Нейроинформатика. Лабораторная работа 8

Динамические сети

Целью работы является исследование свойств некоторых динамических нейронных сетей, алгоритмов обучения, а также применение сетей в задачах аппроксимации функций и распознавания динамических образов.

Выполнил Лисин Роман, М8О-406Б-20.

```
!pip install neurolab
!pip install pyrenn
!pip install fireTS

import neurolab as nl
import numpy as np
import numpy.matlib
from sklearn.metrics import mean_squared_error
import pyrenn
from matplotlib import pyplot as plt
import math
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from sklearn.neural_network import MLPRegressor
from fireTS.models import NARX
```

```
Collecting neurolab
  Downloading neurolab-0.3.5.tar.gz (645 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/645.3 kB ? eta -:--:--
━━━━━━━ ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 122.9/645.3 kB 3.5 MB/s eta
0:00:01 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 645.3/645.3 kB 9.3
MB/s eta 0:00:00
etadata (setup.py) ... e=neurolab-0.3.5-py3-none-any.whl size=22181
sha256=8e45c24e231812f1476c7d5c89ff229764784950c05c8e4e6f782b02908b480
8
  Stored in directory:
/root/.cache/pip/wheels/1d/c0/44/7142fa43c89473c5e63a750a00224e5f9ec9c
a80613de1f97d
Successfully built neurolab
Installing collected packages: neurolab
Successfully installed neurolab-0.3.5
Collecting pyrenn
  Downloading pyrenn-0.1.tar.gz (10 kB)
  Preparing metadata (setup.py) ... ent already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from pyrenn) (1.23.5)
Building wheels for collected packages: pyrenn
```

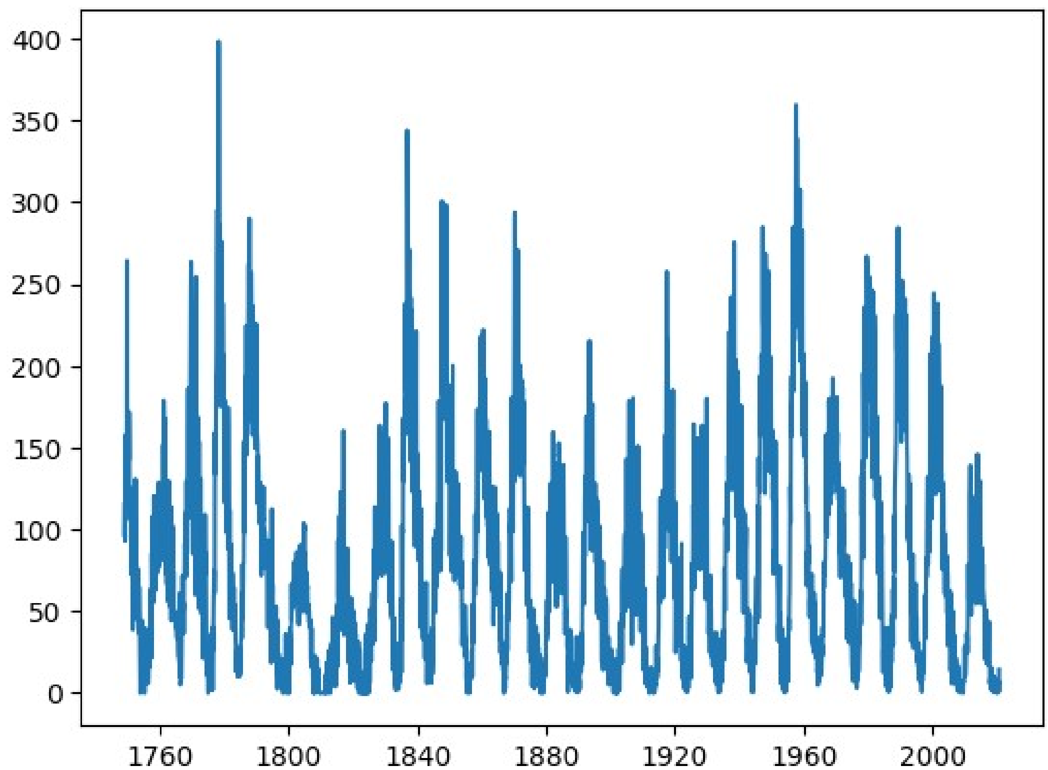
```
  Building wheel for pyrenn (setup.py) ... e=pyrenn-0.1-py3-none-
any.whl size=9239
sha256=c53d0c21cf50c59b7d921c26c473326ecd44457d7e7d20315be5df947ad6c80
2
  Stored in directory:
/root/.cache/pip/wheels/88/73/cf/52f87ad9ea9e987087f5c2b03c8d33e837693
325a2e0305736
Successfully built pyrenn
Installing collected packages: pyrenn
Successfully installed pyrenn-0.1
Collecting fireTS
  Downloading fireTS-0.0.9-py3-none-any.whl (10 kB)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from fireTS) (1.23.5)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages (from fireTS) (1.11.3)
Collecting scikit-learn==1.2.1 (from fireTS)
  Downloading scikit_learn-1.2.1-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.6/9.6 MB 27.0 MB/s eta
0:00:00
ent already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.1-
>fireTS) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn==1.2.1-
>fireTS) (3.2.0)
Installing collected packages: scikit-learn, fireTS
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
bigframes 0.13.0 requires scikit-learn>=1.2.2, but you have scikit-
learn 1.2.1 which is incompatible.
Successfully installed fireTS-0.0.9 scikit-learn-1.2.1
```

##Сеть прямого распространения с запаздыванием для предсказания временного ряда

Число Вольфа - один характерных из показателей солнечной активности. Для заданного момента времени задает количество пятен на Солнце. Для аппроксимации используем среднемесячные значения чисел Вольфа.

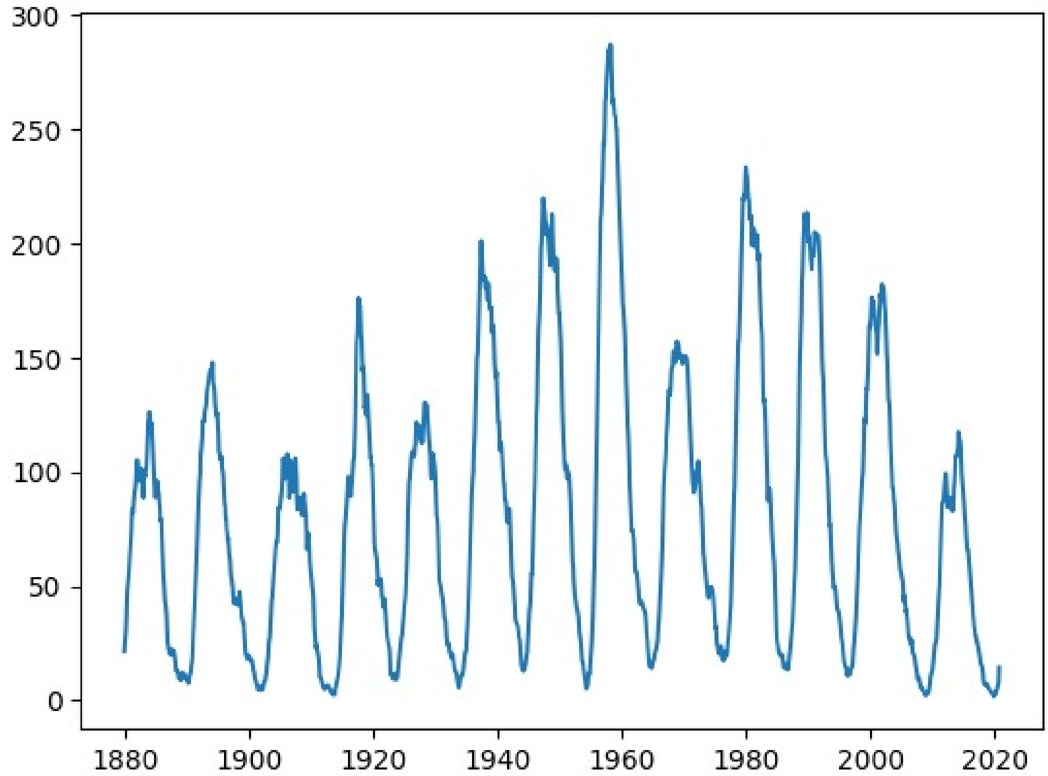
```
date =  "1879-11-01"
df = pd.read_csv('wolfie.csv', sep=';', header=None)
df = df.iloc[:, 0:4]
df[0] = df[0].astype(str)
```

```
df[1] = df[1].astype(str)
df.index = pd.to_datetime(df[0] + '-'+ df[1])
df.drop([0], axis=1, inplace=True)
df.drop([1], axis=1, inplace=True)
df.drop([2], axis=1, inplace=True)
df.head()

plt.plot(df)
plt.show()
```



