МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5

Специальность АС-66

Выполнила
Е. С. Неруш,
студент группы АС-66

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

Брест 2025

Цель работы: На практике изучить Нелинейные ИНС в задачах регрессии.

Задачи:

1. Выполнить моделирование прогнозирующей нелинейной ИНС. Для генерации обучающих и тестовых данных использовать функцию

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

Варианты заданий приведены в следующей таблице:

Вариант 10

a=0.2, b=0.4, c=0.09, d=0.4
Кол-во входов ИНС - 6
Кол-во НЭ в скрытом слое - 2
Для прогнозирования использовать многослойную ИНС с одним скрытым слоем. В качестве функций активации для скрытого слоя использовать сигмоидную функцию, для выходного - линейную.

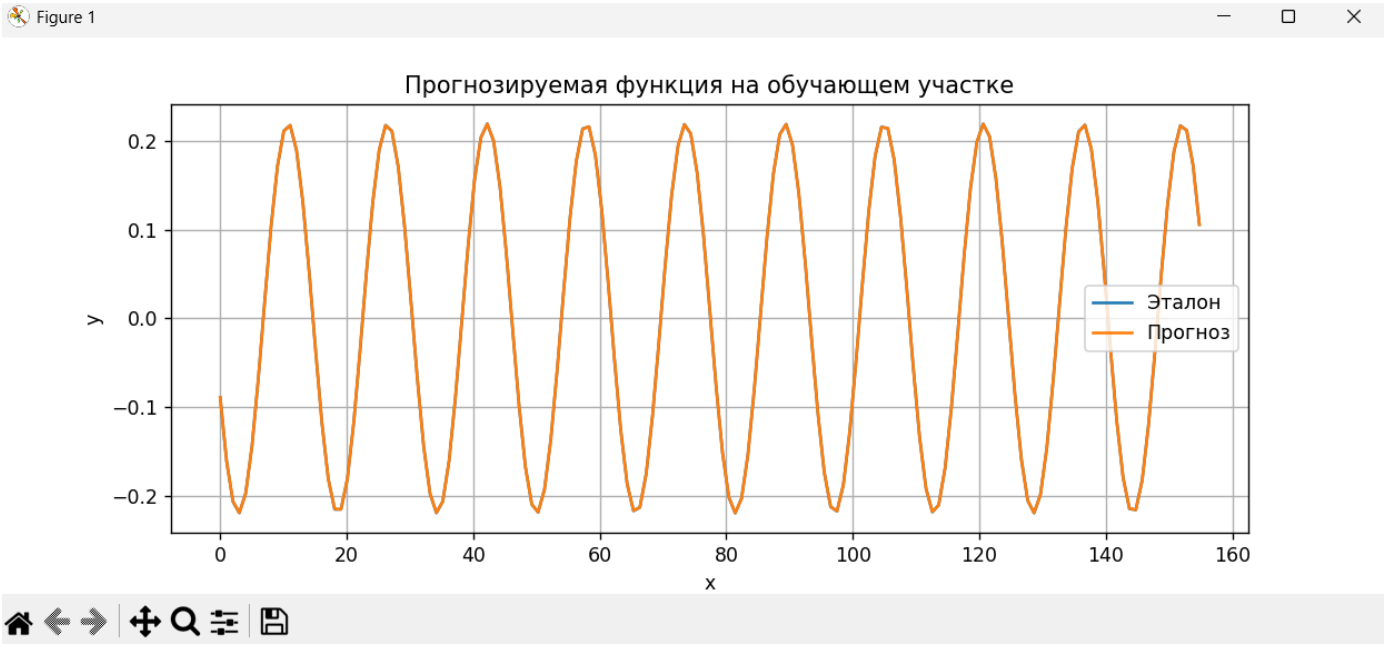
```
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import pandas as pd

# 1. Генерация обучающих и тестовых данных
def target_function(x, a=0.2, b=0.4, c=0.09, d=0.4):
    return a * torch.cos(b * x) + c * torch.sin(d * x)

a, b, c, d = 0.2, 0.4, 0.09, 0.4
input_size = 6
hidden_size = 2
num_samples = 200
train_ratio = 0.8

x = torch.linspace(0, 200, num_samples).reshape(-1, 1)
y = target_function(x, a, b, c, d)

X = torch.cat([y[i:num_samples - input_size + i] for i in range(input_size)],
dim=1)
y_target = y[input_size:]

split_idx = int(X.shape[0] * train_ratio)
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y_target[:split_idx], y_target[split_idx:]

print("Данные сгенерированы для функции y = a*cos(bx) + c*sin(dx)")
print(f"Параметры: a={a}, b={b}, c={c}, d={d}")
print(f"Размер обучающей выборки: {X_train.shape}, тестовой: {X_test.shape}")
```

```
Данные сгенерированы для функции y = a*cos(bx) + c*sin(dx)
Параметры: a=0.2, b=0.4, c=0.09, d=0.4
Размер обучающей выборки: torch.Size([160, 6]), тестовой: torch.Size([40, 6])
```

