SoftwareX 11 (2020) 100470

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

Symbolic Information Flow Measurement (SIFM): A software for measurement of information flow using symbolic analysis

Dhurata Nebiu, Hiqmet Kamberaj *

International Balkan University, Department of Computer Engineering, Makedonsko-Kosovska Brigada BB 1000, Skopje, Republic of North Macedonia

ARTICLE INFO

Article history:
Received 27 September 2019
Received in revised form 1 April 2020
Accepted 1 April 2020

Keywords:
Embedded Parameters
Symbolic transfer entropy
Symbolic local transfer entropy
Symbolic Mutual Information

ABSTRACT

Symbolic Information Flow Measurement software is used to compute the information flow between different components of a dynamical system or different dynamical systems using symbolic transfer entropy. The time series represents the time evolution trajectory of a dynamical system. We introduce a method to perform a symbolic analysis of the time series based on the coarse-graining using a machine learning approach and computation of the embedding parameters. Information flow is measured in terms of the *local* and *average* symbolic transfer entropies. We also introduce a new measure of mutual information based on the symbolic analysis.

Software metadata

Current software version	*1.0*
Permanent link to executables of this version	https://github.com/kamberaj/sifmv1/tree/master/exec/gnu
Legal Software License	*GPL 3*
Computing platform/Operating System	*Linux, OS X, Microsoft Windows, Unix-like*
Installation requirements & dependencies	*MPI Library, Fortran 90 Compilers, and Python 2.7*
If available Link to user manual - if formally published include a reference to the publication in the reference list	https://github.com/kamberaj/sifmv1/tree/master/documentation
Support email for questions	h.kamberaj@gmail.com

Code metadata

Current Code version	*V1.0*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2019_301
Legal Code License	*GPL 3*
Code Versioning system used	*Git*
Software Code Language used	*Fortran 90*
Compilation requirements, Operating environments & dependencies	*MPI, gfortran*
If available Link to developer documentation/manual	https://github.com/kamberaj/sifmv1/tree/master/documentation
Support email for questions	h.kamberaj@gmail.com

1. Introduction

Often, we need to determine the causal directions between parts of the same system or coupled systems to understand system dynamics and make an estimation of its actual physical structure. The process includes observation of the system, recording its behavior as a trajectory in phase space, or the so-called time series of signals and analyzing them. Pearson correlation coefficient [1] does not imply causality, and also, it detects only the linear correlations. Also, it is not sensitive to fluctuations in perpendicular directions, but only to those that distribute along with co-linear directions [2]. Granger causality [3] determines the direction of interaction in terms of the contribution of a random variable X in predicting another random variable Y, and many variations of this concept are suggested.

Information theory measure of transfer entropy quantifies the statistical coherence between two processes that evolve in time.

* Corresponding author.
E-mail address: h.kamberaj@gmail.com (H. Kamberaj).

https://doi.org/10.1016/j.softx.2020.100470

Transfer entropy (TE) is introduced by Schreiber [4] as the deviation from the independence of the state transition of an information destination X from the previous state of an information source Y. TE is an asymmetric measure of the information flow between dynamical variables that represent either different components of a dynamical system or different dynamical systems. TE can distinguish between the dynamical macro-variable characterizing the actual dynamic behavior of the physical system (*source*), and the other macro-variable that responds to the changes (*sink*) [4].

Computing TE is a challenging problem due to its computational complexity. Different numerical recipes are suggested [5]. TE is already used for time series analysis in various fields: clinical electroencephalography [4,6,7], financial data [8], and biophysics [2]. Many algorithms used to calculate TE are subject to statistical errors, and reliable estimations of the transfer entropy are data intensive [2].

This study presents a computer program, written in Fortran 90 programming language, used to perform symbolic information flow measurements.

2. Algorithm design using symbolic analysis

2.1. Analyzing collective variables using machine learning

Dynamical variables, characterizing the components of a system or different dynamical systems, represent the collective degrees of freedom of a system. They are often determined using the principal components analysis (PCA) [9] as for biomolecules [10, 11]. We introduce a new algorithm, which is an improvement of the auto-encoder machine learning (ML) approach [12] for determining the collective variables from higher dimensionality data. ML predicts the properties of a system using decision-making algorithms, based on some predefined features characterizing these properties. Different ML methods are used to predict missing data and discover new patterns during a data mining process [13]. Artificial neural network methods consider a large training dataset to construct a system, which is made up of rules for recognizing the patterns within the training data by an ML process [14].