Выполняем сглаживание и формируем тестовое множество.

```
vals = df.values.flatten()
conv = np.convolve(vals, np.ones(12, dtype=int), 'valid') / 12
r = np.arange(1, 11,2)
start = np.cumsum(vals[:(11)-1])[::2] / r
stop = (np.cumsum(vals[:-(11):-1])[::2] / r)[::-1]
smth_values = np.concatenate((start, conv, stop))

shift = df.values.size - smth_values.size
df.iloc[shift:] = smth_values[:, np.newaxis]
df = df[df.index >= pd.to_datetime(date)]
plt.plot(df)
plt.show()
```




```
deep = 5
split = int(len(df) * 0.7)
train = df[:split]
test = df[split:]

trainData = train.values.squeeze()
xTrain = np.array([trainData[i:i + deep] for i in range(len(trainData)
- deep)])
yTrain = train.iloc[deep:].values

testData = test.values.squeeze()
xTest = np.array([testData[i:i + deep] for i in range(len(testData) -
deep)])
yTest = test.iloc[deep:].values
```

Опишем сеть.

```
model = Sequential()
model.add(Dense(8,activation='relu'))
model.add(Dense(8,))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(xTrain, yTrain, epochs=200)
```

```
Epoch 1/200
37/37 [==============================] - 1s 3ms/step - loss:
10367.1729
Epoch 2/200
37/37 [==============================] - 0s 3ms/step - loss: 1987.0604
Epoch 3/200
37/37 [==============================] - 0s 3ms/step - loss: 202.8123
Epoch 4/200
37/37 [==============================] - 0s 3ms/step - loss: 161.5479
Epoch 5/200
37/37 [==============================] - 0s 3ms/step - loss: 157.3987
Epoch 6/200
37/37 [==============================] - 0s 3ms/step - loss: 154.5793
Epoch 7/200
37/37 [==============================] - 0s 3ms/step - loss: 151.4743
Epoch 8/200
37/37 [==============================] - 0s 3ms/step - loss: 148.6340
Epoch 9/200
37/37 [==============================] - 0s 3ms/step - loss: 145.2246
Epoch 10/200
37/37 [==============================] - 0s 3ms/step - loss: 142.0034
Epoch 11/200
37/37 [==============================] - 0s 3ms/step - loss: 137.9074
Epoch 12/200
37/37 [==============================] - 0s 3ms/step - loss: 134.2640
Epoch 13/200
37/37 [==============================] - 0s 3ms/step - loss: 130.2216
Epoch 14/200
37/37 [==============================] - 0s 3ms/step - loss: 126.2498
Epoch 15/200
37/37 [==============================] - 0s 3ms/step - loss: 122.1356
Epoch 16/200
37/37 [==============================] - 0s 4ms/step - loss: 117.7007
Epoch 17/200
37/37 [==============================] - 0s 5ms/step - loss: 113.3684
Epoch 18/200
37/37 [==============================] - 0s 8ms/step - loss: 108.9858
Epoch 19/200
37/37 [==============================] - 1s 16ms/step - loss: 104.2596
Epoch 20/200
37/37 [==============================] - 0s 11ms/step - loss: 100.1161
Epoch 21/200
37/37 [==============================] - 0s 7ms/step - loss: 95.1845
Epoch 22/200
37/37 [==============================] - 0s 3ms/step - loss: 91.3709
Epoch 23/200
37/37 [==============================] - 0s 3ms/step - loss: 86.1514
Epoch 24/200
37/37 [==============================] - 0s 3ms/step - loss: 82.2318
Epoch 25/200
```

```
37/37 [==============================] - 0s 3ms/step - loss: 77.4748
Epoch 26/200
37/37 [==============================] - 0s 3ms/step - loss: 72.6957
Epoch 27/200
37/37 [==============================] - 0s 3ms/step - loss: 68.1466
Epoch 28/200
37/37 [==============================] - 0s 3ms/step - loss: 64.0883
Epoch 29/200
37/37 [==============================] - 0s 3ms/step - loss: 60.2144
Epoch 30/200
37/37 [==============================] - 0s 3ms/step - loss: 56.1499
Epoch 31/200
37/37 [==============================] - 0s 3ms/step - loss: 51.6322
Epoch 32/200
37/37 [==============================] - 0s 3ms/step - loss: 48.4294
Epoch 33/200
37/37 [==============================] - 0s 3ms/step - loss: 44.0637
Epoch 34/200
37/37 [==============================] - 0s 3ms/step - loss: 40.6353
Epoch 35/200
37/37 [==============================] - 0s 2ms/step - loss: 37.5786
Epoch 36/200
37/37 [==============================] - 0s 3ms/step - loss: 34.7256
Epoch 37/200
37/37 [==============================] - 0s 2ms/step - loss: 31.9612
Epoch 38/200
37/37 [==============================] - 0s 3ms/step - loss: 29.4227
Epoch 39/200
37/37 [==============================] - 0s 3ms/step - loss: 27.2214
Epoch 40/200
37/37 [==============================] - 0s 3ms/step - loss: 25.0941
Epoch 41/200
37/37 [==============================] - 0s 3ms/step - loss: 23.4809
Epoch 42/200
37/37 [==============================] - 0s 3ms/step - loss: 21.7402
Epoch 43/200
37/37 [==============================] - 0s 3ms/step - loss: 20.3988
Epoch 44/200
37/37 [==============================] - 0s 3ms/step - loss: 19.0731
Epoch 45/200
37/37 [==============================] - 0s 3ms/step - loss: 18.4149
Epoch 46/200
37/37 [==============================] - 0s 2ms/step - loss: 17.2780
Epoch 47/200
37/37 [==============================] - 0s 3ms/step - loss: 16.5232
Epoch 48/200
37/37 [==============================] - 0s 3ms/step - loss: 15.9020
Epoch 49/200
37/37 [==============================] - 0s 3ms/step - loss: 15.2121
Epoch 50/200
```

```
37/37 [==============================] - 0s 3ms/step - loss: 14.7747
Epoch 51/200
37/37 [==============================] - 0s 3ms/step - loss: 14.5218
Epoch 52/200
37/37 [==============================] - 0s 3ms/step - loss: 14.8158
Epoch 53/200
37/37 [==============================] - 0s 3ms/step - loss: 14.2488
Epoch 54/200
37/37 [==============================] - 0s 3ms/step - loss: 14.9528
Epoch 55/200
37/37 [==============================] - 0s 3ms/step - loss: 14.3940
Epoch 56/200
37/37 [==============================] - 0s 3ms/step - loss: 13.3274
Epoch 57/200
37/37 [==============================] - 0s 2ms/step - loss: 13.4009
Epoch 58/200
37/37 [==============================] - 0s 2ms/step - loss: 13.4024
Epoch 59/200
37/37 [==============================] - 0s 2ms/step - loss: 13.8014
Epoch 60/200
37/37 [==============================] - 0s 2ms/step - loss: 13.2801
Epoch 61/200
37/37 [==============================] - 0s 2ms/step - loss: 13.4077
Epoch 62/200
37/37 [==============================] - 0s 2ms/step - loss: 13.4161
Epoch 63/200
37/37 [==============================] - 0s 2ms/step - loss: 13.0300
Epoch 64/200
37/37 [==============================] - 0s 2ms/step - loss: 13.0938
Epoch 65/200
37/37 [==============================] - 0s 2ms/step - loss: 13.0503
Epoch 66/200
37/37 [==============================] - 0s 2ms/step - loss: 13.3653
Epoch 67/200
37/37 [==============================] - 0s 2ms/step - loss: 13.5694
Epoch 68/200
37/37 [==============================] - 0s 2ms/step - loss: 13.3832
Epoch 69/200
37/37 [==============================] - 0s 2ms/step - loss: 13.1270
Epoch 70/200
37/37 [==============================] - 0s 2ms/step - loss: 12.6016
Epoch 71/200
37/37 [==============================] - 0s 2ms/step - loss: 12.7151
Epoch 72/200
37/37 [==============================] - 0s 2ms/step - loss: 12.5775
Epoch 73/200
37/37 [==============================] - 0s 2ms/step - loss: 12.4331
Epoch 74/200
37/37 [==============================] - 0s 2ms/step - loss: 12.6666
Epoch 75/200
```

```
37/37 [==============================] - 0s 2ms/step - loss: 12.4601
Epoch 76/200
37/37 [==============================] - 0s 2ms/step - loss: 12.5814
Epoch 77/200
37/37 [==============================] - 0s 2ms/step - loss: 12.5020
Epoch 78/200
37/37 [==============================] - 0s 2ms/step - loss: 12.4348
Epoch 79/200
37/37 [==============================] - 0s 2ms/step - loss: 12.5402
Epoch 80/200
37/37 [==============================] - 0s 2ms/step - loss: 12.2842
Epoch 81/200
37/37 [==============================] - 0s 2ms/step - loss: 12.3040
Epoch 82/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1851
Epoch 83/200
37/37 [==============================] - 0s 2ms/step - loss: 12.2561
Epoch 84/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1522
Epoch 85/200
37/37 [==============================] - 0s 2ms/step - loss: 12.2647
Epoch 86/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1565
Epoch 87/200
37/37 [==============================] - 0s 2ms/step - loss: 12.2188
Epoch 88/200
37/37 [==============================] - 0s 2ms/step - loss: 12.6177
Epoch 89/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1489
Epoch 90/200
37/37 [==============================] - 0s 2ms/step - loss: 11.9744
Epoch 91/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8390
Epoch 92/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8601
Epoch 93/200
37/37 [==============================] - 0s 2ms/step - loss: 11.7965
Epoch 94/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1585
Epoch 95/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8922
Epoch 96/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8596
Epoch 97/200
37/37 [==============================] - 0s 2ms/step - loss: 12.0522
Epoch 98/200
37/37 [==============================] - 0s 2ms/step - loss: 12.2970
Epoch 99/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8502
Epoch 100/200
```