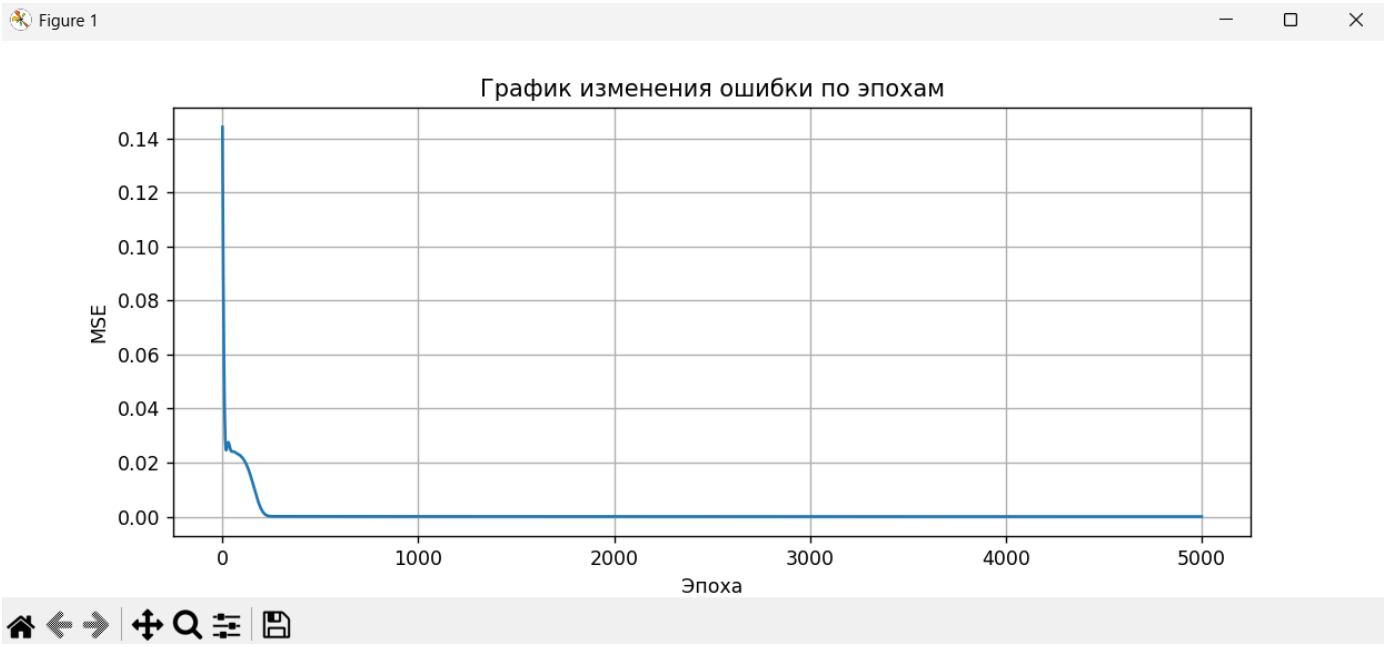
```
# 2. Архитектура ИНС
class NonlinearRegressor(nn.Module):
    def __init__(self, input_dim, hidden_dim):
        super().__init__()
        self.hidden = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.Sigmoid()
        )
        self.output = nn.Linear(hidden_dim, 1)

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

model = NonlinearRegressor(input_size, hidden_size)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

# 3. Обучение модели
losses = []
epochs = 5000

for epoch in range(epochs):
    model.train()
    y_pred = model(X_train)
    loss = criterion(y_pred, y_train)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    losses.append(loss.item())

print("\nОбучение завершено")
print(f"Минимальная ошибка достигнута при a={a}, финальный
MSE={min(losses):.6f}")
```

```
Обучение завершено
Минимальная ошибка достигнута при a=0.2, финальный MSE=0.000000
```

```
# 4. График прогнозируемой функции на обучающем участке
model.eval()
with torch.no_grad():
    y_train_pred = model(X_train)

plt.figure(figsize=(10, 4))
plt.plot(x[:split_idx], y_train, label="Эталон")
plt.plot(x[:split_idx], y_train_pred, label="Прогноз")
plt.title("Прогнозируемая функция на обучающем участке")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid(True)
plt.show()
```




```
# 5. Результаты обучения: таблица и график ошибки
train_table = pd.DataFrame({
    "Эталонное значение": y_train.squeeze().numpy(),
    "Полученное значение": y_train_pred.squeeze().numpy(),
    "Отклонение": (y_train_pred - y_train).squeeze().numpy()
})
print("\nРезультаты обучения (первые строки):")
print(train_table.head())

plt.figure(figsize=(10, 4))
plt.plot(losses)
plt.title("График изменения ошибки по эпохам")
plt.xlabel("Эпоха")
plt.ylabel("MSE")
plt.grid(True)
plt.show()
```




```
# 6. Результаты прогнозирования
with torch.no_grad():
```

```
    y_test_pred = model(X_test)
    test_table = pd.DataFrame({
        "Эталонное значение": y_test.squeeze().numpy(),
        "Полученное значение": y_test_pred.squeeze().numpy(),
        "Отклонение": (y_test_pred - y_test).squeeze().numpy()
    })
print("\nРезультаты прогнозирования (первые строки):")
print(test_table.head())
```

```
Результаты обучения (первые строки):
   Эталонное значение  Полученное значение  Отклонение
0           -0.089110            -0.089042    0.000068
1           -0.160416            -0.160797   -0.000382
2           -0.206143            -0.205933    0.000211
3           -0.219002            -0.218756    0.000245
4           -0.196941            -0.197309   -0.000368
```

```
Результаты прогнозирования (первые строки):
   Эталонное значение  Полученное значение    Отклонение
0            0.022305             0.022443  1.376793e-04
1           -0.064840            -0.064570  2.703294e-04
2           -0.141646            -0.142013 -3.663749e-04
3           -0.195868            -0.195868  2.682209e-07
4           -0.218859            -0.218486  3.733784e-04
```

Вывод: Модель многослойной нейронной сети точно аппроксимировала заданную функцию, достигнув минимальной ошибки при a=0.2, а результаты обучения и прогнозирования показали устойчивую и согласованную работу ИНС.