Chaotic systems forecasting by echo state networks (ESNs) and Delay RC system.

1 Outline of the task

Build very effective predictor for chaotic (irregular real) systems based on delay reservoir system which can compete with state-of-art forecasting techniques.

This tutorial is based on the paper of *Jaeger (2004)*. The initial Matlab codes is from supplement material to this article, which than were simplified and reformatted to be more readable and amateur-friendly. Most of the code for delay system were written from scratch.

2 Model under investigation

Forecasting capability of reservoir systems will be illustrated on the Mackey-Glass system at first. The Mackey-Glass system forecasting is a standard benchmark for time series prediction studies. It generates a subtly irregular time series which we will call Mackey-Glass time series. The model is well-known and has a form of first order delay-differential equation,

$$\frac{dx}{dt} = \frac{0.2x(t-\tau)}{1+x(t-\tau)^{10}} - 0.1x(t),$$

where τ is delay. Here $\tau = 17$ is used, the value standardly employed in most of the MGS prediction literature. The system can be simulated by any available solver for delay differential equation (dde23 Matlab, dde23 from Python pydelay or directly by simple Euler method). A step-size of 1.0 is used. The resulting time series were shifted by -1 and passed through a tanh function, whereby they fell into a range $(-0.5, 0.3)$. From all data series thus generated, the first 1000 were discarded to get rid of initial washouts.

3 Echo state network design

To construct the ESN, a 1000×1000 reservoir weight matrix W_{int} was generated with 1% connectivity (sparse matrix with density 0.01) and random weights drawn from a uniform distribution over $(-1, 1)$, then rescaled to spectral radius 0.85. Output feedback connection weights W_{ofb} were randomly selected from a uniform distribution over $(-1, 1)$. An auxiliary input unit was attached with similar random connections W_{inp} to feed in a constant bias input of size 0.2. It should be noted that for forecasting task input nodes are optional. The reservoir are fed by backward propagation of output (teacher signal). The outline of the reservoir is presented in Fig.1 Building the ESN in Matlab is easy procedure. One should define number of inputs, number of internal reservoir nodes, number of outputs and generate all connection matrix (weight matrices W_{xxx}) between main three elements: inputs, reservoir and outputs. Here is example of code:

```
netDim = 1000;                % number of neurons (nodes) in reservoir
connectivity = 0.01;          % sparse  density
inputLength = 1;              % number of input nodes
outputLength = 1;             % number of outputs nodes
totalDim = netDim + inputLength + outputLength; % total number of nodes
spectralRadius = 0.85;        % spectral radius for internal weight matrix
outputFeedbackScaling = 1;    % feedback connections scaling


%% Define internal weight matrix: connection between nodes in reservoir

% initial sparse matrix
intWM = sprand(netDim, netDim, connectivity);
% shift non-zero elements on -0.5
intWM = spfun(@(x)(x-0.5),intWM);
```

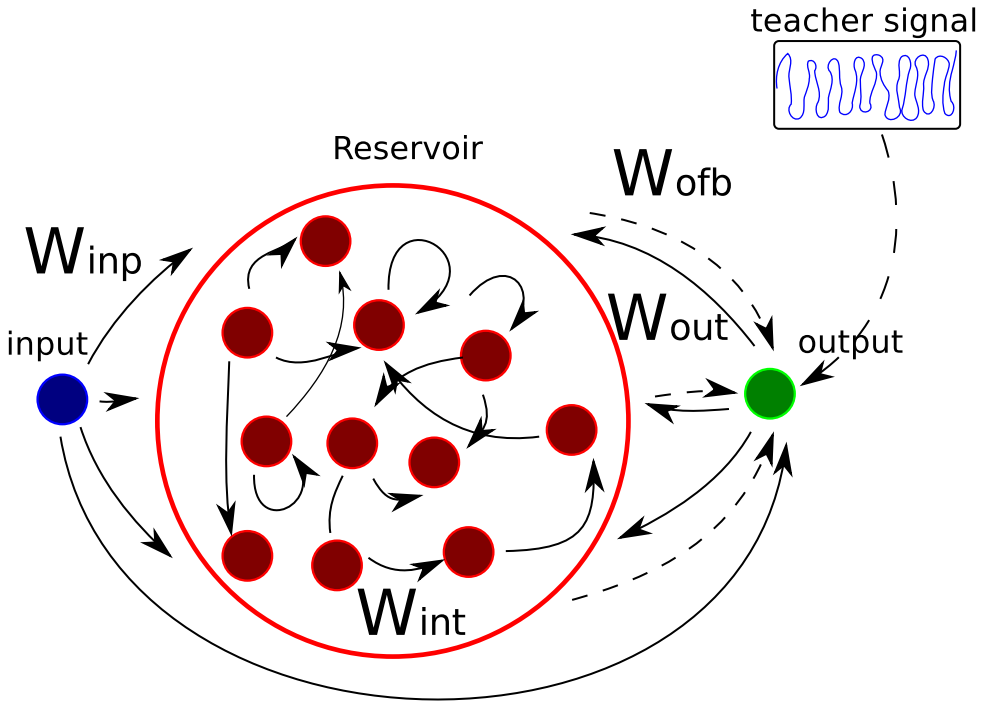

Fig. 1: ESN structure for forecasting task

```
% scale to make spectral radius equal to 1
intWM = intWM/max(abs(eigs(intWM)));
% scale to desired spectral radius
intWM = spectralRadius * intWM;



%% Input weight matrix has weight vectors per input unit in colums
inWM = 2.0 * rand(netDim, inputLength)- 1.0;

%% Output weight matrix has weights for output units in rows
% includes weights for input-to-output connections
initialOutWM = zeros(outputLength, netDim + inputLength);

%% Output feedback weight matrix has weights in columns
ofbWM = outputFeedbackScaling * (2.0 * rand(netDim, outputLength)- 1.0);
```