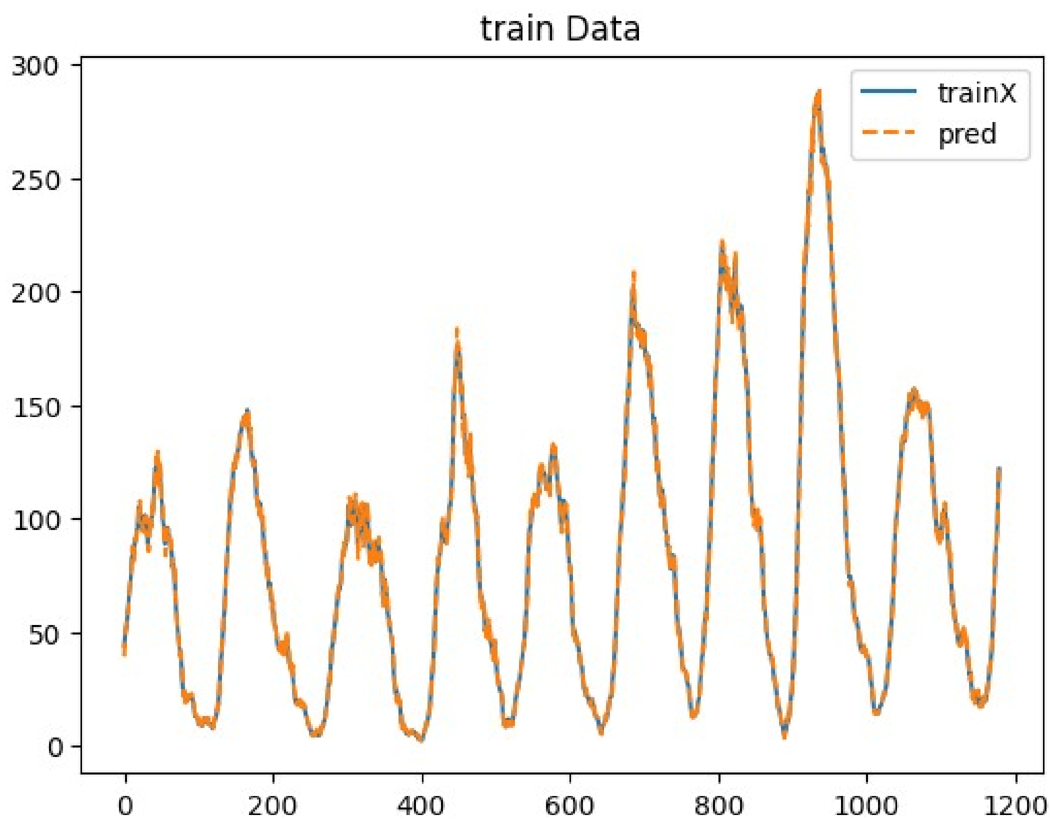
```
37/37 [==============================] - 0s 2ms/step - loss: 11.5687
Epoch 101/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1667
Epoch 102/200
37/37 [==============================] - 0s 2ms/step - loss: 11.5730
Epoch 103/200
37/37 [==============================] - 0s 2ms/step - loss: 11.7037
Epoch 104/200
37/37 [==============================] - 0s 2ms/step - loss: 11.4170
Epoch 105/200
37/37 [==============================] - 0s 2ms/step - loss: 11.4544
Epoch 106/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8650
Epoch 107/200
37/37 [==============================] - 0s 2ms/step - loss: 11.2115
Epoch 108/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1032
Epoch 109/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1619
Epoch 110/200
37/37 [==============================] - 0s 2ms/step - loss: 11.5939
Epoch 111/200
37/37 [==============================] - 0s 2ms/step - loss: 11.9952
Epoch 112/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1806
Epoch 113/200
37/37 [==============================] - 0s 2ms/step - loss: 11.2916
Epoch 114/200
37/37 [==============================] - 0s 2ms/step - loss: 11.4637
Epoch 115/200
37/37 [==============================] - 0s 2ms/step - loss: 11.6667
Epoch 116/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1628
Epoch 117/200
37/37 [==============================] - 0s 2ms/step - loss: 11.2290
Epoch 118/200
37/37 [==============================] - 0s 2ms/step - loss: 10.8941
Epoch 119/200
37/37 [==============================] - 0s 2ms/step - loss: 10.7798
Epoch 120/200
37/37 [==============================] - 0s 2ms/step - loss: 10.8595
Epoch 121/200
37/37 [==============================] - 0s 2ms/step - loss: 10.7630
Epoch 122/200
37/37 [==============================] - 0s 2ms/step - loss: 11.7199
Epoch 123/200
37/37 [==============================] - 0s 2ms/step - loss: 10.9342
Epoch 124/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1219
Epoch 125/200
```

```
37/37 [==============================] - 0s 2ms/step - loss: 11.0037
Epoch 126/200
37/37 [==============================] - 0s 2ms/step - loss: 10.9191
Epoch 127/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1111
Epoch 128/200
37/37 [==============================] - 0s 2ms/step - loss: 12.3145
Epoch 129/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1871
Epoch 130/200
37/37 [==============================] - 0s 2ms/step - loss: 10.6497
Epoch 131/200
37/37 [==============================] - 0s 2ms/step - loss: 10.6684
Epoch 132/200
37/37 [==============================] - 0s 2ms/step - loss: 10.4203
Epoch 133/200
37/37 [==============================] - 0s 2ms/step - loss: 11.8557
Epoch 134/200
37/37 [==============================] - 0s 2ms/step - loss: 10.6801
Epoch 135/200
37/37 [==============================] - 0s 2ms/step - loss: 10.8437
Epoch 136/200
37/37 [==============================] - 0s 2ms/step - loss: 10.6466
Epoch 137/200
37/37 [==============================] - 0s 2ms/step - loss: 10.4940
Epoch 138/200
37/37 [==============================] - 0s 2ms/step - loss: 11.9229
Epoch 139/200
37/37 [==============================] - 0s 2ms/step - loss: 10.4987
Epoch 140/200
37/37 [==============================] - 0s 2ms/step - loss: 11.5222
Epoch 141/200
37/37 [==============================] - 0s 2ms/step - loss: 10.5721
Epoch 142/200
37/37 [==============================] - 0s 2ms/step - loss: 10.3251
Epoch 143/200
37/37 [==============================] - 0s 2ms/step - loss: 10.7726
Epoch 144/200
37/37 [==============================] - 0s 2ms/step - loss: 10.3914
Epoch 145/200
37/37 [==============================] - 0s 2ms/step - loss: 10.3755
Epoch 146/200
37/37 [==============================] - 0s 2ms/step - loss: 10.6851
Epoch 147/200
37/37 [==============================] - 0s 2ms/step - loss: 11.0393
Epoch 148/200
37/37 [==============================] - 0s 2ms/step - loss: 10.3761
Epoch 149/200
37/37 [==============================] - 0s 2ms/step - loss: 10.7788
Epoch 150/200
```

```
37/37 [==============================] - 0s 2ms/step - loss: 10.1346
Epoch 151/200
37/37 [==============================] - 0s 2ms/step - loss: 9.9843
Epoch 152/200
37/37 [==============================] - 0s 2ms/step - loss: 9.9648
Epoch 153/200
37/37 [==============================] - 0s 2ms/step - loss: 10.2385
Epoch 154/200
37/37 [==============================] - 0s 2ms/step - loss: 12.1009
Epoch 155/200
37/37 [==============================] - 0s 2ms/step - loss: 11.0953
Epoch 156/200
37/37 [==============================] - 0s 2ms/step - loss: 10.3763
Epoch 157/200
37/37 [==============================] - 0s 2ms/step - loss: 10.4858
Epoch 158/200
37/37 [==============================] - 0s 2ms/step - loss: 10.0879
Epoch 159/200
37/37 [==============================] - 0s 2ms/step - loss: 10.1591
Epoch 160/200
37/37 [==============================] - 0s 2ms/step - loss: 10.3025
Epoch 161/200
37/37 [==============================] - 0s 2ms/step - loss: 10.2232
Epoch 162/200
37/37 [==============================] - 0s 2ms/step - loss: 9.9605
Epoch 163/200
37/37 [==============================] - 0s 2ms/step - loss: 10.1884
Epoch 164/200
37/37 [==============================] - 0s 2ms/step - loss: 10.0063
Epoch 165/200
37/37 [==============================] - 0s 2ms/step - loss: 9.9748
Epoch 166/200
37/37 [==============================] - 0s 2ms/step - loss: 9.9670
Epoch 167/200
37/37 [==============================] - 0s 2ms/step - loss: 10.7118
Epoch 168/200
37/37 [==============================] - 0s 2ms/step - loss: 10.1969
Epoch 169/200
37/37 [==============================] - 0s 2ms/step - loss: 10.9986
Epoch 170/200
37/37 [==============================] - 0s 2ms/step - loss: 9.6914
Epoch 171/200
37/37 [==============================] - 0s 2ms/step - loss: 10.2458
Epoch 172/200
37/37 [==============================] - 0s 2ms/step - loss: 10.4329
Epoch 173/200
37/37 [==============================] - 0s 2ms/step - loss: 9.7611
Epoch 174/200
37/37 [==============================] - 0s 2ms/step - loss: 10.0624
Epoch 175/200
```

