SoftwareX 15 (2021) 100773

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

NetCausality: A time-delayed neural network tool for causality detection and analysis

Riccardo Rossi [a,*,1], Andrea Murari [b,1], Luca Martellucci [a], Pasquale Gaudio [a]

[a] Department of Industrial Engineering, University of Rome "Tor Vergata", via del Politecnico 1, Roma, Italy
[b] Consorzio RFX (CNR, ENEA, INFN, Università di Padova, Acciaierie Venete SpA), Corso Stati Uniti 4, 35127 Padova, Italy

ARTICLE INFO

Article history:
Received 16 July 2020
Received in revised form 1 July 2021
Accepted 16 July 2021

Keywords:
Causality detection
Time-delayed neural network
Time-series

ABSTRACT

The analysis of causality between systems is still an important research activity, which finds application in several fields of science. The software presented is a new tool for causality detection and analysis between time series. The proposed technique is based on time-delayed neural networks (TDNN). The tool is developed in MATLAB and it comprises three main functions. The first one returns the total causality between two or more systems of equations. The second tool is used to find the "time horizon", id est the time delay at which the influence between the systems occurs. The last function is a causality feature detection to determine the time intervals, in which the mutual coupling is sufficiently strong to have a real influence on the target.

Code metadata

Current code version	Version: 1.0
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-20-00005
Code Ocean compute capsule	
Legal Code Licence	BSD-3-Clause
Code versioning system used	
Software code languages, tools, and services used	MATLAB
Compilation requirements, operating environments & dependencies	MATLAB, neural network toolbox, statistics toolbox, parallel computing toolbox (optional)
If available Link to developer documentation/manual	(documentation/manual will be included in the repository with the last code version)
Support email for questions	r.rossi@ing.uniroma2.it riccardo.rossi.en@gmail.com

1. Motivation and significance

The analysis of the influence between physical systems evolving in time (time series) is still challenging, despite the great efforts exerted in the last decades and the variety of proposed methods [1]. The linear Granger, kernel Granger, mutual information and cross-mapping are few of the most used techniques. On the other hand, it has been demonstrated with various numerical tests that they can fail, return contradictory results and in general they are all more performing in certain specific applications than others [2–5]. Moreover, these methods are often much more reliable when the task is limited to the analysis of two scalar variables (evolving in time, of course). Unfortunately, in several applications, more than two observables can be linked together, and their mutual influence can be non-linear, rendering most of the causality detection approaches inadequate [6].

The software tool presented in this work implements the architecture of time-delayed neural networks (TDNN) for causality detection in physical time dependent systems. The topology of the used TDNN is reported in Fig. 1 [7]. TDNN have clearly demonstrated to be able to deal with challenging tasks in the field of time series, such the ones reported in [8–11].

The use of the TDNN, in the way proposed in this paper, is based on the causality concept defined by Wiener, which considers that if it is possible to predict Y more accurately using the

* Corresponding author.
E-mail addresses: r.rossi@ing.uniroma2.it (Riccardo Rossi), andrea.murari@istp.cnr.it (Andrea Murari), martellucci@ing.uniroma2.it (Luca Martellucci), gaudio@ing.uniroma2.it (Pasquale Gaudio).
[1] These authors have contributed equally.

