Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5
По дисциплине: «ОМО»
Тема:” Нелинейные ИНС в задачах регрессии”

Выполнил:
Студент 3-го курса
Группы АС-66
Езепчук А.С.
Проверил:
Крощенко А.А.

Брест 2025

Цель: Выполнить моделирование прогнозирующей нелинейной ИНС.

Вариант 3

Задание:

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:

№ 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
2	0.2	0.2	0.06	0.2	8	3
3	0.3	0.3	0.07	0.3	10	4
4	0.4	0.4	0.08	0.4	6	2
5	0.1	0.5	0.09	0.5	8	3
6	0.2	0.6	0.05	0.6	10	4
7	0.3	0.1	0.06	0.1	6	2
8	0.4	0.2	0.07	0.2	8	3
9	0.1	0.3	0.08	0.3	10	4
10	0.2	0.4	0.09	0.4	6	2
11	0.3	0.5	0.05	0.5	8	3

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

a, b, c, d = 0.3, 0.3, 0.07, 0.3
window_size = 10
hidden_size = 4
epochs = 1000
lr = 0.05
train_ratio = 0.7

OUT_DIR = "lab5_results"
os.makedirs(OUT_DIR, exist_ok=True)

def sigmoid(x):
    x_clip = np.clip(x, -50, 50)
    return 1.0 / (1.0 + np.exp(-x_clip))

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

def generate_series(a, b, c, d, x_min=-50, x_max=50, step=0.01):
    x_vals = np.arange(x_min, x_max + step, step)
    y_vals = a * np.cos(b * x_vals) + c * np.sin(d * x_vals)
    return x_vals, y_vals
```

```
def make_supervised_from_series(y_vals, window):
    X, Y = [], []
    for i in range(len(y_vals) - window):
        X.append(y_vals[i:i + window])
        Y.append(y_vals[i + window])
    X = np.array(X, dtype=np.float64)
    Y = np.array(Y, dtype=np.float64).reshape(-1, 1)
    return X, Y

def init_weights(input_size, hidden_size, output_size=1, seed=42):
    rng = np.random.default_rng(seed)
    W1 = rng.normal(0.0, np.sqrt(1.0 / input_size), size=(input_size, hidden_size))
    b1 = np.zeros((1, hidden_size), dtype=np.float64)
    W2 = rng.normal(0.0, np.sqrt(1.0 / hidden_size), size=(hidden_size, output_size))
    b2 = np.zeros((1, output_size), dtype=np.float64)
    return W1, b1, W2, b2

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

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

def compute_mse(y_pred, y_true):
    err = y_pred - y_true
    return np.mean(err ** 2), err

def backward_and_update(W1, b1, W2, b2, cache, err, lr):
    X, z1, a1, z2 = cache
    N = X.shape[0]

    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_from_activation(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
```

```
    return W1, b1, W2, b2

def train(X_train, y_train, W1, b1, W2, b2, epochs, lr):
    loss_history = []
    for epoch in range(1, epochs + 1):
        y_pred, cache = forward(X_train, W1, b1, W2, b2)
        loss, err = compute_mse(y_pred, y_train)
        loss_history.append(loss)

        W1, b1, W2, b2 = backward_and_update(W1, b1, W2, b2, cache, err, lr)

        if epoch % 250 == 0 or epoch == 1:
            print(f"Epoch {epoch:4d} | MSE={loss:.8f}")

    return W1, b1, W2, b2, loss_history

def pretty_style():
    plt.style.use("seaborn-v0_8")
    plt.rcParams.update({
        "figure.dpi": 150,
        "font.size": 10,
        "lines.linewidth": 2.0,
        "axes.titleweight": "bold",
        "axes.labelsize": 11,
        "legend.frameon": True,
        "legend.framealpha": 0.9,
        "legend.edgecolor": "0.8"
    })

def main():
    pretty_style()

    x_vals, y_vals = generate_series(a, b, c, d, x_min=-50, x_max=50)

    X, Y = make_supervised_from_series(y_vals, window_size)
    split = int(train_ratio * len(X))
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = Y[:split], Y[split:]

    W1, b1, W2, b2 = init_weights(window_size, hidden_size, output_size=1, seed=42)
    W1, b1, W2, b2, loss_history = train(X_train, y_train, W1, b1, W2, b2, epochs, lr)

    y_train_pred = predict(X_train, W1, b1, W2, b2)
    y_test_pred = predict(X_test, W1, b1, W2, b2)

    output_path = os.path.join(OUT_DIR, "all_predictions.txt")

    with open(output_path, "w", encoding="utf-8") as f:
        f.write("===== TRAIN PREDICTIONS =====\n")
        for i, (true, pred) in enumerate(zip(y_train.flatten(), y_train_pred.flatten())):
            f.write(f"Точка {i:5d}: Истинное = {true:.6f} | Предсказание = {pred:.6f}\n")

        f.write("\n===== TEST PREDICTIONS =====\n")
        for i, (true, pred) in enumerate(zip(y_test.flatten(), y_test_pred.flatten()),
```