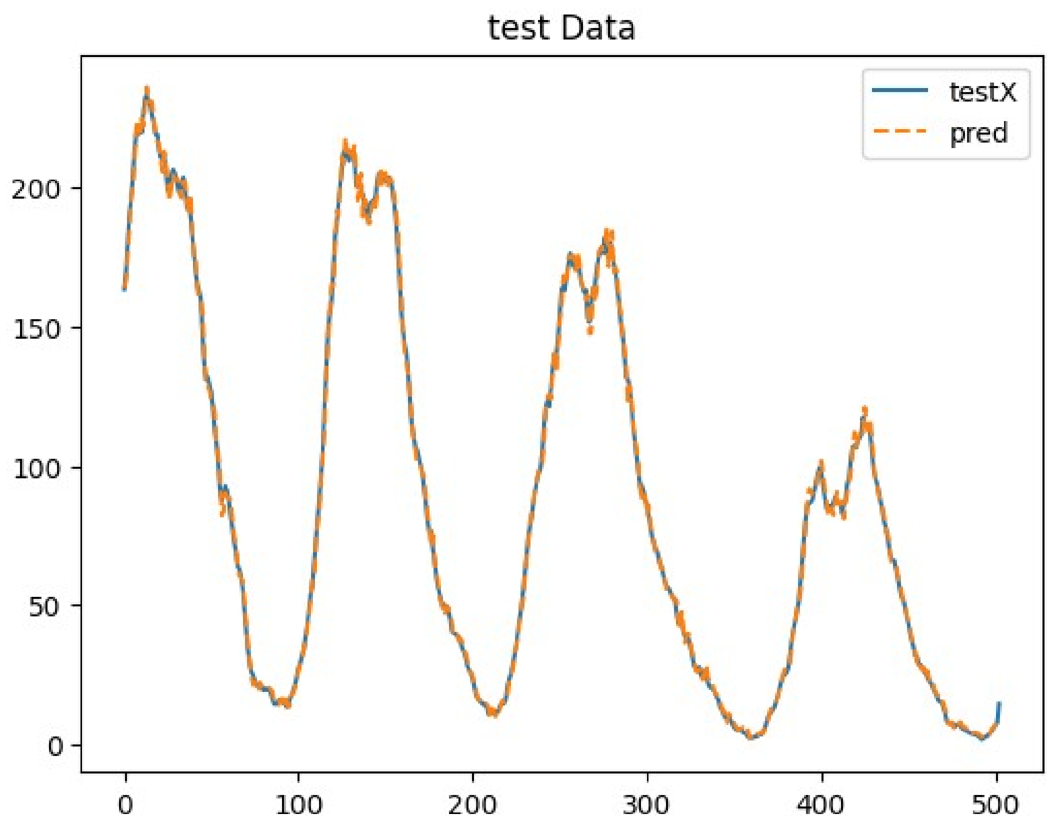
```
37/37 [==============================] - 0s 2ms/step - loss: 9.7900
Epoch 176/200
37/37 [==============================] - 0s 2ms/step - loss: 11.1896
Epoch 177/200
37/37 [==============================] - 0s 2ms/step - loss: 11.6564
Epoch 178/200
37/37 [==============================] - 0s 2ms/step - loss: 10.0222
Epoch 179/200
37/37 [==============================] - 0s 2ms/step - loss: 9.6936
Epoch 180/200
37/37 [==============================] - 0s 2ms/step - loss: 9.5938
Epoch 181/200
37/37 [==============================] - 0s 3ms/step - loss: 10.1343
Epoch 182/200
37/37 [==============================] - 0s 3ms/step - loss: 10.3124
Epoch 183/200
37/37 [==============================] - 0s 3ms/step - loss: 9.7132
Epoch 184/200
37/37 [==============================] - 0s 3ms/step - loss: 12.4168
Epoch 185/200
37/37 [==============================] - 0s 3ms/step - loss: 9.6924
Epoch 186/200
37/37 [==============================] - 0s 3ms/step - loss: 9.6206
Epoch 187/200
37/37 [==============================] - 0s 3ms/step - loss: 10.2141
Epoch 188/200
37/37 [==============================] - 0s 3ms/step - loss: 9.9265
Epoch 189/200
37/37 [==============================] - 0s 3ms/step - loss: 10.9133
Epoch 190/200
37/37 [==============================] - 0s 3ms/step - loss: 9.7267
Epoch 191/200
37/37 [==============================] - 0s 3ms/step - loss: 9.7897
Epoch 192/200
37/37 [==============================] - 0s 3ms/step - loss: 9.7818
Epoch 193/200
37/37 [==============================] - 0s 3ms/step - loss: 9.6845
Epoch 194/200
37/37 [==============================] - 0s 3ms/step - loss: 10.2967
Epoch 195/200
37/37 [==============================] - 0s 3ms/step - loss: 9.5064
Epoch 196/200
37/37 [==============================] - 0s 3ms/step - loss: 9.7351
Epoch 197/200
37/37 [==============================] - 0s 2ms/step - loss: 9.5214
Epoch 198/200
37/37 [==============================] - 0s 2ms/step - loss: 9.7673
Epoch 199/200
37/37 [==============================] - 0s 2ms/step - loss: 9.4348
```

```
Epoch 200/200
37/37 [==============================] - 0s 2ms/step - loss: 10.1606

<keras.src.callbacks.History at 0x7ec74fcbb700>

pred = model.predict(xTrain)
plt.plot(yTrain)
plt.plot(pred, '--')

plt.legend(['trainX', 'pred'])
plt.title("train Data")
plt.show()

37/37 [==============================] - 0s 2ms/step
```




```
pred = model.predict(xTest)
plt.plot(yTest)
plt.plot(pred, '--')

plt.legend(['testX', 'pred'])
plt.title("test Data")
plt.show()
MSE = mean_squared_error(yTest, pred)
print('MSE = {}'.format(MSE))
```

```
16/16 [==============================] - 0s 2ms/step
```




```
MSE = 7.852698324915448
```

