Home-Made NTC code: simulating Nonlinear Transient Computing (NTC) with delay dynamics

Laurent Larger
†Université de Franche–Comté, FEMTO–ST Institute, UMR CNRS 6174
Optics Dept. Pierre–Michel Duffieux, OPTO group,
15B Avenue des Montboucons, 25030 Besançon, France

1 Objectives, background, motivations

This document is intended to give a description of the different steps involved in a program intended to a fully computer-simulated classification task of the spoken digit recognition (SDR), via a *Nonlinear Tansient Computing* (NTC) approach implemented with an Ikeda-type nonlinear delay differential dynamics.

In the Computer Science and Machine Learning community, NTC is more frequently referred as to *Reservoir Computing* [1], initially introduced independently in the literature as *Echo State Network* [2] and *Liquid State Machine* [3]. An interesting review of the concept can be found in [4]. The term NTC is here preferred because of many sceptic Physics reviewers' reporting on the first publications submitted to the Physics community about this originally Computer Science and Brain Cognitive Sciences concepts. However, due to successful and unusual hardware implementation of the RC/ESN/LSM concepts, it indeed appeared relevant to make an attempt for a more meaningful naming of the concept within the Nonlinear Physics & Nonlinear Dynamics community.

The Home-Made code for NTC is strongly inspired by the RC-Toolbox designed by Ghent University (later re-programmed into python), a Toolbox however very difficult to use for Physics community people (most probably the Toolbox was thought for a usage within the Machine Learning community).

2 A possible Big Picture of the Concept

Before describing into details the code lines programmed in Matlab for a spoken digit classification task through a delay dynamics NTC, we propose to introduce the NTC concept with a very rough overview of the different processing steps. This will allow to introduce, hopefully in a clear way, the notations and the variable names used later in the program.

Figure 1 represents the processing steps over time within an NTC. The input information $\overrightarrow{u}(n) = (u_1(n), \ldots, u_j(n), \ldots, u_{N_f}(n))$ is here assumed as a discrete time information flow, n being the number of time steps from a starting date $n = 1$ and ending at $n = N_s$. The vector $\overrightarrow{u}$ has a dimension N_f, which will be 86 in the case of an SDR task. This number N_f corresponds

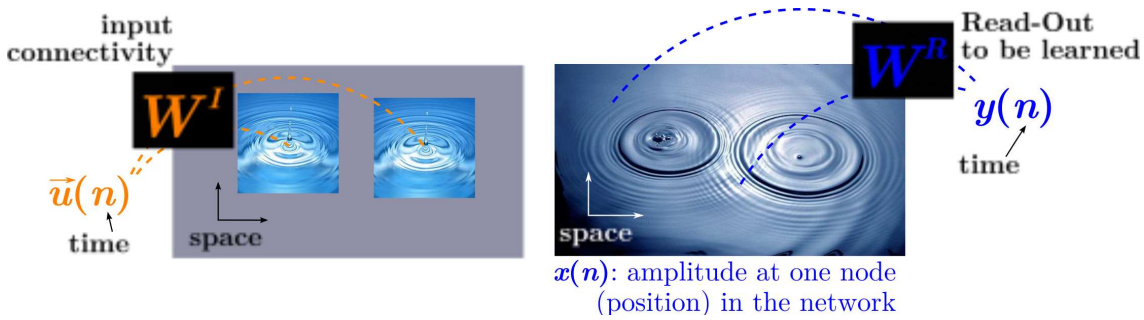

Figure 1: NTC processing steps, viewed through transient ripples at the surface of a liquid, trigerred by the "thrown" input information, and Read-Out to compute the result.

to the number of Fourier frequency channels into which the 1D input acoustic waveform will be converted via a so-called cochleagram (Time-Frequency representation, known as the Lyon-Ear model). Time n refers to the number of successive time slots over the full duration of the digit pronounciation, each time slot providing a 86-Fourier frequency channel decomposition. This input data can be represented as a cochleagram matrix M^c of dimension $N_f \times N_S$ with elements $[m^c_{jn} = u_j(n)]$, which columns are corresponding to the vecteur $\overrightarrow{u}$, i.e. the Fourier spectrogram over the selected 86 frequency channels, as time n is running.

