Министерство образования Республики Беларусь
Учреждение образования

«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6
По дисциплине: «ОМО»
Тема: «Рекуррентные нейронные сети»

Выполнил:
Студент 3-го курса
Группы АС-65
Гуща И.В.
Вариант 3
Проверил:
Крощенко А.А.

Брест 2025

Цель работы: Выполнить моделирование прогнозирующей нелинейной ИНС.

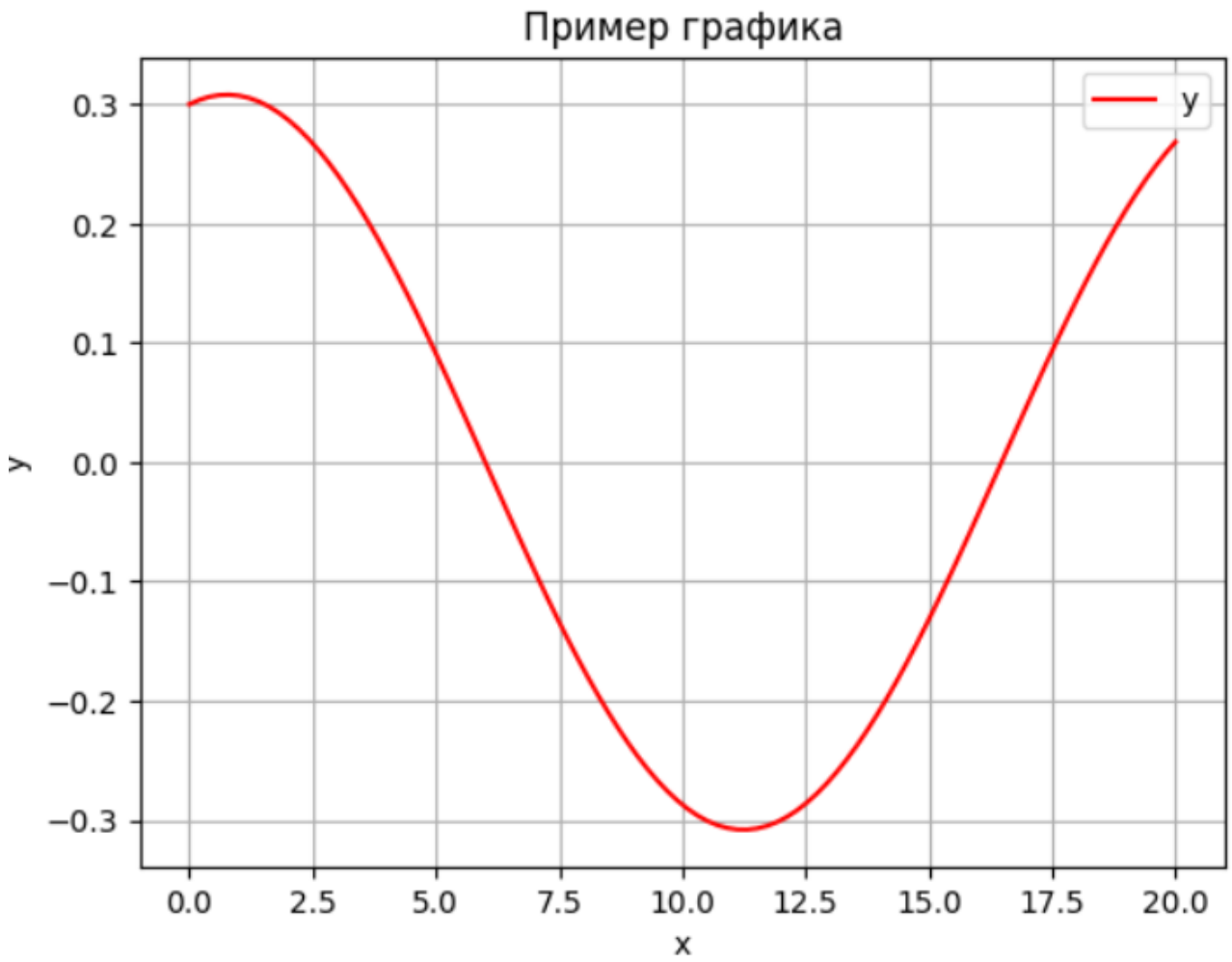
Ход работы

Вариант 3

1. По вариантам предыдущей лабораторной работы реализовать предложенный вариант рекуррентной нейронной сети. Сравнить полученные результаты с ЛР 5.