Выполним многошаговый прогноз.

```
a = np.array(xTrain[0], ndmin=2)
preds = np.array([])
for i in range(200):
  pred = model.predict(a)
  a = np.append(a[:,1:], pred, axis=1)
  preds = np.append(preds, pred)

plt.plot(yTrain[:200])
plt.plot(preds, '--')

plt.legend(['testX', 'pred'])
plt.show()
MSE = mean_squared_error(yTrain[:len(preds)], preds)
print('MSE = {}'.format(MSE))

1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
```

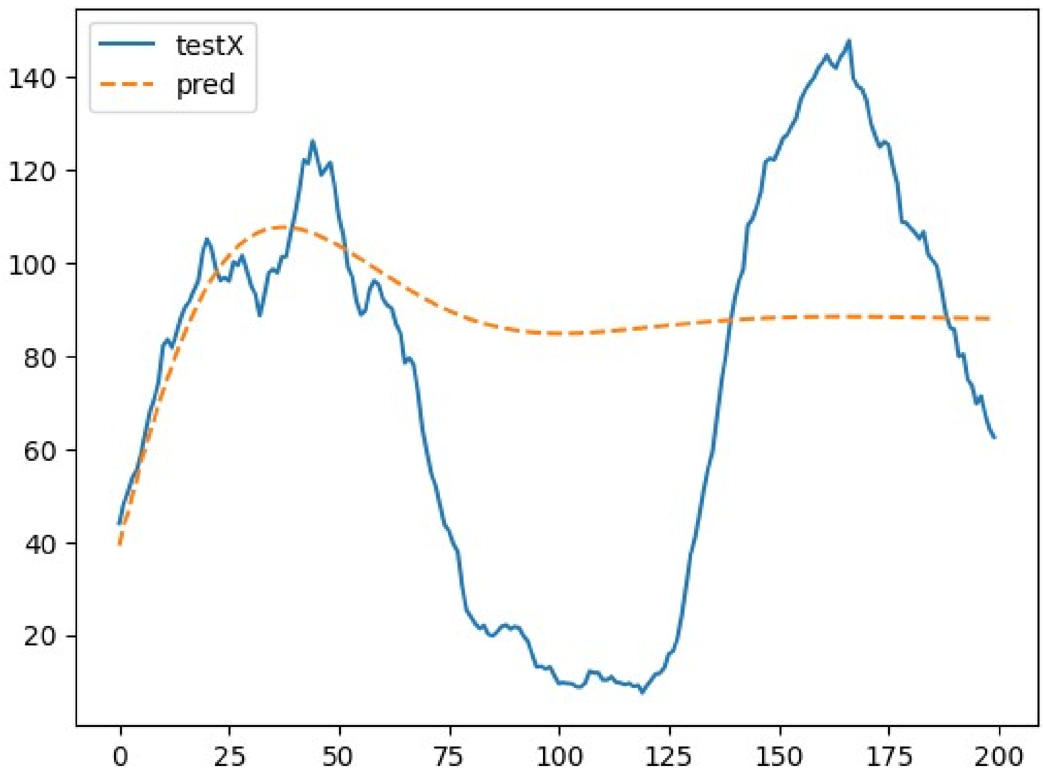
```
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
```

```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 114ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
```

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
```

```
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 29ms/step
```



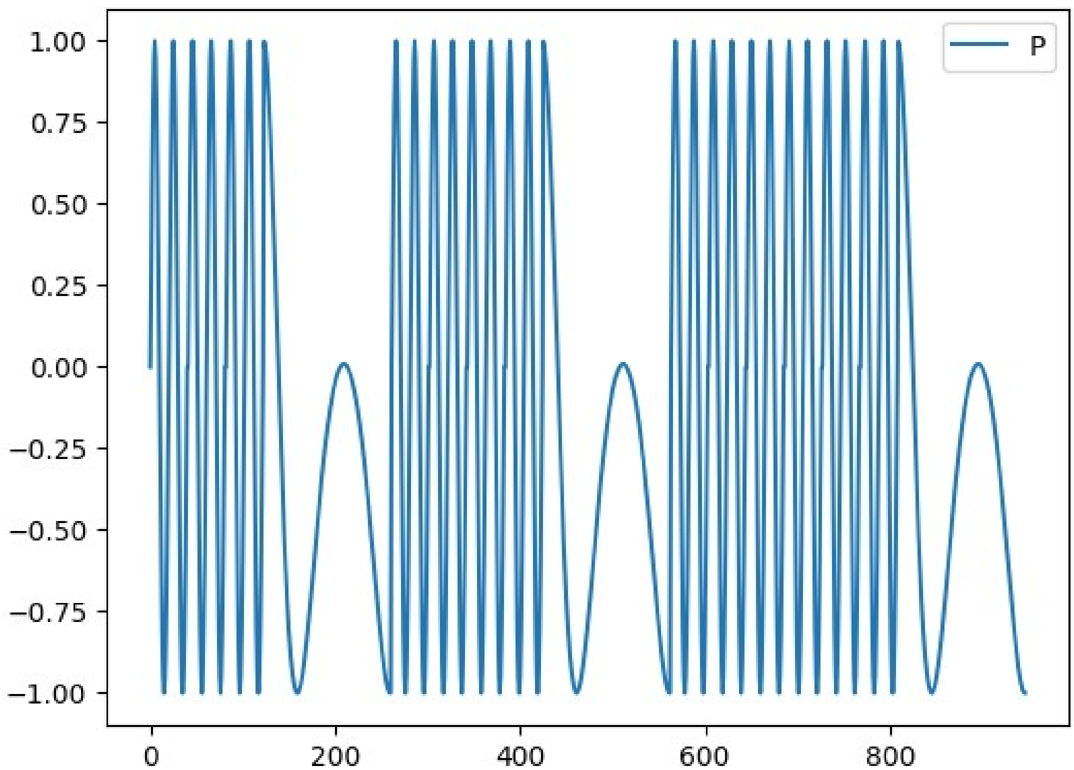