```
                                        start=len(y_train)):
            f.write(f"Точка {i:5d}: Истинное = {true:.6f} | Предсказание = {pred:.6f}\n")

    print(f"\nВсе предсказания сохранены в файл: {output_path}\n")

    def print_head_tail(name, true_vals, pred_vals, offset=0):
        total = len(true_vals)
        show_n = 10

        print(f"===== {name} (первые {show_n}) =====")
        for i in range(min(show_n, total)):
            print(f"Точка {i+offset:5d}: Истинное = {true_vals[i]:.6f} | Предсказание =
{pred_vals[i]:.6f}")

        print(f"===== {name} (последние {show_n}) =====")
        for i in range(max(0, total - show_n), total):
            print(f"Точка {i+offset:5d}: Истинное = {true_vals[i]:.6f} | Предсказание =
{pred_vals[i]:.6f}")


    print_head_tail("TRAIN", y_train.flatten(), y_train_pred.flatten(), offset=0)
    print_head_tail("TEST", y_test.flatten(), y_test_pred.flatten(), offset=len(y_train))


    plt.figure(figsize=(8, 4.5))
    epochs_range = np.arange(1, len(loss_history) + 1)
    plt.plot(epochs_range, loss_history, label="MSE (train)", linewidth=2)
    plt.fill_between(epochs_range, loss_history, np.max(loss_history), alpha=0.06)
    final_loss = loss_history[-1]
    plt.title("График изменения ошибки (MSE) по эпохам")
    plt.xlabel("Эпоха")
    plt.ylabel("MSE")
    plt.grid(alpha=0.35)
    plt.legend()
    plt.annotate(f"Final MSE = {final_loss:.4e}", xy=(0.98, 0.95), xycoords="axes fraction",
                 ha="right", va="top", fontsize=9, bbox=dict(boxstyle="round,pad=0.3",
fc="white", ec="0.8"))
    plt.tight_layout()
    plt.savefig(os.path.join(OUT_DIR, "loss_curve.png"))
    plt.show()

    plt.figure(figsize=(8, 4.5))
    plt.plot(x_vals, y_vals, lw=2.2, label="Эталонная функция")
    plt.title("Эталонная функция")
    plt.xlabel("x")
    plt.ylabel("y(x)")
    plt.grid(alpha=0.25)
    plt.legend()
    plt.tight_layout()
    plt.savefig(os.path.join(OUT_DIR, "true_function_dense.png"))
    plt.show()

    plt.figure(figsize=(8, 4.5))
    idx = np.arange(len(y_vals))
```

```
    plt.plot(idx, y_vals, lw=2, label="Эталонная функция", zorder=1)

    train_start = window_size
    train_end = window_size + len(y_train_pred)
    test_start = train_end
    test_end = train_end + len(y_test_pred)

    plt.axvspan(train_start, train_end - 1, alpha=0.06, label="Train region")
    plt.axvspan(test_start, test_end - 1, alpha=0.04, label="Test region", color="C1")

    plt.plot(range(train_start, train_end),
             y_train_pred.flatten(), "--", marker="o", markersize=4, label="Прогноз (train)",
zorder=3)
    plt.plot(range(test_start, test_end),
             y_test_pred.flatten(), "--", marker="s", markersize=4, label="Прогноз (test)",
zorder=3)



    plt.title("Прогнозируемая функция — участки обучения и теста")
    plt.xlabel("Индекс точки")
    plt.ylabel("Значение y")
    plt.grid(alpha=0.25)
    plt.legend(loc="upper right")
    plt.tight_layout()
    plt.savefig(os.path.join(OUT_DIR, "prediction_vs_true.png"))
    plt.show()