4 Learning

Learning procedure can be outlined as follows. First a teacher signal should be generated form the MGS equations. Then it should be fed into the reservoir through output neuron. Importantly, the output node was equipped with random connections W_{ofb} that project teacher signal back into the reservoir. A 3000 step teacher sequence $d(1)$,..., $d(3000)$ was generated from the MGS equation and fed into the output node. It excites the internal neurons through the output feedback connections. After an initial transient, they started to exhibit systematic individual variations of the teacher sequence. (Note again, the input neuron is not particularly used for learning and testing, it is connected to the reservoir only to supply constant value bias).

To produce the teacher sequence $d(n)$ the Euler method were used `trainseq=generateMGSeq(samplelength)`, where `samplelength` is number of generated samples. The code for function `generateMGSeq` is straightforward,

```
function outseq=generateMGSeq(samplelength)
% this script generates a sequence of the MG tau=17 attractor from a random
% seed, so every call of this script produces another sequence.


initWashoutLength = 1000;

tau = 17;
```

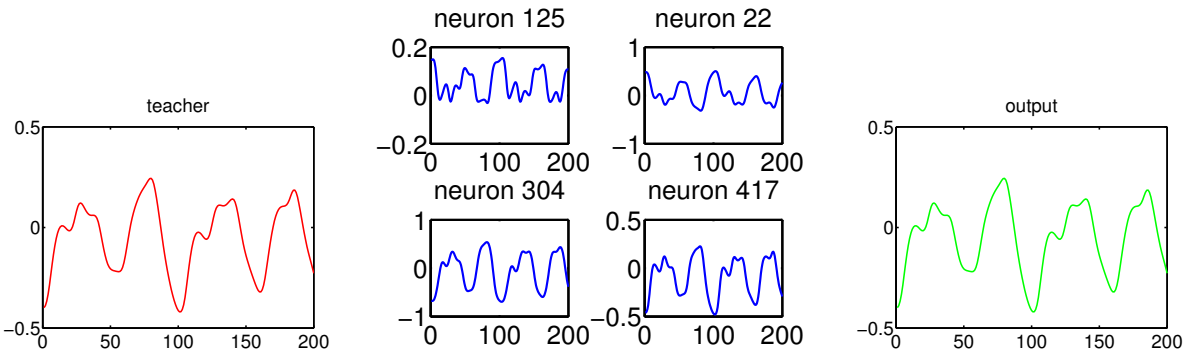

Fig. 2: Example of the free run for 200 steps after training: teacher signal, four randomly selected neurons and output.

```
incrementsperUnit = 10;
genHistoryLength = tau * incrementsperUnit ;
seed = 1.2 * ones(genHistoryLength,1)+ 0.2 * (rand(genHistoryLength,1)-0.5);
oldval = 1.2;
genHistory = seed;
speedup = 1;

sample = zeros(samplelength,1);

step = 0;
for n = 1: samplelength + initWashoutLength
    for i = 1:incrementsperUnit * speedup
        step = step + 1;
        tauval = genHistory(mod(step,genHistoryLength)+1,1);
        newval = oldval + (0.2 * tauval/(1.0 + tauval^10) - 0.1 * oldval)/incrementsperUnit;
        genHistory(mod(step,genHistoryLength)+1,1) = oldval;
        oldval = newval;
    end
    if n > initWashoutLength
        sample(n - initWashoutLength,1) = newval;
    end
end


% transform to learning format
outseq = sample(1:samplelength,1) - 1.0;
outseq = tanh(outseq);
```

Learning process has two main steps. Consider both of them in details.

Start with mathematical background for ESNs. If $\mathbf{x}(n)$ is internal states of reservoir then update network equation has a typical form,

$$\mathbf{x}(n+1) = \tanh(W_{int}\mathbf{x}(n) + W_{inp}u(n+1) + W_{ofb}y(n) + v(n)), \tag{1}$$

where W_{int}, W_{inp}, W_{ofb} are internal, input and feedback connection matrices (see Fig.1). $u(n)$ is input signal (optional for prediction task), $v(n)$ is noise (optional) and $y(n)$ is state of output neuron which is governed by following equation,

$$y(n) = \tanh(W_{out}(\mathbf{x}(n), u(n))), \tag{2}$$

where W_{output} is output (readout) connection matrix. This matrix is most important: its elements is not random and should be trained during learning process. The equations (1) and (2) compose the basic system which defines the evolution of the network.

1) Initial run étape. Training set $d(n)$ were generated from the MGS. Length of training sequence was 3000, first 1000 steps were discarded to wash out initial transient. This means that evolution of the network were governed by the system (1) and (2) where output $y(n)$ were not calculated and reservoir were fed by force-teacher signal $d(n)$ through output neuron. In other hands in this étape $y(n) \equiv d(n)$ in Eq. (1). Noise were not applied, $v(n) \equiv 0$.