```
MSE = 1824.252036979638
```

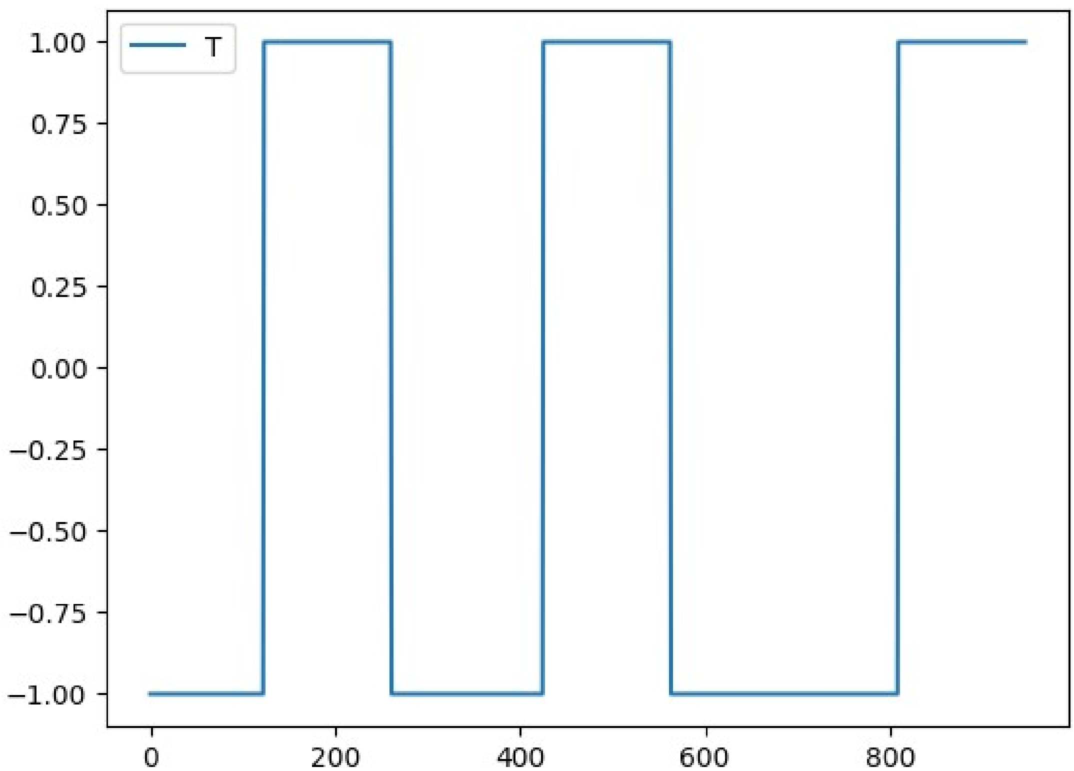
##Сеть прямого распространения с распределенным запаздыванием для распознавания динамического образа

Обучающее множество взято из лабораторной работы №5. Входная последовательность обучающего множества состоит из комбинации основного сигнала p и сигнала, подлежащего распознаванию g. Каждому значению основного сигнала соответствует -1 целевого выхода, каждому значению сигнала g соответствует 1 целевого выхода.

```
# основа
k1 = np.arange(0, 1+0.025, 0.025)
p = np.sin(4*np.pi*k1)
t1 = np.full(len(p),-1)

# для распознавания
k2 = np.arange(2.84, 6.25+0.025, 0.025)
g = np.sin(k2**2-10*k2+3)
t2 = np.full(len(g), 1)

R = [3,4,6]
```

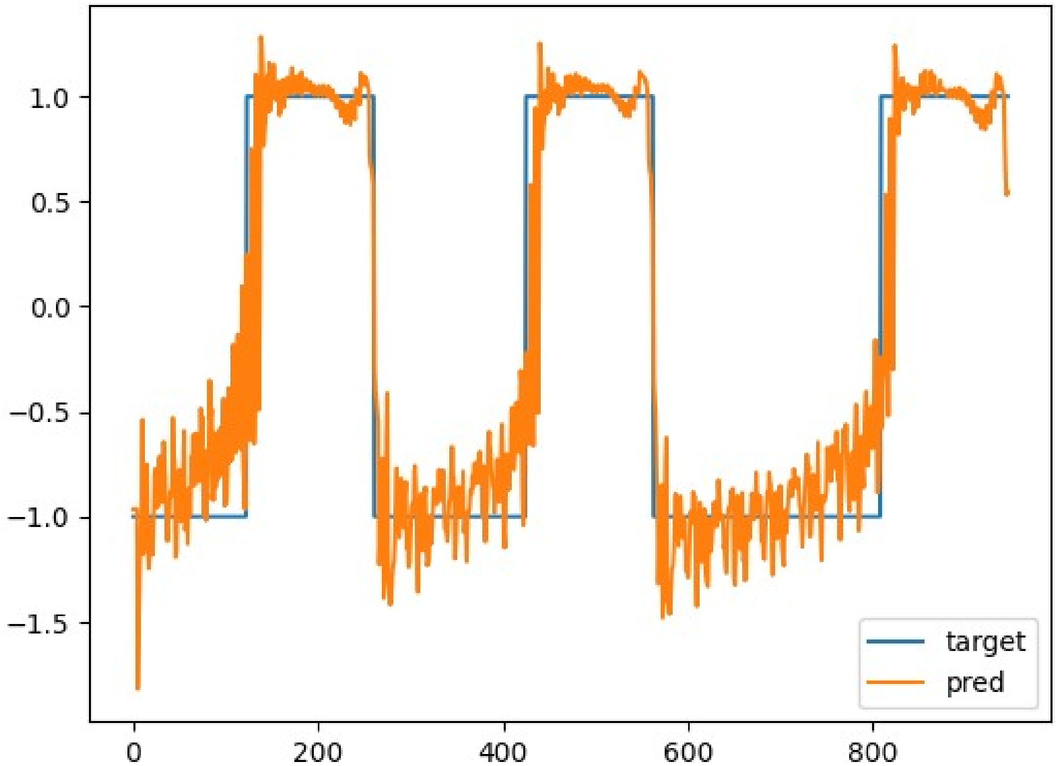
```
P =
np.array(np.append(np.append(np.append(np.append(np.append(np.matlib.r
epmat(p,1,R[0]), g),
np.matlib.repmat(p,1,R[1])),g),np.matlib.repmat(p,1,R[2])),g),ndmin=2)
.reshape(-1,1)
T =
np.array(np.append(np.append(np.append(np.append(np.append(np.matlib.r
epmat(t1,1,R[0]), t2),
np.matlib.repmat(t1,1,R[1])),t2),np.matlib.repmat(t1,1,R[2])),t2),ndmi
n=2).reshape(-1,1)

plt.plot(P.reshape(len(P)))
plt.legend(['P'])
plt.show()
plt.plot(T.reshape(len(T)))
plt.legend(['T'])
plt.show()
```





Опишем сеть.

```
T = T.ravel()
P = P.ravel()
nn = pyrenn.CreateNN([1, 8, 1], dIn=[5], dIntern=[5])
nn = pyrenn.train_LM(P, T, nn, E_stop=1e-5, k_max=100)

