Algorithm:

for supervised learning with a teaching signal (or a sample desired time-sequence):

// generate sparse density of reservoir, input and feedback matrices //

Win = 2.0 \* rand(reservoirSize, 1+inSize)- 1.0;

Wfb = 2.0 \* rand(reservoirSize, outSize)- 1.0;

// reservoir matrix should be generated with a definite spectral radius to avoid chaotic behavior in the reservoir//

conenctivity = min(10/reservoirSize, 1)

while success == 0

% following block might fail, thus we repeat until we obtain a valid

% internalWeights matrix

try

W = sprand(resSize, resSize, connectivity);

W(W ~= 0) = ...

W(W ~= 0) - 0.5;

opts.disp = 0;

alpha = max(abs(eigs(W,1, 'lm', opts)));

W = W/alpha;

success = 1 ;

catch

success = 0 ;

end

end

W = full(W);

The above weight initializations are to make the reservoir exhibit a stable behavior, and this is not supported by any mathematical proof other than experimental success.

Note that in the above W calculation, spectral radius is not multiplied to W, as I have used it as an additional parameter below. In supervised learning setting, this should be multiplied to the resultant W as spec\_radius\*W.

when the network is supplied with an input U, the output and state activations at each instant of time should be calculated as:

for each sample n:

x(n+1) = (1 – leak)\*x(n) + tanh( Win\*[1; U(n+1)] + spec\_radius\*W\*x(n) + Wfb\*y(n) )

y(n+1) = Wout\*[1; U(n+1); x(n+1)]

where Wout is the read-out neuron weight(the parameter to be learnt), leak is the leaking rate of the reservoir, spectral radius determines the type of signals that can be learnt by a given reservoir. i.e., fast changing signals or slow changing signals, usually this is set by a person based upon intuition.

**Training the Network:**

The supervised learning mechanism is very simple as given in the tutorial of ESN’s, and can learn any complex trajectory, given good values of “leak” and “spec\_radius”

But, in optimization settings like Genetic Algorithms, or Policy Gradient based reinforcement learning(instead of calculating an analytical gradient for the given cost function, it is estimated through trial and error by perturbing the policy-parameters(ESN parameters). MATLAB’s fmincon is also similar as it does not require gradient specification), both “leak” and “spectral radius” can be considered as extra variables along with the read-out weights Wout. All these parameters, Wout+leak+spec\_radius, should be learnt to reduce a given policy-dependent cost.