3	0.3	0.3	0.07	0.3	10	4	Мультирекуррентная

```
import numpy as np

import matplotlib.pyplot as plt

# Генерация данных
def y(x):
    return 0.3*np.cos(0.3*x) + 0.07*np.sin(0.3*x)

x = np.linspace(0, 20, 100)
y_data = y(x)
plt.plot(x, y_data, label='y', color='red')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Пример графика')
plt.legend()
plt.grid(True)
plt.show()
```




```python
# Активация
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Прямой проход RNN
def rnn_forward(x_seq):
    x_seq = x_seq.reshape(1, -1)
    h = np.zeros((1, hidden_size))
    h = sigmoid(x_seq @ Wx + h @ Wh + b_h)
    y_pred = h @ W_out + b_out
    return h, y_pred

# Обучение RNN
def train_rnn(X, Y, lr=0.01):
    global Wx, Wh, b_h, W_out, b_out
    total_loss = 0
    for i in range(len(X)):
        x_seq = X[i]
        y_true = Y[i:i+1]
        h, y_pred = rnn_forward(x_seq)

        delta_out = y_pred - y_true
        total_loss += delta_out**2

        delta_h = delta_out @ W_out.T
        delta_h_raw = delta_h * h * (1 - h)
```

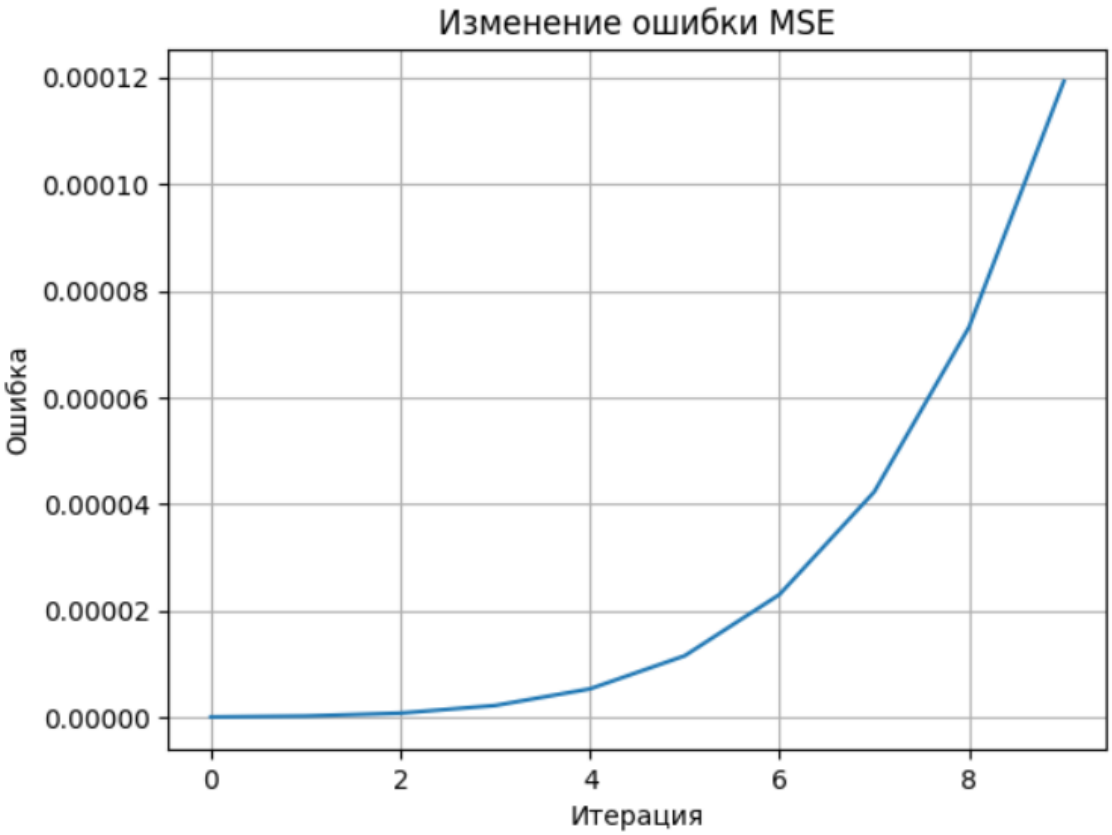
```
            # Обновление весов
            x_seq_2d = x_seq.reshape(1,-1)
            Wx -= lr * x_seq_2d.T @ delta_h_raw
            Wh -= lr * np.zeros_like(h).T @ delta_h_raw  # h_prev=0 для одного
шага
            b_h -= lr * delta_h_raw
            W_out -= lr * h.T @ delta_out
            b_out -= lr * delta_out
        return total_loss / len(X)

# Обучение сети
epochs = 2000
loss_history = []
for epoch in range(epochs):
    loss = train_rnn(X_train, Y_train)
    loss_history.append(loss)
    if epoch % 200 == 0:
        print(f"Epoch {epoch}: loss = {loss.item():.6f}")
```