Transpose vector $\boldsymbol{Q}'$ represents T time frames of dynamical variables: $\boldsymbol{Q}' = (\boldsymbol{q}(0), \boldsymbol{q}(1), \ldots, \boldsymbol{q}(T-1))$, where $\boldsymbol{q}(t)$ represents a configuration of the system (or its components) of g degrees of freedom at t: $\boldsymbol{q}(t) = \left(q_1(t), q_2(t), \ldots, q_g(t)\right)$. That forms a Markov chain of the states of a stationary stochastic process visited by the system. To find a reduced g' dimensional space ($g' < g$), which compresses the data, an encoding function is determined [12]:

$$E: R^g \rightarrow R^{g'} \tag{1}$$

and a decoding function as:

$$D: R^{g'} \rightarrow R^g \tag{2}$$

E provides a non-linear mapping using the Bootstrapping Swarm Artificial Neural Network [14] of the Cartesian coordinates $\boldsymbol{q}(t)$ as:

$$\boldsymbol{X}(t) = E(\boldsymbol{q}(t)) \tag{3}$$

$\boldsymbol{X}(t)$ is an g' dimensional vector in the essential subspace of slow collective variables. Similarly, using the non-linear mapping D, we obtain an approximate time-lagged signal, $\widetilde{\boldsymbol{q}}(t+\tau)$:

$$\widetilde{\boldsymbol{q}}(t+\tau) = D(\boldsymbol{X}(t)) \tag{4}$$

τ is the time-lag of the input signal $\boldsymbol{q}(t)$, and the approach is called time-lagged auto-encoder. For $\tau = 0$, the approach represents the standard auto-encoder method. Both the input and output signal of the encoder–decoder non-linear neural network is the trajectory $\boldsymbol{q}(t)$ in the Cartesian space and the output signal of the encoder, which is the input signal for the decoder, represent the slow collective variables $\boldsymbol{X}(t)$. The input signals are reconstructed using the Cartesian space vectors [12]:

$$\begin{aligned}
\boldsymbol{x}(t) &= \boldsymbol{q}(t) - \frac{1}{T-\tau}\sum_{k=0}^{T-1-\tau}\boldsymbol{q}(k) \\
\boldsymbol{y}(t) &= \boldsymbol{q}(t+\tau) - \frac{1}{T-\tau}\sum_{k=0}^{T-1-\tau}\boldsymbol{q}(k+\tau)
\end{aligned} \tag{5}$$

Covariance matrices are constructed as:

$$\begin{aligned}
\boldsymbol{C}_1 &= \frac{1}{T-\tau}\sum_{t=0}^{T-1-\tau}\boldsymbol{x}(t)\boldsymbol{x}'(t) \\
\boldsymbol{C}_2 &= \frac{1}{T-\tau}\sum_{t=0}^{T-1-\tau}\boldsymbol{y}(t)\boldsymbol{y}'(t)
\end{aligned} \tag{6}$$

Both signals $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ are whitened as:

$$\begin{aligned}
\widehat{\boldsymbol{x}}(t) &= \boldsymbol{C}_1^{-\frac{1}{2}}\boldsymbol{x}(t) \\
\widehat{\boldsymbol{y}}(t) &= \boldsymbol{C}_2^{-\frac{1}{2}}\boldsymbol{y}(t)
\end{aligned} \tag{7}$$

These two signals are the input and output, respectively, of the encoder–decoder algorithm, which defines the non-linear functions E and D, such that the reconstructed error is minimum:

$$\hat{S} = \min_{\mathrm{E,\,D}}\sum_{t=0}^{T-1-\tau}\left\|\widehat{\boldsymbol{y}}(t) - D\left(E\left(\widehat{\boldsymbol{x}}(t)\right)\right)\right\|^2 \tag{8}$$

Input signals $\boldsymbol{q}(t)$ could characterize the dynamics of the entire dynamical systems, such as in measuring the information flow in signal pathways. Besides, $\boldsymbol{q}(t)$ could characterize the dynamics of a component of the dynamical system, such as in measuring the information flow between different parts of the same system. It may represent the atomic coordinates of each amino acid in a protein structure in determining the information flow between different amino acids of a protein. We call this process *structure coarse-graining*.

2.2. Symbolic analysis

The symbolic analysis approach is implemented based on a time coarse-graining of the time series into symbols [2,15–18]. We use the symbolization technique proposed in Ref. [2], found to be computationally very robust. This approach creates a symbolic sequence for each time series $(X_0, X_1, \ldots, X_{N-1})$, namely $(\hat{X}_0, \hat{X}_1, \ldots, \hat{X}_{N-1})$. This process is also called *time coarse-graining* [2] in which all information concerning the dynamics of series is encoded using partitioning of phase space into *symbols*. The time-series $(X_0, X_1, \ldots, X_{N-1})$ is converted into a symbolic sequence using the rule [2]

$$\hat{X}_j = \hat{S}_k, \quad \text{if } X_k^c < X_j < X_{k+1}^c \tag{9}$$

where $\left(X_0^c, X_1^c, \ldots, X_D^c\right)$ is a given set of $D+1$ critical points, and $(\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{D-1})$ is a set of D symbols: $0, 1, 2, \ldots, D-1$. D is such that the Kraft inequality is satisfied [19]: $\sum_k D^{-m_k} \leq 1$, where the sum runs over all state vectors, and m_k is the length of each state vector, which for a single time series is considered to be equal for every state vector. Then, a new symbolic state vector, representing a subset of numbers from 0 to $D-1$, is generated as

$$\hat{X}_k^{\mu} = \left(\hat{S}_1^{(k)}, \hat{S}_2^{(k)}, \ldots, \hat{S}_m^{(k)}\right)^T \tag{10}$$