2) Sample run étape. The echo signals $\mathbf{x}(n)$ were sampled from remaining 2000 steps. The evolution of the network were governed by the system (1) and (2) in same way as previous, the reservoir was fed by force-teacher signal $d(n)$ through output neuron ($y(n) \equiv d(n)$). Noise $v(n)$ of size $1.0e-10$ was added on network units during sampling, a trick to increase stability of the trained network. During this

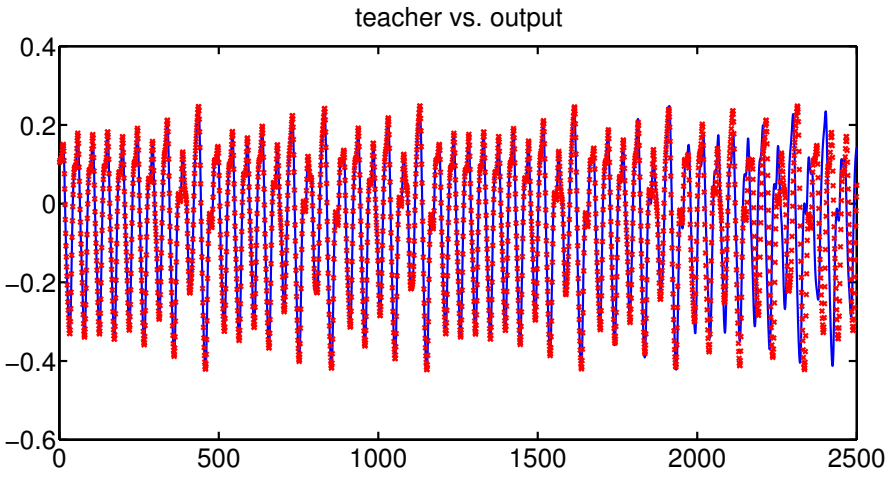

Fig. 3: Example of time series prediction of Mackey-Glass system by trained ESN with 1000 neurons.

étape the reservoir states $\mathbf{x}(n)$ were recorded for further training procedure. The main goal of training is to find readout parameters W_{output} which minimize an error between output $y(n)$ and initial (teacher) signal $d(n)$. The error to be minimized was $MSE_{train} = 1/2000 \sum_{n=1001}^{3000} (\tanh^{-1} d(n) - W_{output}\mathbf{x}(n))^2$. The weight vector W_{output} that minimizes this MSE is made from the regression weights of the linear regression of the $\tanh^{-1} d(n)$ values on the $\mathbf{x}(n)$ values. Any method for computing linear regressions can be used to obtain W_{output}. In this work the pseudoinverse routine `pinv` from Matlab was employed.

After the sample run one can do next several steps of a free run as preliminary test. Fig.4 shows an example of the free run for 2000 steps after training. Moreover the test error of the free run can be calculated straightforward $NMSE = \sum_{i=1}^{200} (d(n+i) - y(n+i))^2 / 200\sigma^2$, where σ^2 is variance of MG signal. The test error for presented example was $NMSE_{test} = 2.67175e-08$

5 Testing

5.1 Prediction test

For testing, the MGS time series were teacher-forced on the trained network for 2000 steps. This run is same as initial run étape in training. Trained network is just go through procedure of adopting states to new 2000 teacher's samples. Then the network was left running freely and the network's continuation was compared to the true continuation after 84 steps to obtain an average normalized root mean square error for the 84 step prediction of $NMSE_{84} = (d(+84) - y(+84))^2/\sigma^2$, where $d(+84)$ is correct continuation, $y(+84)$ network prediction, σ^2 is variance of MG signal. To calculate the statistical error one need to make several test runs. It is possible to make with one teacher-forced series. The procedure is simple: first error is calculated as above (2000 teacher-forces steps followed by 84 test free run steps), second trial is continuation of first trial (2084 teacher-forced steps followed by 84 new free run test steps) and so on. Total error can be calculated as root mean square of all trials $NRMSE_{84} = \sqrt{\sum_{i=1}^{100} (d(+84) - y(+84))^2 / 100\sigma^2}$, where supposed that number of trials is 100. The resulting $NRMSE_{84}$ of 100 test trials for above presented simulation with outlined in text parameters is very good: $NRMSE_{84} = 2.0738e-05$.

5.2 Free run test

Another interesting test is to make free run (after training) for a long time to see the deviation of original MGS signal and output of trained network. For example, Fig.3 shows an overlay of a 2500 step free running continuation of the trained network with the continuation of the original MGS series. The deviation is only visible after around 1500 free run steps. This is excellent result.

