности функции')
plt.xlabel('x1')
plt.ylabel('y')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

x1_extended = np.linspace(-5, 35, 800)
x2_fixed = np.median(X[:, 1])
X_extended = np.column_stack([x1_extended, np.full_like(x1_extended, x2_fixed)])
X_extended_tensor = torch.tensor(X_extended, dtype=torch.float32)

with torch.no_grad():
    y_extended_pred = model(X_extended_tensor).numpy().flatten()

y_extended_true = target_function(X_extended)

plt.figure(figsize=(12, 5))
plt.plot(x1_extended, y_extended_true, 'b-', label='Истинная функция', linewidth=2)
plt.plot(x1_extended, y_extended_pred, 'r--', label='Прогноз модели', linewidth=1.5)
plt.axvspan(0, periods_x1 / b, alpha=0.2, color='green', label='Обучающий диапазон')
plt.title('Тест экстраполяции: прогнозирование за пределами обучающих данных')
plt.xlabel('x1')
```

```
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```








```
PS C:\Users\Пипка> & C:/Users/Пипка/AppData/Local/Prog
Обучение модели...
Эпоха 0, Ошибка: 0.498399
Эпоха 200, Ошибка: 0.003814
Эпоха 400, Ошибка: 0.003633
Эпоха 600, Ошибка: 0.003490
Эпоха 800, Ошибка: 0.003338
Эпоха 1000, Ошибка: 0.003067
Эпоха 1200, Ошибка: 0.002669
Эпоха 1400, Ошибка: 0.001223
Обучение завершено. Финальная ошибка: 0.000702

==================================================
ТЕСТИРОВАНИЕ МОДЕЛИ
==================================================
Метрики качества на тестовой выборке:
MSE: 0.00072018
MAE: 0.02284899
R²: 0.893736
PS C:\Users\Пипка> & C:/Users/Пипка/AppData/Local/Prog
```

Вывод: я изучила и выполнила моделирование прогнозирующей нелинейной ИНС.