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

Отчёт по лабораторной работе №6

Выполнил:
Студент 3 курса
Группы АС-65
Романюк Д. А.
Проверил:
Крощенко А. А.

Брест 2025

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

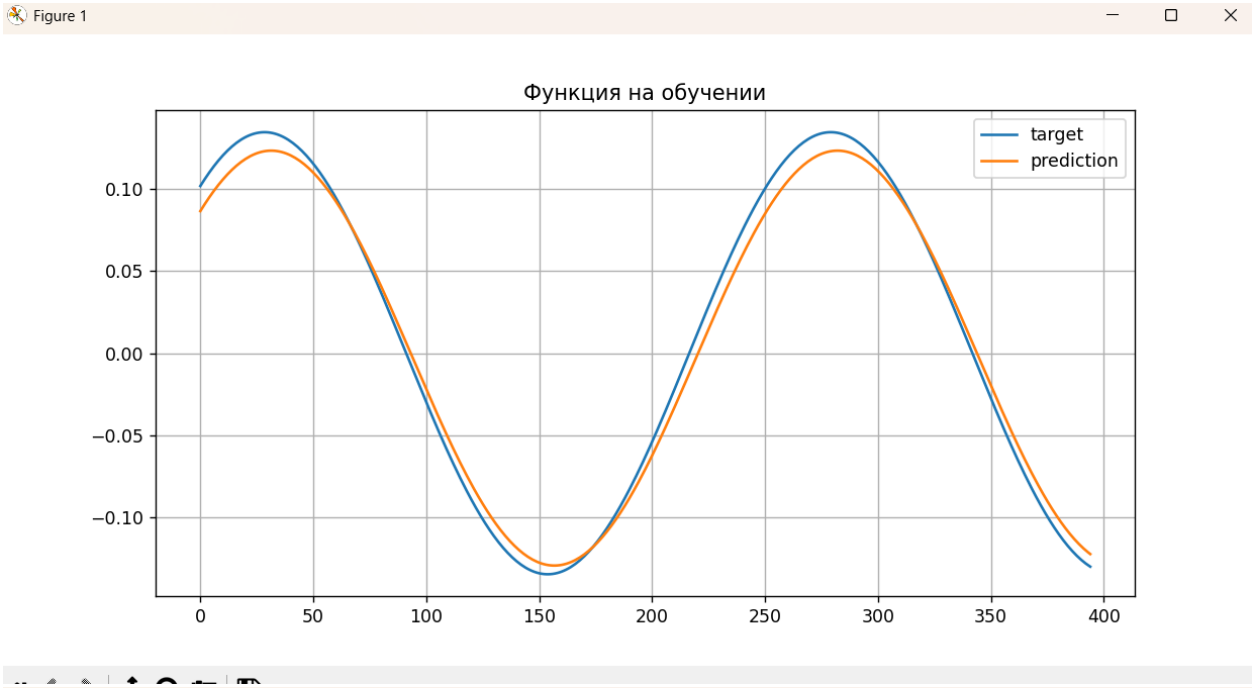
№	a	b	c	d	Кол-во входов ИНС	Кол-во НЭ в скрытом слое	Тип РНС
5	0.1	0.5	0.09	0.5	8	3	Джордана

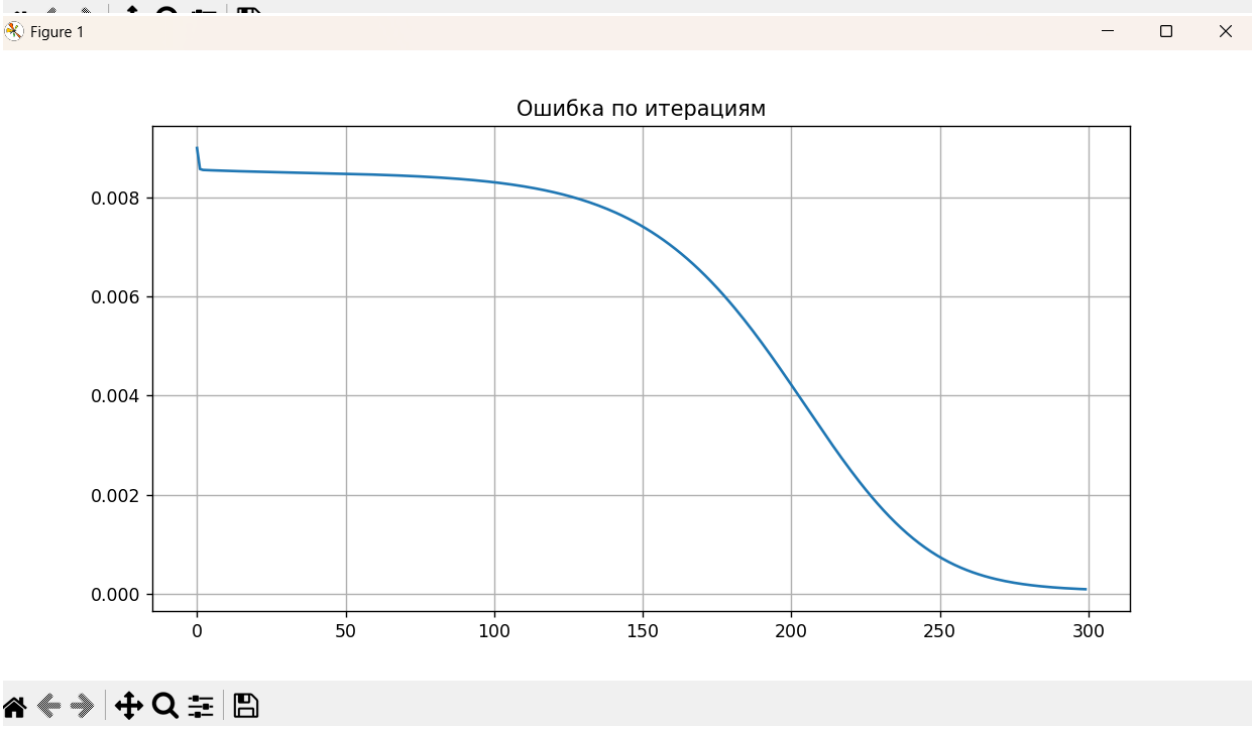
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