https://doi.org/10.1016/j.softx.2021.100773

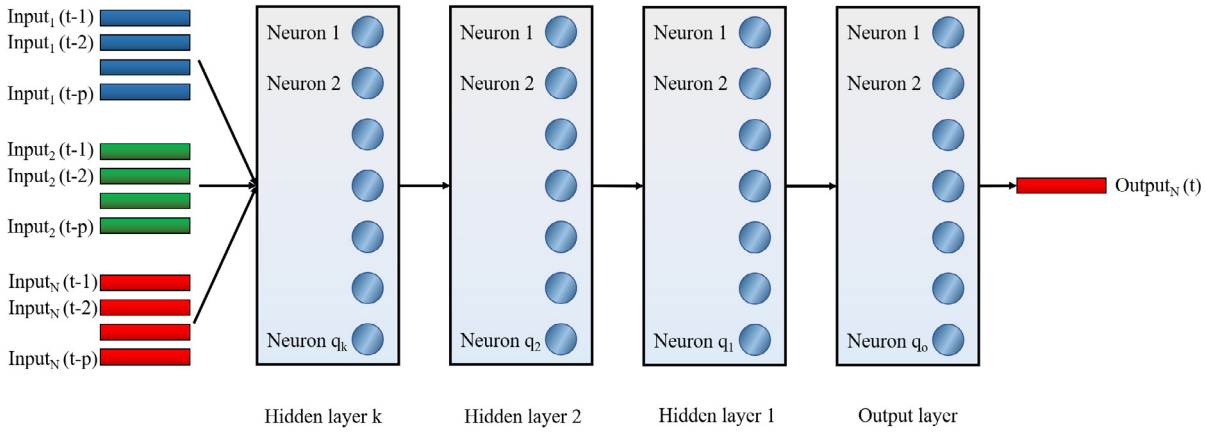

Fig. 1. General architecture of a Time Delay Neural Network implemented by the software package.

past values of X, then X can be assumed to influence Y (see also Fig. 2). All the time series under consideration are provided as inputs to the network, with their past values. Each of the time series is then predicted using, at first, all the signals available, and then by eliminating one input at a time. When an input is removed and the predictive capability of the network results in a significant reduction in the quality of its predictions, it is considered as evidence that the input has a causal influence on the one predicted [12].

The software package that implements this new approach consists of three different main functions, which address "the total causality detection", the measurement of "the time horizon" and the detection of the time intervals in which causality is active. The tool has no intrinsic limits regarding the number of systems, which can be analysed simultaneously. Moreover, each system can be made up of any number of equations, rendering the tool applicable to any type of time series.

Up to now, the tool has been tested on synthetic data, among which some of the most challenging consist of time series generated by three interacting Lorenz systems and autoregressive functions. The new method has demonstrated the capability of correctly detecting the causality between the three systems, handling well also non-linear influences between observables in the deterministic chaotic regime (also feedback loops).

2. Software description

The software provides three different tools with three specific functionalities. The first one is meant to calculate the global influence and it is named "Causality Detection". It returns information about the influence of the "*i*th" variable on the "*j*th" observable. The second tool is called "Time Horizon Detection" and aims to find the correct time delay at which the influence of the driver is transmitted to the target. Thus, if the previous tool indicates that the "*i*th" variable influences the "*j*th", using the time horizon feature it is possible to determine the time delay, with which the influence occurs. The last software tool is the Causality Feature Detection and it aims at finding the time intervals, in which the driver exerts an actual influence on the target (for example, sometimes actual influences can occur only above a certain threshold).

The software is composed of six scripts: three main functions, one for each tool (MAIN_TDNN_CausalityDetection, MAIN_TDNN_HorizonTimeDetection and MAIN_TDNN_Feature CausalityDetection) and three function scripts used by the mains (NN_config, NN_train, NN_test). NN_train and NN_test should not modified, while NN_config is a configuration file where the user can change the most important parameter of the net, such as the hidden layers (number and size), time delays, convergence parameters (minimum gradient, convergence goal, training algorithm). The codes are commented in details and their settings are user-friendly.

Consider N physical systems, where the *i*th system is compounded by M_i signals (or equations). The influence of the *i*th system on the *j*th one is computed by calculating the prediction error using all systems as input (E_{all}) and the prediction error excluding the *i*th system from the inputs (E_i).

The Causality detection tool perform a statistical test to evaluate if the variance of E_i is statistically different (and larger) from E_{all}. If the output has a p-value lower than the selected one, the *i*th is considered to influence the *j*th, otherwise not. The statistical test is the F-test of equality of variances for two samples (E_{all} and E_i).