Maximum number of iterations reached

pred = pyrenn.NNOut(P, nn)
plt.plot(T)
plt.plot(pred)
plt.legend(['target', 'pred'])
plt.show()
```

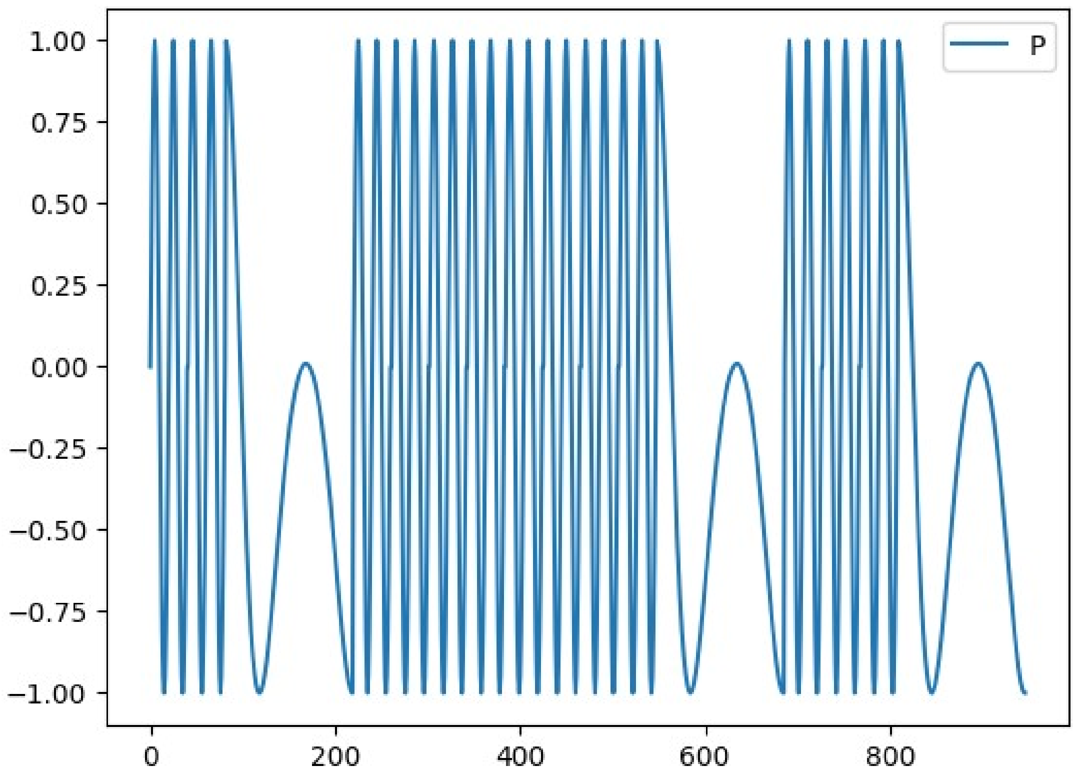


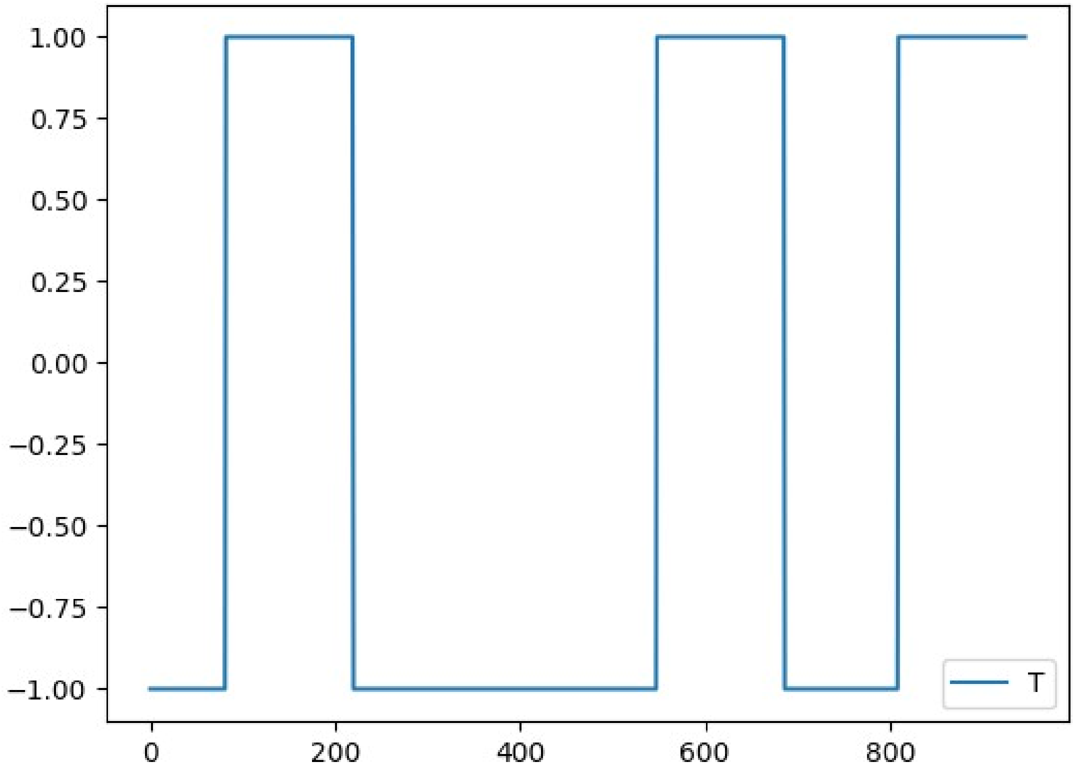

```
out = [1 if(i>=0) else -1 for i in pred]
print('final accuracy = {}'.format((out ==
T.reshape(len(pred))).mean()))

final accuracy = 0.9704329461457233
```

Проверим качество модели, поменяв длительность основного сигнала R.

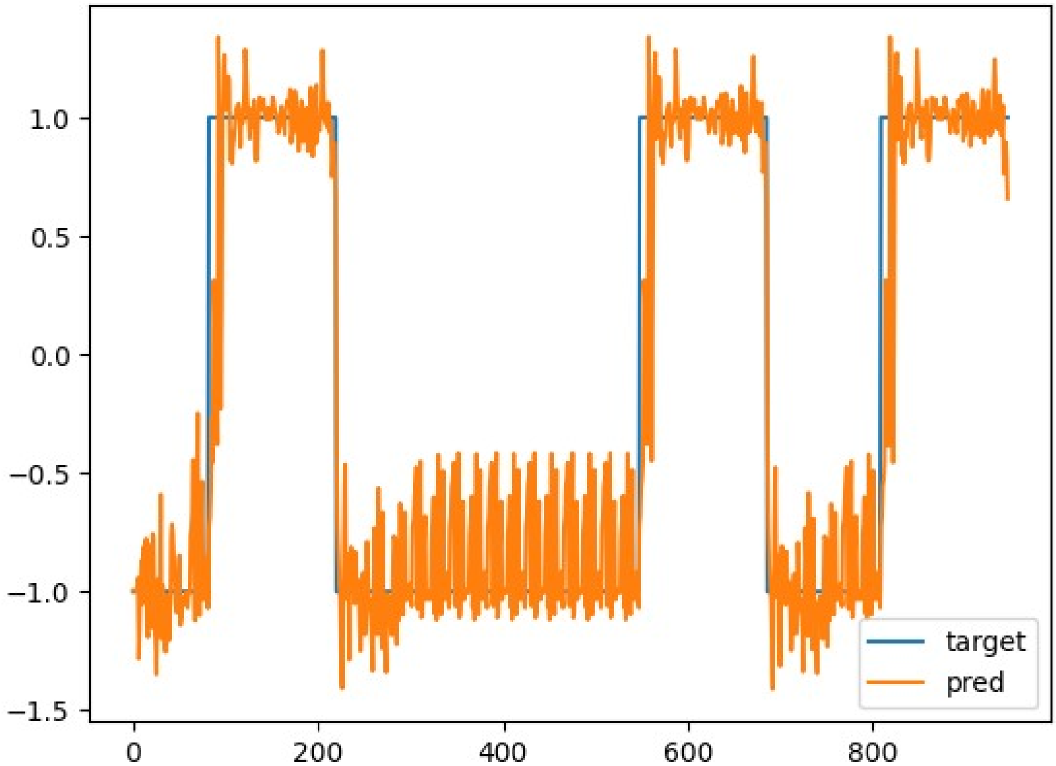
```
R = [2,8,3]

P =
np.array(np.append(np.append(np.append(np.append(np.append(np.matlib.r
epmat(p,1,R[0]), g),
np.matlib.repmat(p,1,R[1])),g),np.matlib.repmat(p,1,R[2])),g),ndmin=2)
.reshape(-1,1)
T =
np.array(np.append(np.append(np.append(np.append(np.append(np.matlib.r
epmat(t1,1,R[0]), t2),
np.matlib.repmat(t1,1,R[1])),t2),np.matlib.repmat(t1,1,R[2])),t2),ndmi
n=2).reshape(-1,1)

plt.plot(P.reshape(len(P)))
plt.legend(['P'])
plt.show()
plt.plot(T.reshape(len(T)))
```