if __name__ == "__main__":
    main()
```

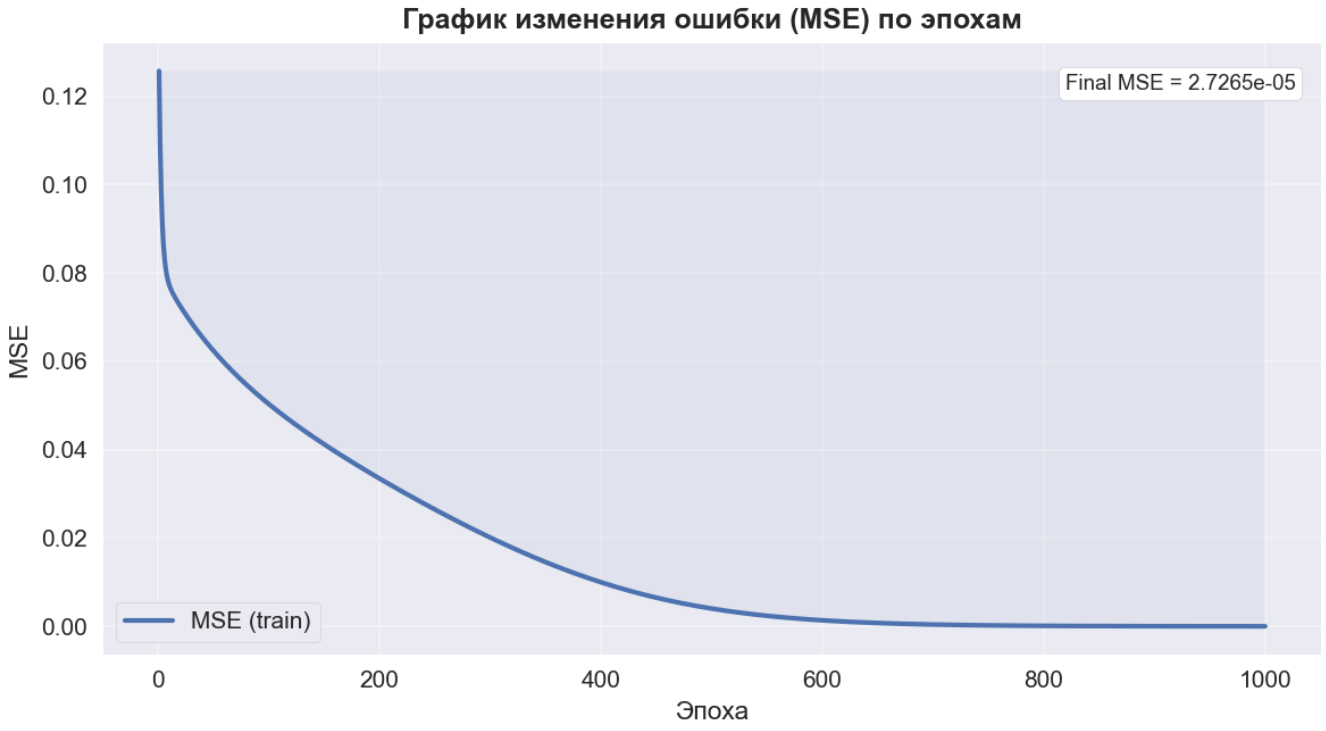
Результаты:

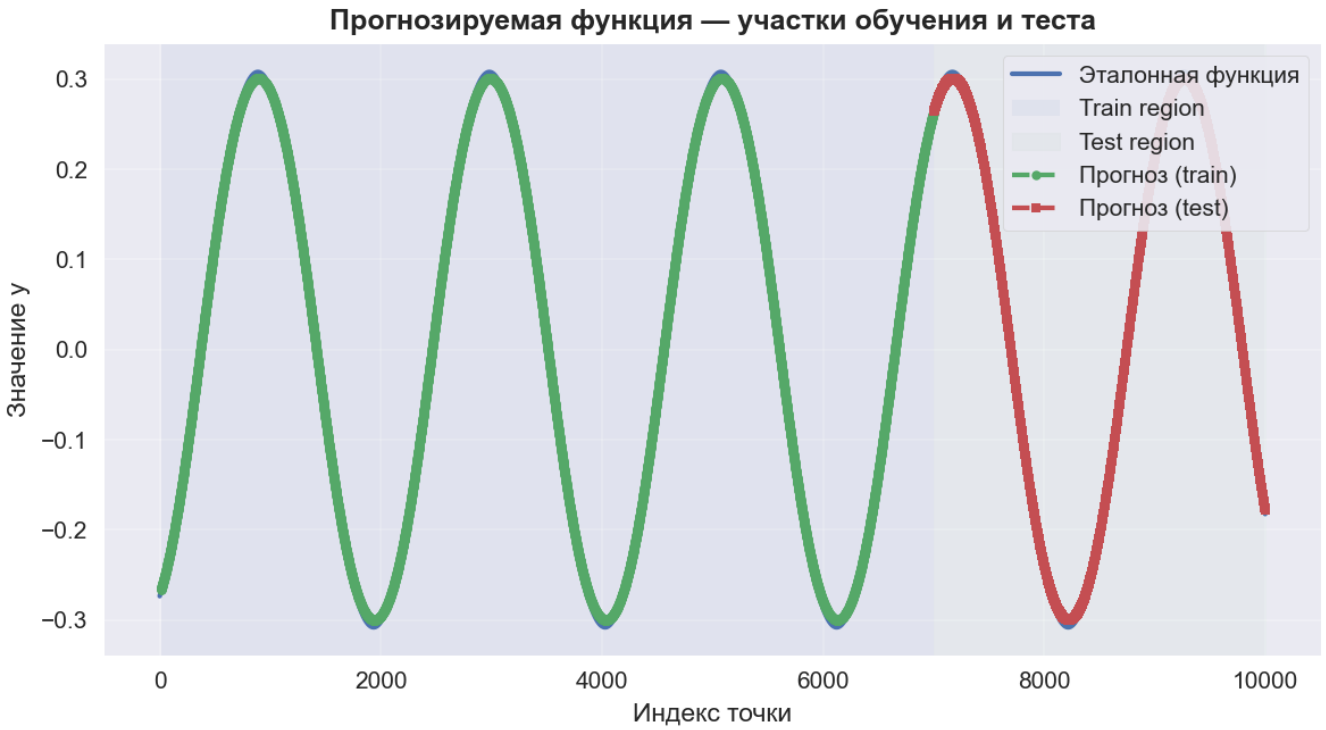
```
Epoch    1 | MSE=0.12569170
Epoch  250 | MSE=0.02648347
Epoch  500 | MSE=0.00404351
Epoch  750 | MSE=0.00023834
Epoch 1000 | MSE=0.00002726