The Time Horizon Detection works exactly as the Causality detection tool, but the analysis is performed varying the time delays. The software is open, and it allows choosing freely the combination of lags to investigate and it also highlights the first time delay (the shortest), at which a statistically relevant difference of the error is detected (using again the F-test).

The Causality Feature Detection compares the evolution of the errors in time. The absolute value of the error difference is directly correlated to the intensity of the influence, which the *i*th observable has on the *j*th one. The median value and the standard deviation of E_{all} are calculated and then the parameter feature is calculated as follows:

$$Feature\,(t) = \begin{matrix} 1 & if & \frac{|E_i(t) - median(E_{all}(t))|}{std(E_{all}(t))} > threshold \\ 0 & if & \frac{|E_i(t) - median(E_{all}(t))|}{std(E_{all}(t))} \leq threshold \end{matrix} \quad (1)$$

All the tools are provided with the parallel computing option, which allows to run most of the algorithm in parallel, reducing the time required to perform the analysis. The computational demand is strongly dependent on the boundary conditions, such as the number of inputs, the length of the time series, the complexity of the neural network, the convergence parameter, etc.

The software is also equipped with other three files, which are used to generate the input to run the illustrative examples in the following section and are described in the user manual provided in the software folder:

- MAIN_generate_system;
- MAIN_generate_system_forTimeHorizon;
- MAIN_generate_system_forFeatureDetection.

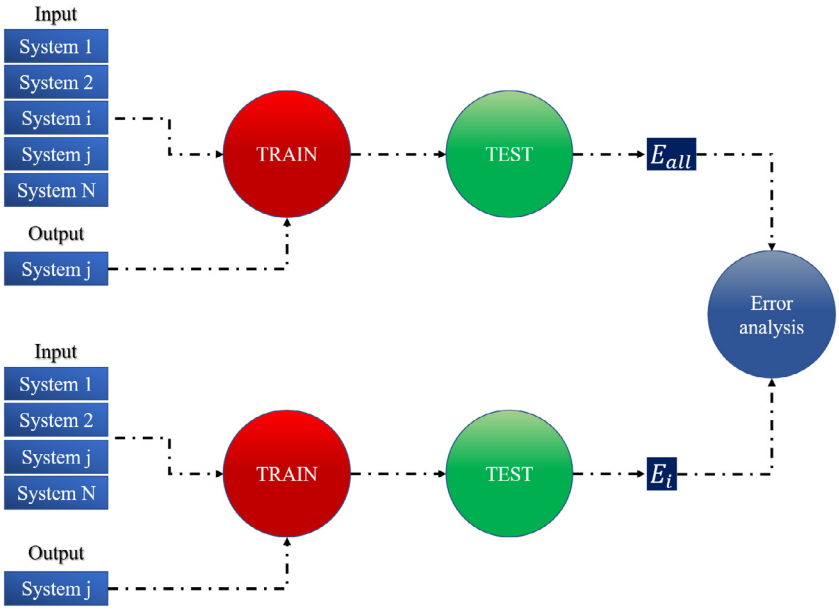

Fig. 2. Schematics of the main building blocks of the software package.

3. Illustrative examples

Some illustrative examples for each main functionality are shown in this section. Moreover, the same examples are reported in the user guide furnished to the journal together with the code. All the examples are ran using synthetic data generated by the Lorenz system. It has to be highlighted that by running again the examples, slightly different results can be obtained. At first, the generation functions have random initialization parameters and they also add random noise to the signals. Moreover, the data are strongly influenced by the convergence quality of the neural network. Sometimes, it may happen that the neural network does not reach the best convergence and false positives and false negatives may arise. For the moment, it is strongly suggested to run the algorithm more than one time and perform a statistic of the results.