```
plt.legend(['T'])
plt.show()
```






```
T = T.ravel()
P = P.ravel()
nn = pyrenn.CreateNN([1, 8, 1], dIn=[5], dIntern=[5])
nn = pyrenn.train_LM(P, T, nn, E_stop=1e-5, k_max=120)
pred = pyrenn.NNOut(P, nn)
plt.plot(T)
plt.plot(pred)
plt.legend(['target', 'pred'])
plt.show()
out = [1 if(i>=0) else -1 for i in pred]
print('final accuracy = {}'.format((out ==
T.reshape(len(pred))).mean()))

Maximum number of iterations reached
```




```
final accuracy = 0.9746568109820486
```

##Нелинейная авторегрессионная сеть с внешними входами для аппроксимации траектории динамической системы

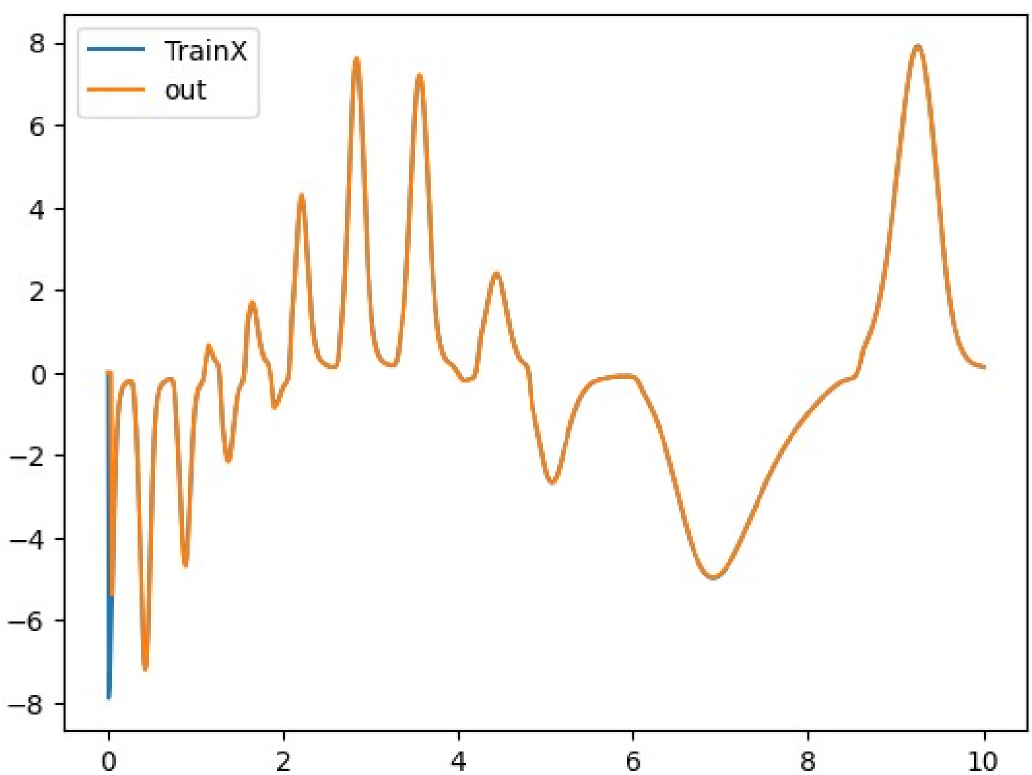
Определим входной и выходной сигналы.

```
def u(k):
    return np.cos(k**2 - 15 * k + 3) - np.cos(k)

def y_next(k):
    y = [0.]
    for i in k:
        y.append(y[-1] / (1 + y[-1]**2) + u(i)**3)
    return y[:-1]

k = np.arange(0, 10.01, 0.01)
y = y_next(k)
input = u(k)[:, np.newaxis]
target = y
delay = 3

xTrain = k[:700]
xTest = k[700:900]
xValid = k[900:997]
```

```
yTrain = y[:700]
yTest = y[700:900]
yValid = y[900:997]
```

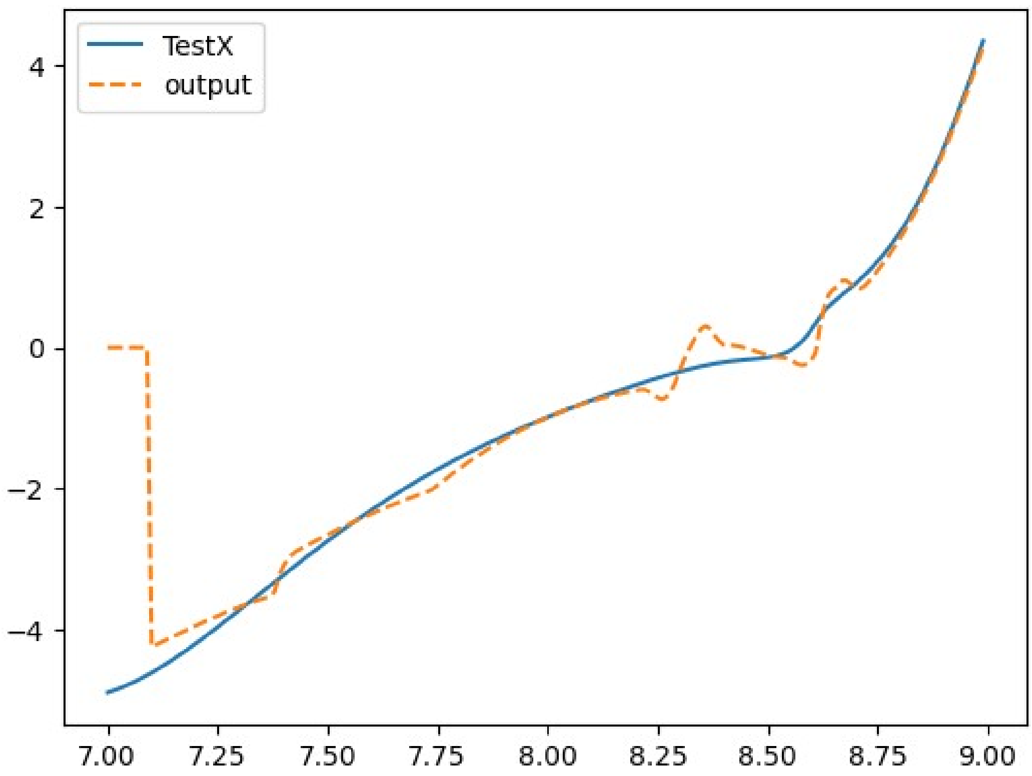
Опишем сеть NARX.

```
narx = NARX(MLPRegressor(hidden_layer_sizes=(10, 10)), solver='lbfgs',
max_iter=600,
                         auto_order=2, exog_order=[2],
exog_delay=[delay])

narx.fit(input, target)

output = narx.predict(input, target, step=1)
output[np.isnan(output)] = 0
MSE = mean_squared_error(target, output)
print('MSE = {}'.format(MSE))

MSE = 0.20938654125762532

plt.plot(k, y)
plt.plot(k, output)
plt.legend(['TrainX', 'out'])
plt.show()
```



Выполним многошаговый прогноз.

```
narx = NARX(MLPRegressor(hidden_layer_sizes=(10, 10)), solver='lbfgs',
max_iter=1000,
                         auto_order=2, exog_order=[delay],
exog_delay=[delay])
narx.fit(input, target)

inputTest = u(xTest)[:, np.newaxis]
targetTest = yTest
out = narx.predict(inputTest, targetTest, step=5)
out[np.isnan(out)] = 0
MSE = mean_squared_error(targetTest, out)
print('MSE = {}'.format(MSE))
plt.plot(xTest, yTest)
plt.plot(xTest, out, '--')
plt.legend(['TestX', 'output'])
plt.show()

MSE = 1.1724809504921836
```



Вывод

В данной лабораторной работе я изучил и обучил динамические нейронные сети для многошагового прогноза и аппроксимации траектории динамической системы. Сеть NARX прекрасно справляется с многошаговым прогнозом.