a, b, c, d = 0.1, 0.5, 0.09, 0.5

t = np.linspace(0, 20, 400)
y = a*np.sin(b*t) + c*np.cos(d*t)

seq_len = 5

X, Y = [], []
for i in range(len(y) - seq_len):
    X.append(y[i:i+seq_len])
    Y.append(y[i+seq_len])

X = np.array(X)
Y = np.array(Y)

class JordanRNN:
    def __init__(self, input_size, hidden_size):
        self.hidden_size = hidden_size

        self.Wxh = np.random.randn(hidden_size, input_size) * 0.1
        self.Whh = np.random.randn(hidden_size, 1) * 0.1
        self.Why = np.random.randn(1, hidden_size) * 0.1

        self.h = np.zeros((hidden_size, 1))
        self.y_prev = np.zeros((1, 1))

        self.lr = 0.01

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def dsigmoid(self, y):
        return y * (1 - y)

    def forward(self, x):
        x = x.reshape(-1, 1)
        self.h = self.sigmoid(self.Wxh @ x + self.Whh @ self.y_prev)
        return self.Why @ self.h

    def train_step(self, x, target):
        y_pred = self.forward(x)
        error = y_pred - target
```

```
        dWhy = error @ self.h.T
        dh = (self.Why.T @ error) * self.dsigmoid(self.h)
        dWxh = dh @ x.reshape(1, -1)
        dWhh = dh @ self.y_prev.T

        self.Why -= self.lr * dWhy
        self.Wxh -= self.lr * dWxh
        self.Whh -= self.lr * dWhh

        self.y_prev = y_pred.copy()

        return (error**2).item()

rnn = JordanRNN(input_size=seq_len, hidden_size=3)
errors = []

for epoch in range(300):
    err = 0
    rnn.y_prev = np.zeros((1,1))
    for i in range(len(X)):
        err += rnn.train_step(X[i], Y[i])
    errors.append(err / len(X))

pred_train = []
rnn.y_prev = np.zeros((1,1))

for i in range(len(X)):
    pred_train.append(rnn.forward(X[i]).item())

pred_train = np.array(pred_train)

df_train = pd.DataFrame({
    "Эталонное значение": Y[:10],
    "Полученное значение": pred_train[:10],
    "Отклонение": pred_train[:10] - Y[:10]
})

print("\n=== РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (ПЕРВЫЕ 10) ===\n")
print(df_train.to_string(index=False))

future_steps = 30
input_seq = list(Y[-seq_len:])
pred_future = []

rnn.y_prev = np.zeros((1,1))

for _ in range(future_steps):
    x_arr = np.array(input_seq[-seq_len:])
    y_hat = rnn.forward(x_arr).item()
    pred_future.append(y_hat)
    input_seq.append(y_hat)

t_future = np.linspace(t[-1], t[-1] + (20/400)*future_steps, future_steps)
y_future = a*np.sin(b*t_future) + c*np.cos(d*t_future)

df_forecast = pd.DataFrame({
    "Эталонное значение": y_future[:10],
    "Полученное значение": np.array(pred_future[:10]),
```

```
    "Отклонение": np.array(pred_future[:10]) - y_future[:10]
})

print("\n=== РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (ПЕРВЫЕ 10) ===\n")
print(df_forecast.to_string(index=False))

plt.figure(figsize=(10,5))
plt.plot(Y, label="target")
plt.plot(pred_train, label="prediction")
plt.legend()
plt.title("Функция на обучении")
plt.grid()
plt.show()

plt.figure(figsize=(10,5))
plt.plot(errors)
plt.title("Ошибка по итерациям")
plt.grid()
plt.show()
```

```
=== РЕЗУЛЬТАТЫ ОБУЧЕНИЯ (ПЕРВЫЕ 10) ===

 Эталонное значение  Полученное значение  Отклонение
           0.101793            0.086501   -0.015292
           0.103965            0.088705   -0.015260
           0.106073            0.090852   -0.015221
           0.108113            0.092938   -0.015175
           0.110086            0.094964   -0.015121
           0.111989            0.096928   -0.015061
           0.113822            0.098829   -0.014993
           0.115584            0.100666   -0.014918
           0.117273            0.102437   -0.014836
           0.118889            0.104142   -0.014747

=== РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ (ПЕРВЫЕ 10) ===

 Эталонное значение  Полученное значение  Отклонение
          -0.129919           -0.123236    0.006683
          -0.130779           -0.122482    0.008297
          -0.131552           -0.121734    0.009817
          -0.132236           -0.120629    0.011607
          -0.132833           -0.119228    0.013604
          -0.133340           -0.117122    0.016218
          -0.133758           -0.115950    0.017808
          -0.134087           -0.114759    0.019328
          -0.134326           -0.113477    0.020849
          -0.134476           -0.112159    0.022316
```





Вывод: по результатам сравнения можно сделать вывод, что рекуррентная нейронная сеть типа Джордана обеспечивает более высокое качество прогнозирования и более устойчивое поведение модели, чем многослойная сеть прямого распространения из ЛР5. Это связано с тем, что РНС обладает механизмом памяти, позволяющим учитывать временные зависимости, что существенно улучшает точность прогнозирования временных рядов.