The Causality Detection feature is assessed using three interacting Lorenz systems:

$$\textit{Sytem } X: \quad \begin{aligned} \tfrac{dx_1}{dt} &= \sigma\,(x_2 - x_1) \\ \tfrac{dx_2}{dt} &= r x_1 - x_2 - x_1 x_3 + \mu_{21} y_2^2 + \mu_{31} z_3^2 \\ \tfrac{dx_3}{dt} &= x_1 x_2 - b x_3 \end{aligned} \tag{2}$$

$$\textit{Sytem } Y: \quad \begin{aligned} \tfrac{dy_1}{dt} &= \sigma(y_2 - y_1) \\ \tfrac{dy_2}{dt} &= r y_1 - y_2 - y_1 y_3 + \mu_{12} x_2^2 + \mu_{32} z_3^2 \\ \tfrac{dy_3}{dt} &= y_1 y_2 - b y_3 \end{aligned} \tag{3}$$

$$\textit{Sytem } Z: \quad \begin{aligned} \tfrac{dz_1}{dt} &= \sigma(z_2 - z_1) \\ \tfrac{dz_2}{dt} &= r z_1 - z_2 - z_1 z_3 + \mu_{13} x_2^2 + \mu_{23} y_2^2 \\ \tfrac{dz_3}{dt} &= z_1 z_2 - b z_3 \end{aligned} \tag{4}$$

The coefficient μ_{ij} is the interaction or coupling coefficients. If it is equal to zero, it means that the ith system does not affect the jth ones, while if it is different from zero, the ith system influences the jth. In the following example, the interaction coefficients are $\mu_{21} = \mu_{31} = \mu_{32} = \mu_{13} = 0$, $\mu_{12} = 2$ and $\mu_{23} = 0.5$, thus X influences Y and Y influences Z. The output of the Causality Detection tool in this example is shown in Fig. 3. The software tool returns three matrices, which have a size NxN, where N is the number of analysed systems (three in this case). The first matrix returns the p-value of the F-test, while the second matrix reports a Boolean value calculated from the p-value matrix and the p-value threshold: if the p-value is larger than the threshold, the ith system does not influence the jth one. For the present example, as shown also in Fig. 3, it is found that X is not influenced by Y and Z (while it is influenced by its past values), Y is influenced by X and Y, and Z is influenced by Y and Z, as expected from the value of the coupling coefficients. The last matrix returns the variance ratio ($\mathrm{var}(E_i)/\mathrm{var}(E_{all})$). This parameter is interesting to quantify the correlation strength. As you can see, in the main diagonal, the variance ratio is very large since the prediction of the ith system, where without the knowledge of its past values, this is essentially impossible.

The second example uses the X system and a randomly generated y signal. The X system is influenced by the y signals with a time delay τ as follows:

$$\begin{aligned} x_1(t) &= x_1(t-1) + \sigma\,(x_2(t-1) - x_1(t-1))\,\Delta t \\ x_2(t) &= x_2(t-1) + (r x_1 x_1(t-1) - x_2 x_1(t-1) \\ &\quad - x_1 x_1(t-1)\, x_3 x_1(t-1) + \mu y^2(t-\tau))\,\Delta t \\ x_3(t) &= x_3(t-1) + (x_1(t-1)\, x_2(t-1) - b x_3(t-1))\,\Delta t \end{aligned} \tag{5}$$

The interaction coefficient has been set equal to 10. Fig. 4 shows the results obtained by running the Time Horizon Detection function, choosing $\tau = 3$ and scanning a time delay from 1 to 5. The error variance ratio is almost equal to 1 for the time delays 1:1 and 1:2; this is correct, since the values of $y(t-1)$ and $y(t-2)$ do not influence the value of the X system at the time t. Once the scanned time delay is 1:3, an increase of the error variance ratio is observed (since $y(t-3)$ influences the X system) and the decision from the F-test turns into one. Increasing the time delays, there is an increase of the error variance ratio. The tool highlights (red circle in Fig. 4) the first time-delay at which influence is observed (time horizon).