Все предсказания сохранены в файл: lab5_results\all_predicti
ons.txt

===== TRAIN (первые 10) =====
Точка     0: Истинное = -0.269047 | Предсказание = -0.266726
Точка     1: Истинное = -0.268596 | Предсказание = -0.266323
Точка     2: Истинное = -0.268142 | Предсказание = -0.265917
Точка     3: Истинное = -0.267686 | Предсказание = -0.265509
Точка     4: Истинное = -0.267227 | Предсказание = -0.265099
Точка     5: Истинное = -0.266766 | Предсказание = -0.264687
Точка     6: Истинное = -0.266303 | Предсказание = -0.264272
Точка     7: Истинное = -0.265837 | Предсказание = -0.263854
Точка     8: Истинное = -0.265369 | Предсказание = -0.263434
Точка     9: Истинное = -0.264898 | Предсказание = -0.263012
```

```
===== TRAIN (последние 10) =====
Точка  6983: Истинное = 0.265261 | Предсказание = 0.259587
Точка  6984: Истинное = 0.265730 | Предсказание = 0.260035
Точка  6985: Истинное = 0.266196 | Предсказание = 0.260480
Точка  6986: Истинное = 0.266660 | Предсказание = 0.260923
Точка  6987: Истинное = 0.267122 | Предсказание = 0.261364
Точка  6988: Истинное = 0.267581 | Предсказание = 0.261802
Точка  6989: Истинное = 0.268038 | Предсказание = 0.262238
Точка  6990: Истинное = 0.268492 | Предсказание = 0.262672
Точка  6991: Истинное = 0.268944 | Предсказание = 0.263103
Точка  6992: Истинное = 0.269393 | Предсказание = 0.263532
===== TEST (первые 10) =====
Точка  6993: Истинное = 0.269840 | Предсказание = 0.263958
Точка  6994: Истинное = 0.270285 | Предсказание = 0.264383
Точка  6995: Истинное = 0.270727 | Предсказание = 0.264804
Точка  6996: Истинное = 0.271167 | Предсказание = 0.265224
Точка  6997: Истинное = 0.271604 | Предсказание = 0.265641
Точка  6998: Истинное = 0.272039 | Предсказание = 0.266055
Точка  6999: Истинное = 0.272472 | Предсказание = 0.266468
Точка  7000: Истинное = 0.272902 | Предсказание = 0.266878
Точка  7001: Истинное = 0.273329 | Предсказание = 0.267285
Точка  7002: Истинное = 0.273754 | Предсказание = 0.267690
===== TEST (последние 10) =====
Точка  9981: Истинное = -0.175617 | Предсказание = -0.171484
Точка  9982: Истинное = -0.176376 | Предсказание = -0.172234
Точка  9983: Истинное = -0.177133 | Предсказание = -0.172981
Точка  9984: Истинное = -0.177888 | Предсказание = -0.173727
Точка  9985: Истинное = -0.178642 | Предсказание = -0.174471
Точка  9986: Истинное = -0.179394 | Предсказание = -0.175213
Точка  9987: Истинное = -0.180144 | Предсказание = -0.175954
Точка  9988: Истинное = -0.180893 | Предсказание = -0.176692
Точка  9989: Истинное = -0.181641 | Предсказание = -0.177429
Точка  9990: Истинное = -0.182386 | Предсказание = -0.178164
```





Вывод: На практике выполнили моделирование прогнозирующей нелинейной ИНС.