6 Delay system as predictor

6.1 System

The system is an electro-optical delay system with one delay feedback loops. The system can be modeled by a delay-differential equation,

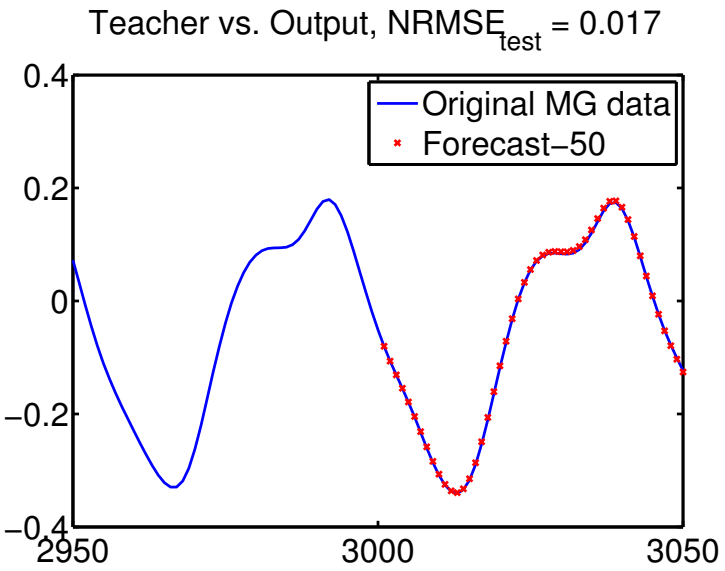

Fig. 4: Free run prediction after training stage

$$\tau\frac{\mathrm{d}x}{\mathrm{d}t} + x(t) = \beta \sin^2\left[x(t-\tau_D) + \gamma_{in} Inp(t) + \gamma_{out} Out(t) + \phi_0\right], \tag{3}$$

where the total delay time is denoted by τ_D. $Inp(t)$ and $Out(t)$ are input and output-feedback signals respectively. Virtual nodes ($N_{node} = 150$) are equally spaced ($\delta\tau_D$) and fill total delay time τ_D. Continuous input and output-feedback signals are commonly generated from discrete input (constant bias $u = 0.2$) and teacher $d(n)$ by operation,

$$\begin{cases} Inp(t)_n = ext(W_{inp} \times u, \tau_D), \\ Out(t)_n = ext(W_{ofb} \times d(n), \tau_D), \end{cases}$$

where $ext(x, \tau_D)$ function which takes a vector x and extend it continuously to τ_D interval as step-wise function of t.

Everywhere in this report the parameter were fixed as,

$$\begin{aligned} &\beta = 0.5, \\ &\gamma_{in} = \gamma_{out} = 1.0, \\ &\phi_0 = \pi/4, \\ &\tau = 1, \\ &\delta\tau_D = 0.2\tau. \end{aligned}$$

6.2 MG test

Free run prediction test for 50 points forward is shown in Fig.4. Best model over 500 randomly generated were selected by minimizing prediction test to step 42, $NRMSE42_{test}$. It is clear that delay reservoir system can predict MG dynamics quite well up to 50 steps forward. Literally speaking the resulting $NRMSE_{test} = 0.017$. To compare this result with state-of-the-art predictions we need more information on statistic of MGS dynamics forecasting with the same parameters. The test for more then 50 steps prediction were performed too. The result is not stable: predicted sequence goes to infinity from time to time depending on warm-up MG sequence which is generated from random seed.

To demonstrate how the delay reservoir behaves during learning (responds to an external signal from MGS) the reservoir states for the last 300 steps of training phase are shown in Fig. Reservoir states are presented both as matrix for all states and as traces for particular nodes.

6.3 Solar activity test

Sun spots data are commonly known as Wolf number records.

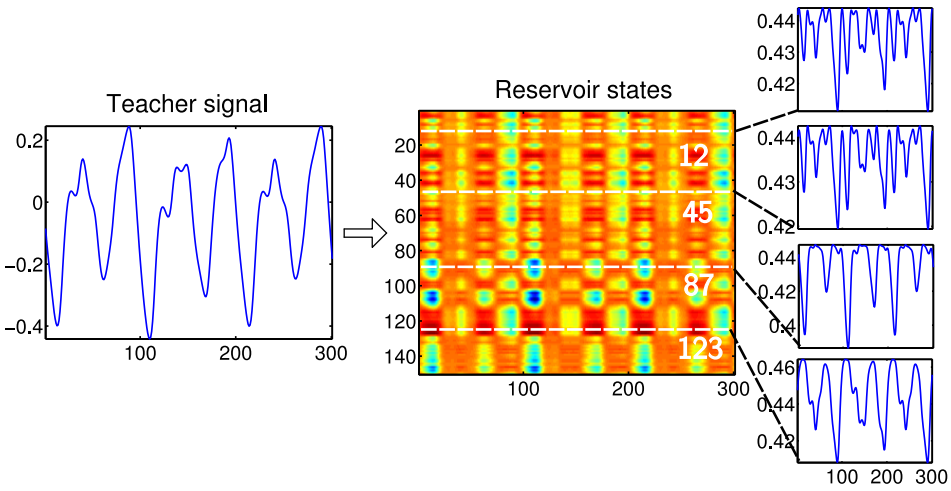

Fig. 5: Reservoir states for the delay reservoir system (3)

The Wolf number (also known as the International sunspot number, relative sunspot number, or Zürich number) is a quantity that measures the number of sunspots and groups of sunspots present on the surface of the sun.

The idea of computing sunspot numbers was originated by Rudolf Wolf in 1849 in Zürich, Switzerland and, thus, the procedure he initiated bears his name (or place). The combination of sunspots and their grouping is used because it compensates for variations in observing small sunspots.

This number has been collected and tabulated by researchers for around 300 years. They have found that sunspot activity is cyclical and reaches its maximum around every 9.5 to 11 years (note: Using data from SIDC for the last 300 years and running a FFT function on the data gives an average maximum at 10.4883 years/cycle). This cycle was first noted by Heinrich Schwabe in 1843.