Another example shows the functionalities of the Causality Feature Detection. All the three systems are simulated, and the interaction coefficients are set to $\mu_{21} = \mu_{31} = \mu_{32} = \mu_{13} = 0$ and $\mu_{23} = 1$, while μ_{12} is equal to 1 if x_2 is higher than 10, otherwise it is equal to 0. This feature implies that X influences Y only in specific time intervals, which are reported in Fig. 5(a). Running the software, both the graphs in Fig. 5(b) and (c) are obtained. Graph (b) shows the absolute error $|E_i^2\text{-}E_{all}^2|$, which is characterized by a peaked structure. Then, using Eq. (1), the

Fig. 3. Output of the Causality Detection example.

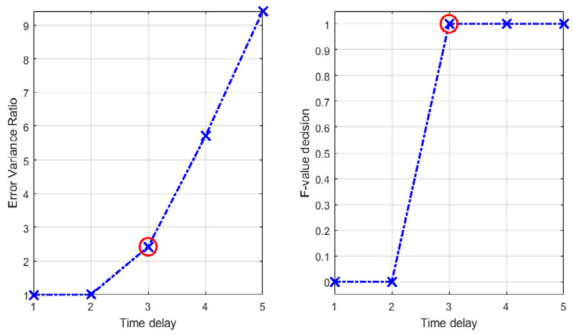

Fig. 4. Output of the Time Horizon Detection example.

intervals where the causality is high because above the threshold ($x2 > 10$) are detected and plotted, such as in Fig. 5(c).

In the end, we would like to compare the performances of this new tool with the standard methods in the literature for a quite simple example. In the review by Krakovská et al. [6], the authors tested six causality detection methods (Granger's vector autoregressive test, Extended Granger test, Kernel Granger test, Conditional Mutual Information, Convergent Cross Mappings, Predictability Improvement) with several bivariate time series and showed that there is not an algorithm that is suitable for every case. The one that showed the best performance is the predictability improvement, which however returns 71% of false negatives in the simple AR case. We have replicated the same AR model and tested our algorithm (see equation 1 of reference [6]), and the results are significantly better (neither false positives nor false negatives). It has to be noted that also the Granger-based algorithm and the Conditional Mutual Information produced the same results. However, it is easy to understand and also to verify that, if the coupling between the two-time series is not linear, the classical Granger will fail, while the Kernel Granger presupposes that the type of kernel to use is known. The Conditional Mutual Information is much more flexible to investigate non-linear influences, but its use is strongly limited to small-time series sizes (unless an enormous amount of data is available to calculate the N-size pdf). Our new tool, on the contrary, is potentially able to detect any functionality (we tested linear, polynomial, exponential, sine, power-law) and is computationally efficient.

4. Impact and limitations

The tests performed have shown that the proposed tools can always detect the right causal relations and the relative times for all the cases of mutual influence between three observables, summarized graphically in Fig. 6 [12,13]. It is well documented in the literature that all the other methods typically fail in at least one of the cases.

TDNNs are quite intuitive and do not require an excessive amount of data to provide robust conclusions. However, for small datasets (<1000 samples) it is suggested to perform various runs to improve the reliability and general performances. In terms of computational resources, they are also not excessively demanding. The computational load and time are generally higher than standard methods (mainly due to the training of the neural networks) and they depend on the amount of data to be analysed. However, since too complex neural network can be avoided, the computational time is acceptable (for example, in the case of the AR model (N sample $=$ 1e4), the computational time is around 135 s). They are potentially able to catch any type of influence (linear, polynomial, exponential, sine, power-law, etc.). Their capability to work with a high number of time series simultaneously constitutes a huge advantage with respect to most algorithms, such as Conditional Mutual Information and Transfer Entropy, which are usually used only in bivariate cases, due to requirement to calculate the conditional probability density functions.