The input data $\overrightarrow{u}$ is then expanded with many different coupling coefficient onto each of the virtual nodes of the nonlinear delay dynamics. This expansion, or addressing, of the input samples is performed by the multiplication of the cochleagram by a so-called input connectivity matrix W^I of dimension $N_{\text{node}} \times N_f$, where N_{node} is the number of virtual nodes provided by the delay dynamics. The elements of W^I are generally randomly chosen, and this matrix is typically defined with a certain sparsity ($1/10^{\text{th}}$ in our case). The input of the delay dynamics takes then the form of a matrix $M^u = W^I \times M^c$ of dimension $N_{\text{node}} \times N_S$ and having elements $[m^u_{kn}]$.

Remark: each element $[m^u_{kn}]$ of M^u corresponds by definition of the matrix product, to a linear combination of the amplitude of a few selected (according to the positions of non-zero elements in a line of W^I) Fourier components (the elements of a column in M^c) at a given time n:

$$m^u_{kn} = \sum_{i,j} w^I_{ki} \cdot m^c_{jn}$$

The coefficients w^I_{ki} of this linear combination of the Fourier amplitudes m^c_{jn} at time n are defined by the non-zero elements of each line k in the matrix W^I.

Since our delay dynamics is a purely temporal system, the input information to be injected in this dynamics is necessarily a 1D-signal. The latter is obtained from the matrix M_u by taking each column of N_{node} elements as a sequential amplitude applied to each virtual node within one time delay. Stacking horizontally each column after one another, one gets the 1D-signal to be injected into the delay dynamics. This signal $u_{\text{in}}(t)$ can then be defined as

$$u_{\text{in}}(t) = \sum_{k,n} m^u_{kn} \cdot p_{\delta\tau}(t - n\tau_D - k\,\delta\tau), \tag{1}$$

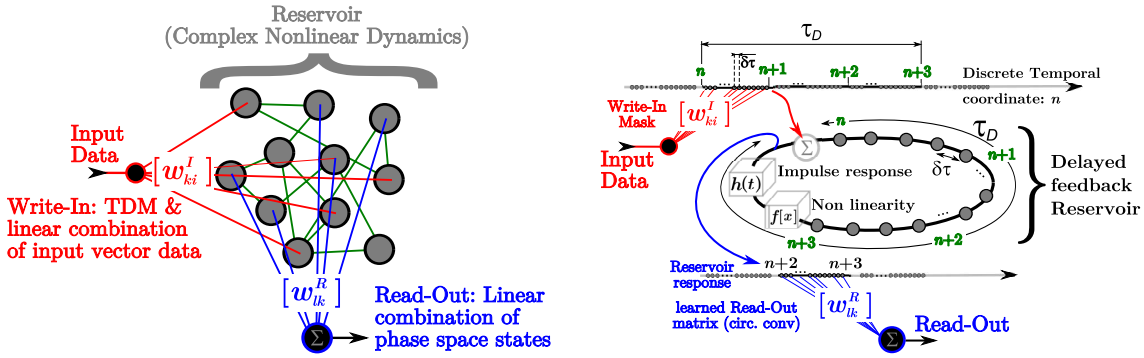

Figure 2: Picture view of the transposition from a spatially extended neural network dynamics to a time dynamics, together with its time division multiplexing (TDM) implementation for the virtual node addressing (in this example, the TDM interleaving period is 3).

where $p_{\delta\tau}(t)$ is a rectangular temporal door starting at $t = 0$ and with a width $\delta\tau$, the latter being the temporal spacing between two neighboring virtual nodes. The quantity τ_D is the time delay of the dynamics, over which N_{node} are virtually expanded. Each time delay interval corresponds then physically to one column of the matrix M^u.