The relative sunspot number R is computed using the formula (collected as a daily index of sunspot activity):

$$R = k(10g + s),$$

where

- s is the number of individual spots,
- g is the number of sunspot groups, and
- k is a factor that varies with location and instrumentation (also known as the observatory factor or the personal reduction coefficient K). [2]

Fig.6 show Wolf numbers records starting from 1750.

6.4 Next solar cycle prediction

Sunspot number prediction is very speculative task. Prediction on one, two or several points ahead is out of interest. More challenging task is to predict position and value of next maximum or minimum (solar cycle prediction). Fig.7 shows several possible behavior of sunspots number for the next 4 years starting from current date (September 2011). Dash-dot lines are predictions provided by delay reservoir system with six different best models selected from 500 randomly chosen. The parameters of the delay system were kept to be same as in Sec.6.1. Possible prediction trends are depicted in colors. For comparison the black dash line presents the prediction made by NASA solar activity research group (NASA/Marshal Space Flight Center, Solar Physics)[3]. NASA predicts next maximum in May 2013 with value 70. Our models predict it in April 2013 with 77, in April 2014 with 71 etc. For comparison in Fig.7b) the prediction made by conventional reservoir with 700 nodes are shown (see Sec.3–5). It predicts next maximum in October 2014 with value around 80.

Another interesting picture on prediction challenge is presented in Fig.8. Same range prediction issued by Space Weather Prediction Center of NOAA (Nation Oceanic and Atmospheric Administration). They predict next maximum in May 2013 with value 90.[4]

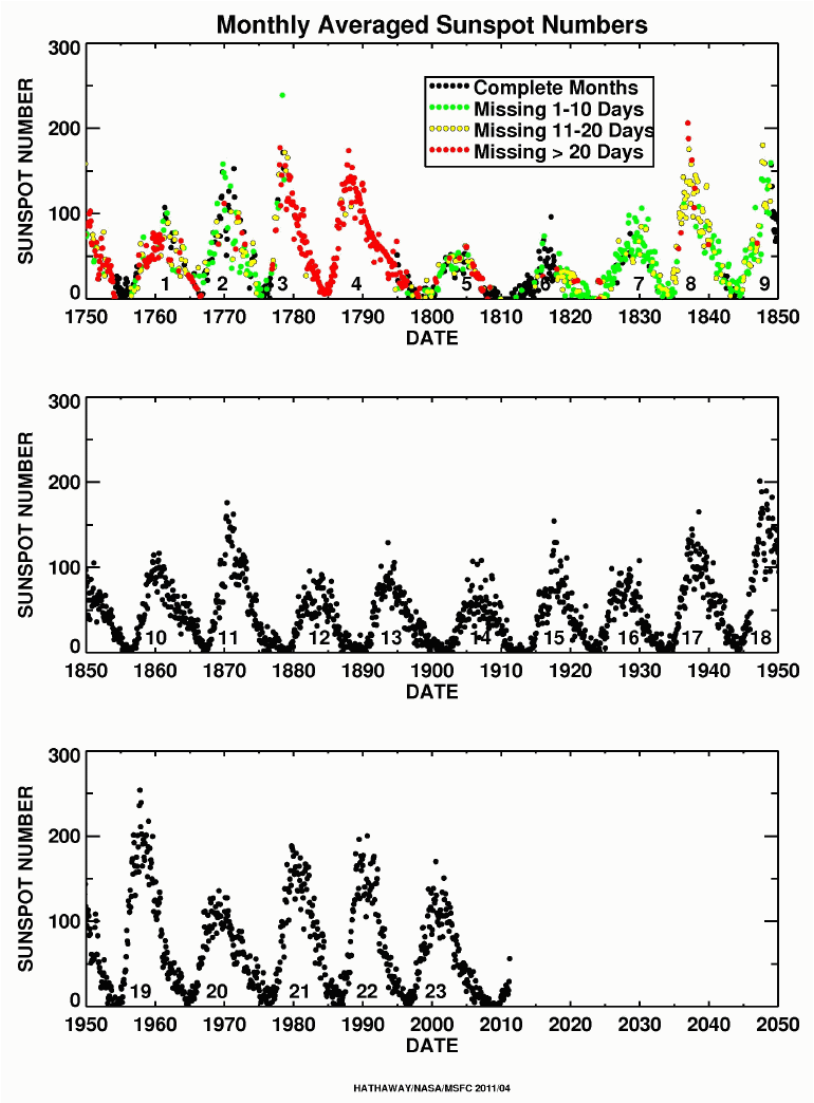

Fig. 6: Wolf number since 1750.