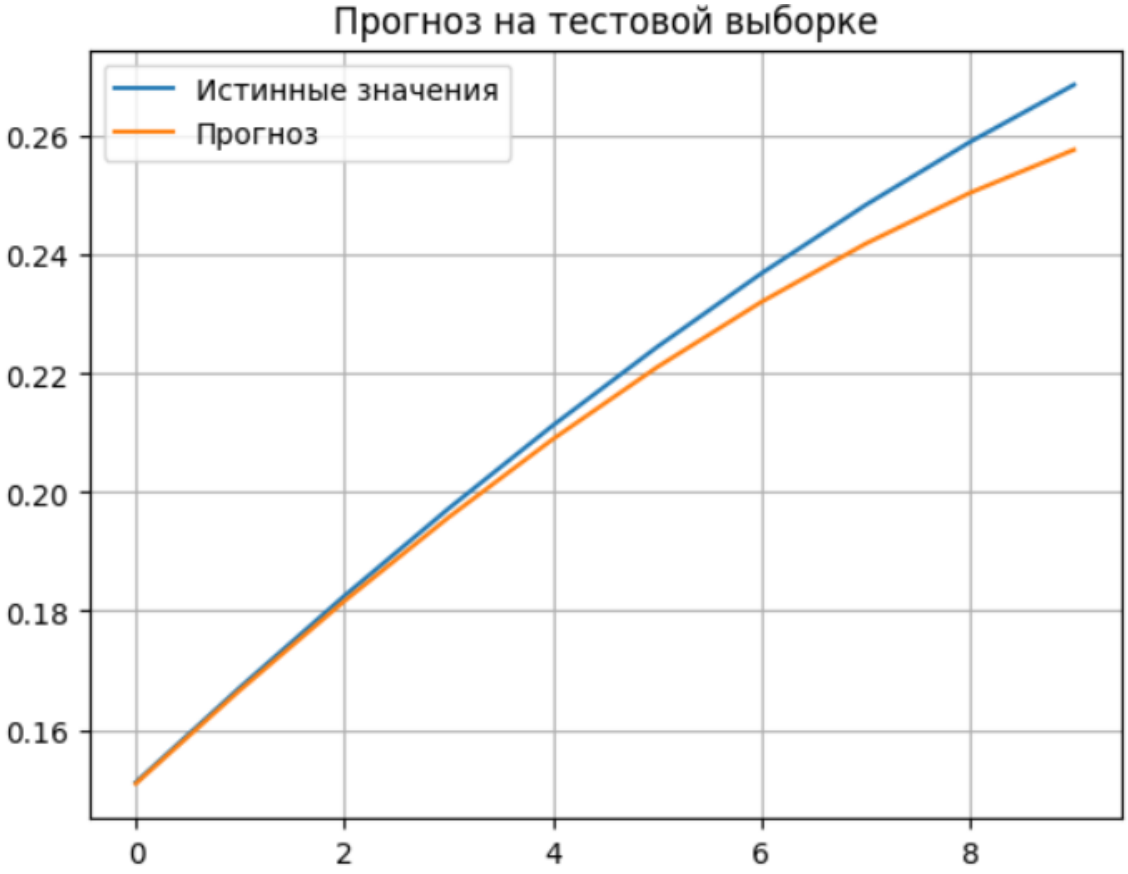
```
Epoch 0: loss = 0.127964
Epoch 200: loss = 0.002394
Epoch 400: loss = 0.001533
Epoch 600: loss = 0.000858
Epoch 800: loss = 0.000411
Epoch 1000: loss = 0.000172
Epoch 1200: loss = 0.000065
Epoch 1400: loss = 0.000023
Epoch 1600: loss = 0.000009
Epoch 1800: loss = 0.000004
```

```
# Прогноз на тестовой выборке
y_test_pred = []
X_seq = X_test[0]
for i in range(len(Y_test)):
    h, y_pred = rnn_forward(X_seq)
    y_test_pred.append(y_pred.item())
    X_seq = np.append(X_seq[1:], y_pred.item())

# Сравнение
print("    Эталон | Предсказано | Отклонение")
for y_true, y_pred in zip(Y_test, y_test_pred):
    print(f"{y_true:.4f}     | {y_pred:.4f}       | {(y_true-y_pred):.4f}")
```

```
      Эталон | Предсказано | Отклонение
  0.1512     | 0.1510      | 0.0002
  0.1672     | 0.1667      | 0.0005
  0.1825     | 0.1816      | 0.0009
  0.1972     | 0.1957      | 0.0015
  0.2112     | 0.2089      | 0.0023
  0.2244     | 0.2210      | 0.0034
  0.2368     | 0.2320      | 0.0048
  0.2483     | 0.2418      | 0.0065
  0.2589     | 0.2503      | 0.0086
  0.2685     | 0.2576      | 0.0109
```

```
# График MSE
mse = (Y_test - np.array(y_test_pred))**2
plt.plot(mse)
plt.title("Изменение ошибки MSE")
plt.xlabel("Итерация")
plt.ylabel("Ошибка")
plt.grid()
plt.show()

# Сравнение прогнозов и истинных значений
plt.plot(Y_test, label="Истинные значения")
plt.plot(y_test_pred, label="Прогноз")
plt.title("Прогноз на тестовой выборке")
plt.legend()
plt.grid()
plt.show()
```





Вывод: В ходе работы была реализована мультирекуррентная нейронная сеть. Она обучалась и тестировалась для той же функции и по итогу показала более точные значения прогноза, чем простая однослойная нейросеть.