Remark: The expansion of the input information over each virtual node of the delay dynamics can be interpreted in terms of signal theory. This corresponds to a well known technique referred as to time division multiplexing. This technique is however generally implemented with interleaved pulses originating from different sources of information, the interleaving being periodically repeated over the common bit period of each information source. In our case, the interleaving is temporaly ruled by the sampling time of the original signal $\overrightarrow{u}$, i.e. with a symbol period corresponding to an increment from n to $n + 1$. This symbol period will correspond physically to one time delay τ_D, but situations with an integer number of time delays have been experimented as well, and can represent also a signal processing architecture providing significant computation efficiency improvements.

Figure 2 represents a graphical view of the delay dynamics processing with its input information injection following the time division multiplexing principle.

The signal $u_{\text{in}}(t)$ is programmed in an AWG (arbitrary waveform generator) which is physically connected to an input of the photonic delay dynamics. The latter dynamics provide an output signal (or *nonlinear transient*) $x(t)$, which is ruled by the following non-autonomous (since the external signal $u_{\text{in}}(t)$ is contributing to its motion) delay differential equation:

$$\tau \frac{\mathrm{d}x}{\mathrm{d}t}(t) = -x(t) + \beta \, \sin^2 \left[\alpha \, x(t - \tau_D) + \rho \, u_{\text{in}}(t - \tau_D) + \Phi_0\right], \tag{2}$$

where α is wether 0 (open loop processing) or 1 (closed processing, with indeed a recurrent action of the past samples on the present ones), and ρ is an amplification factor for the information injected into the delay dynamics (this weight typically defines how nonlinear will be the transient the $\sin^2 -$ function). The parameter β represents the weight of the feedback in the delay loop and its setting has to fulfill stability conditions of the dynamics

in autonomous regime (what is referred as to a spectral radius smaller than unity in the conventional dynamics in the form of a network of interconnected neurons). The parameter Φ_0 is an offset phase setting the average rest point along the nonlinear function when no information is present (autonomous regime or $u_{\rm in} = 0$). For values close to 0 or $\pi/2$ modulo π, the nonlinear function operates as a quadratic non linearity, whereas it is linear of cubic for Φ_0 close to $\pm\pi/4$.

The previous delay differential equation (2) is simulated with a $4^{\rm th}$ order Runge-Kutta algorithm having a constant integration time step h defined as a fraction of the internal response time of the dynamics τ (so that a quasi-continuous time dynamics can indeed be simulated). The signal $x(t)$ develops a transient motion in the infinite dimensional phase space of the delay dynamics. Sampled states $x(t_{mp}) = x_{mp}$ within this phase space will be later used as the input data for the remaining Read-Out operation which will provide the computed answer for the original problem, i.e. in our case the classification of the spoken digit initially encoded in $\overrightarrow{u}(n)$.
The sampling times t_{mp} of the Read-Out are not fixed a priori, however considering the timing procedure of the Write-In, a natural choice for this sampling dates are $t_{mp} = m\,\delta\tau + p\,\tau_D$, which are thus consistent with the Write-In timing.
The transient response of the dynamics can then accordingly be represented by a response matrix M^x of elements $[x_{kn}]$, which columns indexed by n can be interpreted similarly to those of the matrix M^u as the virtual node amplitudes over one time delay.

Learning how to Read-Out adequately the nonlinear transient response $x(t)$ when excited by the input information $\overrightarrow{u}(n)$, can be reduced to a simple ridge regression procedure. Indeed, if ones knows a number of couples of encoded spoken digits and their corresponding digit value, the Read-Out issue can be written as follows with a matrix notation:

$$W^R \times M^x = M^{\rm Tg}, \tag{3}$$

where $M^{\rm Tg}$ is a so-called target matrix describing unambiguously the actual value of the digit spoken during the N_S initial samples of $\overrightarrow{u}(n)$. The dimension of $M^{\rm Tg}$ can be defined as N_S columns corresponding to a spoken digit duration, and $N_{\rm mod}$ lines corresponding to the different modalities or possible solutions. A basic construction of $M^{\rm Tg}$ could consist in setting to 1 all the elements of the right modality for a given spoken digit, and all the other lines of the wrong remaining modalities to 0.
For a set of $N_{\rm learn}$ learning couples $(M^x, M^{\rm Tg})$, one can concatenate horizontally all M^x matrices into a big matrix A of dimension $N_{\rm node} \times N_L$, where $N_L = \sum_{l=1}^{N_{\rm learn}} N_S(l)$ with $N_S(l)$ being the spoken digit duration of the $l^{\rm th}$ digit in this learning set. The same concatenation has then to be done with the $N_{\rm learn}$ target matrices, resulting in a matrix B of dimension $N_{\rm mod} \times N_L$.
The ridge regression has the role to calculate the optimal matrix $W^R_{\rm opt}$ providing the best possible answer for all the elements of the training set. A single Matlab code line performs this ridge regression through the use of a regression parameter λ and a classicial More-Penrose matrix inversion algorithm available in Matlab through the backslash symbol "\":

$$(W^R_{\rm opt})^{\rm t} = \left(A\,A^{\rm t} - \lambda\,I_{N_{\rm node}}\right)^{-1}\,A\,B^{\rm t}, \tag{4}$$

where $I_{N_{\rm node}}$ is the identity matrix of dimension $N_{\rm node}$ and $(\cdot)^{\rm t}$ denotes the matrix transposition operation.

The calculated Read-Out matrix is then applied to spoken digits that are not belonging to the learning set, i.e. a testing set of N_{test} spoken digits which is complementary with respect to the learning set in the global set of digits to be process for classification.
When testing, the resulting calculated target matrices $\tilde{M}^{\text{Tg}} = W_{\text{opt}}^{R}\, M^{x}$ are not perfectly shaped in the form of the learning target matrix M^{Tg}. One needs then to define classification criteria in order to interpret as properly as possible the approximated computed value of the each tested digit. A simple *winner-take-all* procedure is applied, summing over time the amplitudes along each line of modality in $\tilde{M}^{\text{Tg}}$. This results in a modality vector, the selected modality being the one with the highest score.

Knowing actually the answers of both the learning and testing sets, one can easily count, identify, and analyze the errors occurring under different processing conditions (the parameter of the dynamics, e.g. τ, τ_D, β, Φ_0, and the parameters of the Write-In and Read-Out, e.g. $\delta\tau$, N_f, λ, N_{learn} or N_{test}, etc...). This analysis makes use for example of the WER (word error rate) typically given in %. This number however might be inconvenient for discussion when the performance reaches a level above a certain classification quality, due to the rather limited size of the test (500 different spoken digits for 10 modalities, thus 50 digits for each of the same modalities). One erroneous digit among the 500 leads to a WER of 0.2%, and zero errors ony means the WER is below that minimum value.
Another indicator, with however a continuous measure instead of the discrete one issued from the count of errors, is given by the *score margin* between the correct digit with the highest score, and the first wrong digit with the second highest score. This continuous measure can reveal how robust is actually the classification.

3 The spoken digit recognition task

This task is one of the different benchmarks used in Machine Learning and Neural Network Computing in order to measure the computational efficiency of a given method. It was used in several recent papers on Reservoir Computing [5, 6, 7, 8], and was apparently first introduce in [9].

It consists of a database of 500 recorded spoken digits extracted from the TI-46 corpus. The ten digits from zero to nine are spoken 10 times by 5 different female speakers, thus constituting the 500 spoken digits database.

In order to provide comparable processing efficiency with other methods, we have followed a standard pre-processing step called the Lyon Ear Cochlea model, which is independent from the Nonlinear Transient Computing concept, but which is usually done when performing speech recognition tasks. This model transforms the original 1D acoustic signal into a spectrogram, as it is naturally done within the Human ear. The 12 kHz sampled acoustic pressure variation corresponding to the speech, is transformed into a 2D time-frequency representation, revealing the temporal evolution of the energy distribution in the Fourier domain according to 86 pre-defined frequency channels. A spoken digit recorded with ca. 10^4 amplitude samples is thus converted into a sliding spectral density of 86 channels over ca. 100 consecutive time slots (one time slot covering physically one time delay). This spectrogram represents the input information $\overrightarrow{u}(n)$ to be processed by the delay dynamical system according to the Reservoir Computing concept.