The proposed approach is also general and can therefore be used in any field of science, in which casual relations have to be investigated. The main requirement is the quality of the available measurements. The remarkable reliability and flexibility of the provided tools are motivating their application in various fields, ranging from magnetically confinement thermonuclear fusion to atmospheric physics and epidemiology. The investigation of the mutual influences between various signals due to nonlinear instabilities in thermonuclear plasmas would certainly benefit significantly from the deployment of the developed tools [13–15]. The optimization of measurement methods for the surveys of the environment could also be expedited by the techniques proposed [16].

Even if the work presented in this paper demonstrates that the TDNNs may lead to great improvements to the field of causality detection, it is worth mentioning that, being the method based on the F-test, the performances are strongly influenced by the threshold (p-value). In fact, the p-value should be chosen by a trade-off between sensitivity and specificity of the algorithm. In order to obtain reliable outcomes, particularly in the case of limited samples, the authors strongly suggest calculating the results as the average values of several runs and also complementing the developed tools with surrogate data analysis. Moreover, this tool may be used in combination with other approaches with possibilities to increase its performances and the reliability. For example, it may possible to implement the use of the State Space and VAR models in to avoid the double regression problem [17].

5. Conclusions

The technology of Time Delay Neural Networks has been adapted to the detection of the causal relations between time series. The developed tools can be deployed not only to identify the links between observables but also to determine the time delay of the influence and the intervals, in which the driver

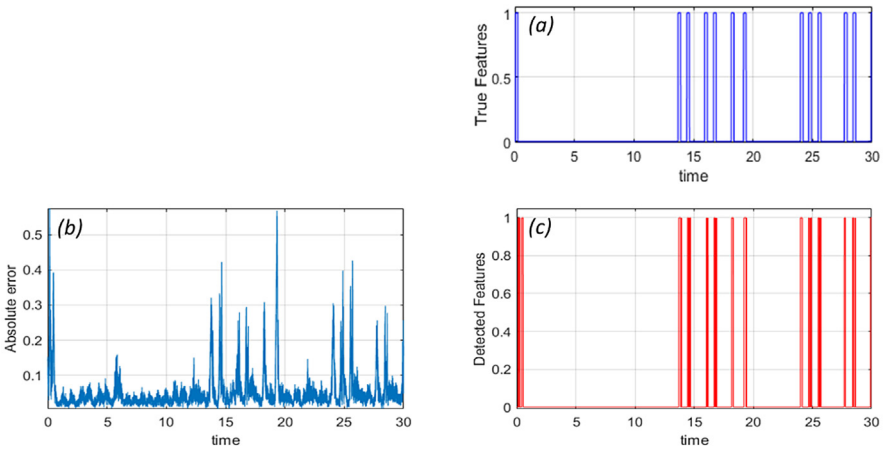

Fig. 5. Output of the feature detection example.

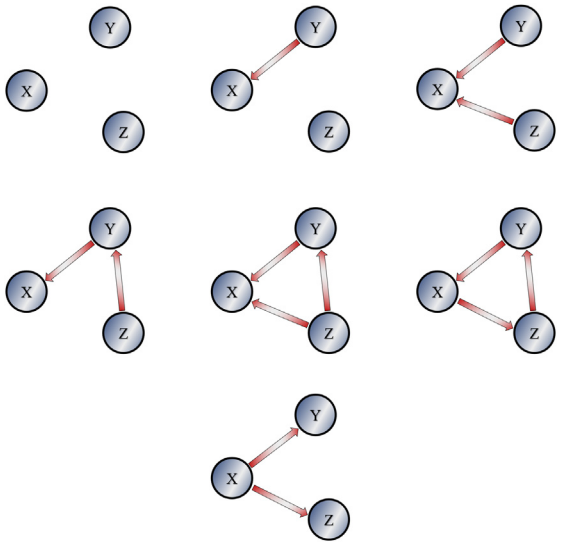

Fig. 6. Graphical representation of the seven tri-variate cases of causal influence between three systems. The circles represent the systems and the arrows the directions of the causal influences.