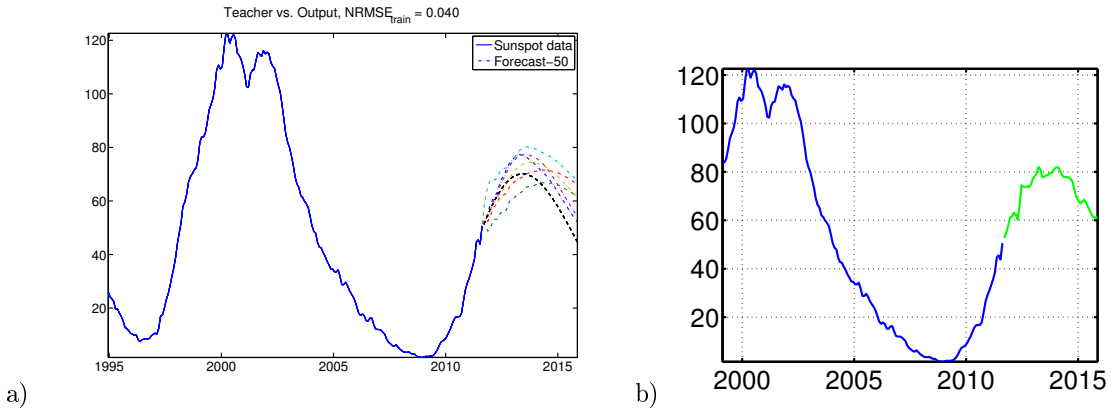

Fig. 7: a) Several possible continuations of sunspot number for the next 4 years based on delay reservoir b) Same prediction made by conventional reservoir with 700 nodes.

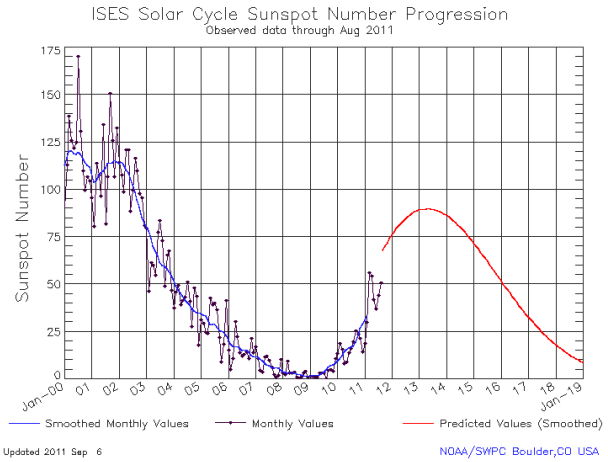

Fig. 8: NOAA Space Weather Prediction Center forecast

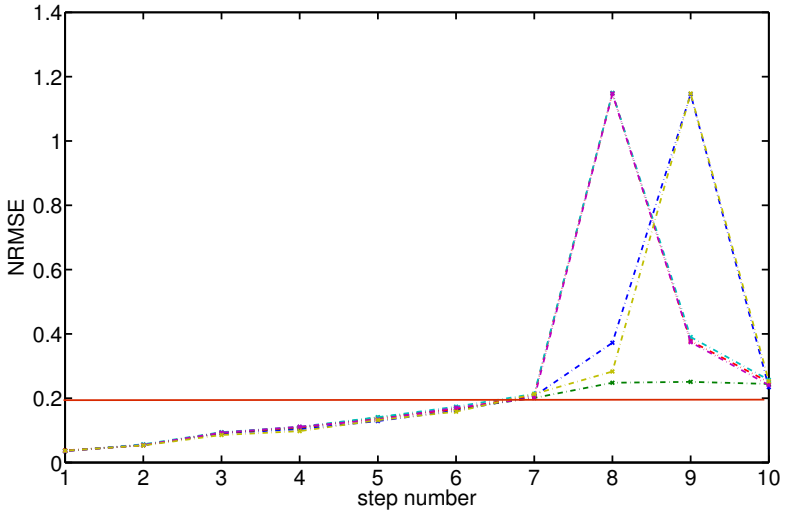

Fig. 9: Delay RC predictor test

6.5 Test delay reservoir predictor

All best models for delay RC system were analyzed on prediction ability described above (see explanation on calculation $NRMSE_{84}$). 100 equally spaced points of sun spot series were used as start points for prediction to $1, 2, 3...10$ steps forward with resulting evaluation of $NRMSE_{1,2,3...10}$. The result is presented in Fig9. It is clear that delay reservoir can predict next point up to 6 steps ahead with accuracy less then 20%.

Note that all analysis of delay RC system were made with fixed set of parameters presented in Sec.6.1. There is big room for investigation of predictions capability of delay RC system for different parameters.

7 Concluding remarks

All above presented procedures prove that building the delay reservoir network with ability to make a prediction task with sufficient accuracy is real. The only caution which arises here is that after training (sample run and evaluation of W_{out}), the system should evolve freely without any inputs. This means that electro optical system will start gradually to demonstrate its own dynamics (which controlled by intrinsic parameters such as: delay time, relaxation time, nonlinearity and so on) but by construction it is forced to produce prediction for external system (MGS, sunspots etc. in above examples). In any case electro-optical system might serve as a good predictor but for a limited period of time.

Technical details on attached codes presented in Appendices below.

8 Appendix A (MGS_ESN)

This part is about prediction of Mackey-Glass system by conventional ESN (Tab.1). It reproduces the result reported by Jaeger in [1].