4 Our Matlab program

The Matlab code "NTC_dde.m" is organized along the following main lines:

- Construction of the 1D input signal information $u_{\text{in}}(t)$, also called the Write-In, which is the source of the transient generated by the delay dynamics. Since this signal is based on a sequence of amplitudes m^u_{kn}, it takes physically the form described by Eq.(1) and corresponding to a sample and hold with order zero, i.e. to a piecewise constant signal. The sampling period is defined by the choice of the node spacing $\delta\tau$, essentially to be scaled with respect to the internal time of the delay dynamics, i.e. τ in Eq.(2) (a common value is ca. $\delta\tau \simeq 0.2\tau$).

  Before this input signal definition, one needs to perform the eventual appropriate pre-processing, e.g. such as the Passive Lyon Ear Model, as well as the definition of a random input mask (definition of the matrix W^I) requirred to construct the Write-In signal. The matrix W^I experiences an amplitude normalization such that the resulting signal $u_{\text{in}}(t)$ for the whole set of spoken digits, has a peak-to-peak amplitude normalized to one. A scaling factor ρ is then used to adjust the weight of the information injected into the delay dynamics, as defined in Eq.(2).

- Following the way $u_{\text{in}}(t)$ is practically processed in the physical experiment, a series of digits are actually gathered in several sequences (the number of spoken digits per sequence depends on the intrenal memory of the AWG used to generate $u_{\text{in}}(t)$, typ. 10 to 20 per sequence, leading to 25 to 50 sequences in order to have all the 500 digits processed). The digits are randomly permuted to avoir any biais potentially introduced through the originally ordered arrangement of the digits. Thirty time delays of zeros are added between the digits forming a sequence, in order to avoid any influence of one digit onto the next.

- A mex-compiled C routine "rk4_ntc.mexglx" is called to perform the numerical integration of the transient generated through Eq.(2). The output time series "yc" is then processed for sampling and extraction of the response matrix M^x, gathering all of them in a big matrix A and building at the same the target matrices used to construct the big target matrix B.

- Training (or learning) and testing is then performed, extracting the various measures of the NTC efficiency (classification errors and margins), and performing the same learning & testing processing for several partitions of the whole set of digits into complentary training and testing sets (this provides a so-called cross-validation of the training performances).

- The previous steps can be repeated for a set of parameter values defined in the beginning of the code in the vector "Vp". Results for each calculation are stored, and finally represented at the end of this main calculation loop.

4.1 Main variables

4.2 Individual processing steps

4.3 Running the program

On a 64 bits laptop installed with Ubuntu, equipped with an Intel i7 quad-core 1.8 GHz, a full processing of the 500 digits by a delay dynamics with 306 virtual nodes, with a 20-fold cross-validation for the learning and testing, takes approx. **12 minutes**.

References

[1] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," Neural Networks. (2007).

[2] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," GMD Report 148 (2001).

[3] W. Maass, T. Natschläger, and H. Markram, "Real–Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," Neural Computation **14,** 2531–2560 (2002).

[4] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," "Computer Science Review" **3,** 127–149 (2009).

[5] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the Liquid State Machine: a case study," Information Processing Letters **30,** 521–528 (2005).

[6] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," Nature Commun.(London) **2,** 1–6 (2011).

[7] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing," Opt. Express **20,** 3241–3249 (2012).

[8] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," Nature Commun. 4:1364 (2013).

[9] J. Hopfield and C. Brody, "What is a moment? "Cortical" sensory integration over a brief interval," Proc. Natl. Acad. Sci. USA **97,** 13919–13924 (2000).