exerts an appreciable effect on the target. The technique has proved to work satisfactorily for all the cases of mutual influence between three systems, which are very challenging for all the main methods available on the market. The software package has been tested with synthetic data, including time series generated by systems in the chaotic regime. The generality and flexibility of the tools are expected to favour their use in many fields and particularly in the investigation of complex systems dynamics. The possibility to use neural networks to perform causality detection through embedded time-series has been also explored recently by the other researchers and the results are notable [18]. In the future, an extensive work will be undertaken to test these new approaches in several challenging cases of causality detection, in order to deeply understand their limits and advantages.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Pearl Judea, Mackenzie Dana. The book of why: the new science of cause and effect, London:Pengiun Books.
[2] Marwan N, Romano MC, Thiel M, Kurths J. Recurrence plots for the analysis of complex systems. Phys Rep 2007;438(5–6):237. http://dx.doi.org/10.1016/j.physrep.2006.11.001, Bibcode:2007PhR.438.237M.
[3] Thomas Schreiber. Measuring information transfer. Phys Rev Lett 2000;85(2):461464. http://dx.doi.org/10.1103/PhysRevLett.85.461, arXiv:nlin/0001042, Bibcode:2000PhRvL.85.461S.
[4] Granger CWJ. Investigating causal relations by econometric models and cross-spectral methods. Econometrica 1969;37(3):424–38. http://dx.doi.org/10.2307/1912791.JSTOR1912791.
[5] George Sugihara, et al. Detecting causality in complex ecosystems (PDF). Science 2013;338(6106):496–500.
[6] Krakovska A, et al. Comparison of six methods for the detection of causality in bivariate time series. Phys Rev E 2018;97:042207.
[7] Hu Yu Hen, Hwang Jenq-Neng. Handbook of neural network signal processing. Broken Sound Parkway, NW: CRC Press.
[8] Anbalagan P, Ramachandran R, Cao J, et al. Global robust synchronization of fractional order complex valued neural networks with mixed time varying delays and impulses. Int J Control Autom Syst 2019;17:509–20, https://doi.org/10.1007/s12555-017-0563-7.
[9] Rajchakit G, Chanthorn P, Niezabitowski M, Raja R, Baleanu D, Pratap A. Impulsive effects on stability and passivity analysis of memristor-based fractional-order competitive neural networks. Neurocomputing 2020;417:290–301.
[10] Chanthorn P, Rajchakit G, Humphries U, Kaewmesri P, Sriraman R, Lim CP. A delay-dividing approach to robust stability of uncertain stochastic complex-valued hopfield delayed neural networks. Symmetry 2020;12:683.
[11] Rajchakit G, Pratap A, Raja R, Cao J, Alzabut J, Huang C. Hybrid control scheme for projective lag synchronization of Riemann–Liouville sense fractional order memristive BAM neuralnetworks with mixed delays. Mathematics 2019;7:759.
[12] Zou Y, et al. Inferring indirect coupling by means of recurrences. Int J Bifurcation Chaos 2011;21(4).
[13] Murari A, et al. Plasma Phys Control Fusion 56(11) https://doi.org/10.1088/0741-3335/56/11/114007.
[14] Murari A, et al. Nucl Fusion 58(5) https://doi.org/10.1088/1741-4326/aaaf9c.
[15] Puiatti ME, et al. Plasma Phys Control Fusion 2002;44(9):1863. http://dx.doi.org/10.1088/0741-3335/44/9/305.
[16] Bellecci C, et al. Application of a CO2 dial system for infrared detection of forest fire and reduction of false alarm. Appl Phys B 2007;87(2):373–8.
[17] Lionel Barnett, Seth Anil K. Granger causality for state-space models. Phys Rev E 2015;91.4:040101.
[18] Alessandro Montalto, et al. Neural networks with non-uniform embedding and explicit validation phase to assess Granger causality. Neural Netw 2015;71:159–71.