Name	Description
`MGS_ESN.m`	Main script. It defines parameters (number of steps for train and test phases), topology of network (`topology` structure with number of nodes and connections properties), runs train phase and test phase. Optional it can run prediction test with additional parameters.
`generateNet.m`	Function. The script generates network model based on topology structure. It returns structure `model` with all connectivity matrices (W_{inp}, W_{int}, W_{ofb} and W_{out}. W_{out} is non-trained at this stage and has all elements equal to 0). One can say that the pair `(topology,model)` totally define network. After training phase it can be saved for further analysis and benchmarking.
`generateMGSeq.m`	Function. This code generates a discrete step sequence of the continuous time MG system with $\tau = 17$ with time-step 1 from a random seed, so every call of this script produces another sequence. Output sequence is shifted by -1 and transformed by `tanh` function as it comes to reservoir through output node with `tanh` non-linearity.
`trainNet.m`	Function. It runs constructed network in train phase for `trainLength` update steps. Main internal states update function is governed by (1,2). Only last `(1-frac_init)*trainLength` states of reservoir are used for training (calculation of W_{out}) and collected in separate matrices for possible further analysis.
`testNet.m`	Function. This script run trained network for further test update steps. Teacher signal is not used. State of output node is fed back to reservoir. The script returns updated `totalstate` and collection of output node states.
`predictTestNet.m`	Function. In fact it is separate script enveloped in function, it uses newly defined parameters on number of steps and runs the trained network from zero state (without current `totalstate`). This script runs prediction test. It iterates the constructed network for `totalLength` steps forced by teacher signal. Every `trialShift` it saves `totalstate` of the network as starting point for trial predictions. Than it runs prediction trials for every saved state for `testStep` steps ahead. For all trials predicted values of output node and teacher signal subtracted and averaged in resulting $NRMSE_{testStep}$.
`f.m, fInverse.m`	Simple functions. Global activation function (and its inversion) for every node of network. It is simple `tanh()` and `atanh()` but extracted in separate scripts for generality of the code package on prediction.

Tab. 1: Prediction of Mackey-Glass system by conventional ESN

9 Appendix B (MGS_Delay)

This part is about prediction of Mackey-Glass system time series by the delay system (Ikeda) reservoir (3). Details of the codes are in Table 2

Name	Description
`MGS_Delay.m`	Main script. It defines parameters (number of steps for train and test phases), topology of network (`topology` structure with number of nodes and connections properties), runs train phase and test phase. Also it defines all parameters of delay dynamical system (3) as structure `params`. At the end it runs several (`NumofPred`) prediction tests with additional parameters. The prediction test is used as criteria to select best model generated between several trials (`NumofExpr`). After total run of the script one best model (`model, topology, params`) saved to hard disk for further analysis (see `AnalyzeBestDelay.m`)
`generateNet.m`	Function. The script generates network model based on topology structure. It returns structure `model` with all connectivity matrices (W_{inp}, W_{int}, W_{ofb} and W_{out}. W_{out} is non-trained at this stage and has all elements equal to 0). One can say that the pair `(topology,model)` totally define network. After training phase it can be saved for further analysis and benchmarking.
`generateMGSeq.m`	Function. This code generates a discrete step sequence of the continuous time MG system with $\tau = 17$ with time-step 1 from a random seed, so every call of this script produces another sequence. Output sequence is shifted by -1 and transformed by `tanh` function as it comes to reservoir through output node with `tanh` non-linearity.
`trainNetDelay.m`	Function. It runs constructed network in train phase for `trainLength` update steps. Main internal states update function is `electrooptic1step.m` which is governed by behavior of the delay dynamical system. Only last `(1-frac_init)*trainLength` states of reservoir are used for training (calculation of W_{out}) and collected in separate matrices for possible further analysis.
`electrooptic1step.m`	Function. Calculate next (internal) state of delay reservoir based on delay dynamics (3). Next internal state is updated states of all virtual nodes after one delay time τ. Excitations of the system come from current states on input and output nodes translated to virtual nodes by connectivity matrices (W_{inp} and W_{ofb}).
`testNetDelay.m`	Function. This script run trained network for further test update steps. Main internal states update function is `electrooptic1step.m` which is governed by behavior of the delay dynamical system. Teacher signal is not used. State of output node is fed back to reservoir. The script returns updated `totalstate` and collection of output node states.
`predictTestNetDelay.m`	Function. In fact it is separate script enveloped in function, it uses newly defined parameters on number of steps and runs the trained network from zero state (without current `totalstate`). This script runs prediction test. It iterates the constructed network for `totalLength` steps forced by teacher signal. Main internal states update function is `electrooptic1step.m`. Every `trialShift` it saves `totalstate` of the network as starting point for trial predictions. Than it runs prediction trials for every saved state for `testStep` steps ahead. For all trials predicted values of output node and teacher signal subtracted and averaged in resulting $NRMSE_{testStep}$.
`f.m, fInverse.m`	Simple functions. Global activation function (and its inversion) for output node of network only. It is simple `tanh()` and `atanh()` but extracted in separate scripts for generality of the code package on prediction. For delay reservoir it can eliminated. The only reason to keep it even for delay dynamics is to constrain a data values (which can be different nature) in range $[-1; 1]$.
`AnalyzeBestDelay.m`	Separate script with extensive analysis of saved models (networks). It loads previously saved model, run `trainNetDelay` to warm-up reservoir, run free continuation prediction (`testNetDelay`), show pictures of reservoir states and run prediction test similar to the main script `MGS_Delay.m`. *Note*: it will never train the network. The purpose of the script is analysis of previously trained reservoirs.

