3 Part 3: Chaotic time series prediction

I construct a reservoir with the settings given in the task description. I train the reservoir and use the training states to estimate the output weights. The output weights are then used to predict the test data time series.

3.1 Code

```
import numpy as np
#import pandas as pd

# Load data
Xtrain = np.genfromtxt('/Users/Jensaeh/skola/Neural'+
                     ' networks/HW3/part3/training-set.csv', delimiter=',')
Xtest = np.genfromtxt('/Users/Jensaeh/skola/Neural'+
                     ' networks/HW3/part3/test-set-8.csv', delimiter=',')

#### Functions for- ####

# -calculating state of reservoir
def calc_rvec(rvec, w, w_in, x_t):
    tmp = np.add( np.matmul(w,rvec) , np.matmul(w_in,x_t) )
    return np.tanh(tmp)
# -training the reservoir
def train_reservoir(Xtrain, N):
    n, T_train = Xtrain.shape
    # Initialize
    w_in = np.random.normal(scale = np.sqrt(0.002), size=(N,n)) # variance = 0.002
    w = np.random.normal(scale = np.sqrt(2/N), size=(N,N)) # variance = 2/N
    R = np.zeros((N,T_train))
    for ti in range(T_train-1):
        R[:,ti+1] = calc_rvec(R[:,ti], w, w_in, Xtrain[:,ti])
    return R, w, w_in
# -estimating output weights
def ridge_regression(Xtrain, R, k):
    RR = np.matmul(R, R.transpose())
    kI = np.identity(R.shape[0]) * k
    para = np.linalg.inv(np.add(RR, kI))
    tmp = np.matmul(R.transpose(), para)
    tmp = np.matmul(Xtrain , tmp)
    return tmp
# -calculating output for a single timestep
def calc_output_vec(w_out_vec, rvec):
    return np.matmul(w_out_vec,rvec)
# -predicting time series with the reservoir
def predict(Xmat, T, W_out, W, W_in):
    n, Ttest = Xmat.shape
    N = W_out.shape[1]
    R = np.zeros((N,Ttest+T))
    O = np.zeros((n,T))
    for t1 in range(Ttest-1):
        R[:,t1+1] = calc_rvec(R[:,t1], W, W_in, Xmat[:,t1])
    O[:,0] = calc_output_vec(W_out, R[:,t1+1])
    r = R[:,t1+1]
    for t2 in range(T-1):
        r = calc_rvec(r, W, W_in, O[:,t2])
        O[:,t2+1] = calc_output_vec(W_out, r)
    return O
```

```
#### Calculations ####

# Training
N = 500 # reservoir neurons
n = 3 # input neurons
R, W, W_in = train_reservoir(Xtrain, N)
# Estimate w_out
k = 0.01 # ridge parameter
w_out = ridge_regression(Xtrain,R,k)
# Predict using Xtest
T = 500
output = predict(Xtest, T, w_out, W, W_in)
# Plot results
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(xs = output[0,:], ys = output[1,:], zs=output[2,:])
np.savetxt('prediction.csv', output[1,:], delimiter=',')
```