Tab. 2: Prediction of Mackey-Glass system time series by the delay system (Ikeda) reservoir

10 Appendix C (SunSpot_Delay)

This part is about prediction of Sunspots numbers time series by the delay system (Ikeda) reservoir (3). Details of the codes are in Table 3. The Sunspot numbers data can be found in [5]

Name	Description
`SunSpot_Delay.m`	Main script. It defines parameters (number of steps for train and test phases), topology of network (`topology` structure with number of nodes and connections properties), runs train phase and test phase. It loads Sunspot time-series from file `monthssn_mod.dat` and transform it to teacher format (`f.m`). Also it defines all parameters of delay dynamical system (3) as structure `params`. At the end it runs prediction test with additional parameters. The prediction test is used as criteria to select best model generated between several trials (`Ntrials`). After total run of the script one best model (`model, topology, params`) saved to hard disk for further analysis (see `AnalyzeSolarDelay.m`)
`generateNet.m`	Function. The script generates network model based on topology structure. It returns structure `model` with all connectivity matrices (W_{inp}, W_{int}, W_{ofb} and W_{out}. W_{out} is non-trained at this stage and has all elements equal to 0). One can say that the pair `(topology,model)` totally define network. After training phase it can be saved for further analysis and benchmarking.
`trainNetDelay.m`	Function. It runs constructed network in train phase for `trainLength` update steps. Main internal states update function is `electrooptic1step.m` which is governed by behavior of the delay dynamical system. Only last `(1-frac_init)*trainLength` states of reservoir are used for training (calculation of W_{out}) and collected in separate matrices for possible further analysis.
`electrooptic1step.m`	Function. Calculate next (internal) state of delay reservoir based on delay dynamics (3). Next internal state is updated states of all virtual nodes after one delay time τ. Excitations of the system come from current states on input and output nodes translated to virtual nodes by connectivity matrices (W_{inp} and W_{ofb}).
`testNetDelay.m`	Function. This script run trained network for further test update steps. Main internal states update function is `electrooptic1step.m` which is governed by behavior of the delay dynamical system. Teacher signal is not used. State of output node is fed back to reservoir. The script returns updated `totalstate` and collection of output node states.
`predictTestNetDelay.m`	Function. In fact it is separate script enveloped in function, it uses newly defined parameters on number of steps and runs the trained network from zero state (without current `totalstate`). This script runs prediction test. It iterates the constructed network for `totalLength` steps forced by teacher signal. Main internal states update function is `electrooptic1step.m`. Every `trialShift` it saves `totalstate` of the network as starting point for trial predictions. Than it runs prediction trials for every saved state for `testStep` steps ahead. For all trials predicted values of output node and teacher signal subtracted and averaged in resulting $NRMSE_{testStep}$.
`f.m, fInverse.m`	Simple functions. Global activation function (and its inversion) for output node of network only. It is simple `tanh()` and `atanh()` but extracted in separate scripts for generality of the code package on prediction. For delay reservoir it can eliminated. The only reason to keep it even for delay dynamics is to constrain a data values (which can be different nature) in range $[-1; 1]$.
`AnalyzeSolarDelay.m`	Separate script with extensive analysis of saved models (networks). It loads previously saved model, loads sunspot data, run `trainNetDelay` to warm-up reservoir, run free continuation prediction (`testNetDelay`) for next solar cycle prediction, show pictures of prediction with comparison to NASA prediction and run prediction test similar to the main script `SunsSpot_Delay.m`. *Note*: it will never train the network. The purpose of the script is analysis of previously trained reservoirs.

Tab. 3: Prediction of Sunspots numbers time series by the delay system (Ikeda) reservoir

11 Appendix D (SunSpot_ESN)

This part is about prediction of Sunspots numbers time series by conventional ESN reservoir (1,2). Details of the codes are in Table 4. The Sunspot numbers data can be found in [5]

Name	Description
`SunSpot_ESN.m`	Main script. It defines parameters (number of steps for train and test phases), topology of network (`topology` structure with number of nodes and connections properties), runs train phase and test phase. It loads Sunspot time-series from file `monthssn_mod.dat` and transform it to teacher format (`f.m`). At the end it runs free-run test. The performance of the free-run test is used as criteria to select best model generated between several trials (`Ntrials`). After total run of the script one best model (`model, topology`) saved to hard disk for further analysis.
`generateNet.m`	Function. The script generates network model based on topology structure. It returns structure `model` with all connectivity matrices (W_{inp}, W_{int}, W_{ofb} and W_{out}. W_{out} is non-trained at this stage and has all elements equal to 0). One can say that the pair `(topology,model)` totally define network. After training phase it can be saved for further analysis and benchmarking.
`trainNet.m`	Function. It runs constructed network in train phase for `trainLength` update steps. Main internal states update function is governed by (1,2). Only last `(1-frac_init)*trainLength` states of reservoir are used for training (calculation of W_{out}) and collected in separate matrices for possible further analysis.
`testNet.m`	Function. This script run trained network for further test update steps. Teacher signal is not used. State of output node is fed back to reservoir. The script returns updated `totalstate` and collection of output node states.
`f.m, fInverse.m`	Simple functions. Global activation function (and its inversion) for every node of network. It is simple `tanh()` and `atanh()` but extracted in separate scripts for generality of the code package on prediction.

Tab. 4: Prediction of Sunspots numbers time series by conventional ESN reservoir

References

[1] H.Jaeger, H. Haas (2004): Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science, April 2, 2004, 78-80

[2] http://www.sidc.be/news/106/sunspotnumberclarified.pdf

[3] http://solarscience.msfc.nasa.gov/predict.shtml

[4] http://www.swpc.noaa.gov/SolarCycle/

[5] http://sidc.oma.be/sunspot-data/