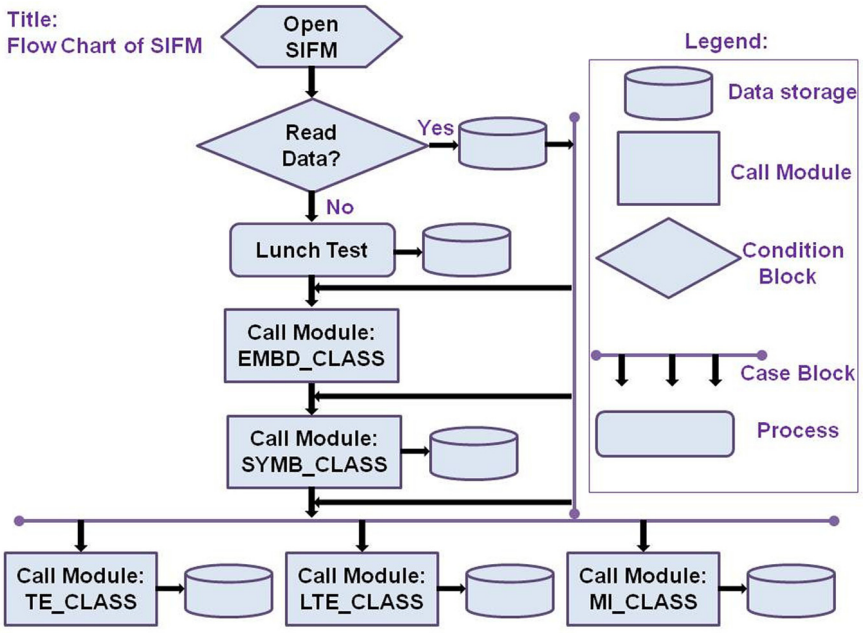

Fig. 1. A flowchart diagram of the SIFM software.

Concatenation of the symbols of a sequence of length m yields the word W_k:

$$W_k = (\hat{X}_{k-(m-1)\tau} \ldots \hat{X}_{k-\tau}\hat{X}_k) \tag{11}$$

A particular sequence of symbols $\left(\hat{X}_0, \hat{X}_1, \ldots, \hat{X}_{N-1}\right)$ is uniquely characterized by the words W_k for $k = k_0, \ldots, N-1$ [2]. The probability of finding a particular value of W_k, is calculated from the simulation data and used to compute the Shannon entropy as a sum over all symbolic state vectors represented by the words W_k:

$$H\left(\hat{X}^{\mu}\right) = -\sum_k p\left(W_k\right) \log p(W_k) \tag{12}$$

Function representing the map from time series $(X_0, X_1, \ldots, X_{N-1})$ to symbolic sequence $\left(\hat{X}_0, \hat{X}_1, \ldots, \hat{X}_{N-1}\right)$ is injective, and hence $H\left(\hat{X}^{\mu}\right)$ and $H\left(X^{\mu}\right)$ coincides [20]. We obtained the critical points $\left\{X_d^c\right\}_{d=0}^{D}$ for a particular series by maximizing the entropy for all possible partitions [2]. We optimize critical points by maximizing the Shannon entropy using a Monte Carlo approach [2]. The joint Shannon information entropy of two discrete symbolic processes $\{\hat{X}_0, \hat{X}_1, \ldots, \hat{X}_{N-1}\}$ and $\{\hat{Y}_0, \hat{Y}_1, \ldots, \hat{Y}_{N-1}\}$ is as

$$H\left(\hat{X}^{\mu_x}, \hat{Y}^{\mu_y}\right) = -\sum_k p\left(\overline{W}_k\right) \log p(\overline{W}_k) \tag{13}$$

$\overline{W}_k$ is the concatenation of two words, $W_k^{(x)}$ and $W_k^{(y)}$, representing the processes X and Y, respectively [2]. With this coarse-graining of the time series, the *symbolic transfer entropy* is as [2]:

$$\begin{aligned}\widehat{T}_{\hat{Y}\to\hat{X}} = &\left(H\left(\hat{X}^{\mu_x+1}\right) - H(\hat{X}^{\mu_x})\right) \\ &- \left(H\left(\hat{X}^{\mu_x+1}, \hat{Y}^{\mu_y}\right) - H\left(\hat{X}^{\mu_x}, \hat{Y}^{\mu_y}\right)\right)\end{aligned} \tag{14}$$

Similarly, the *symbolic local transfer entropy* is given by:

$$\begin{aligned}\hat{t}_{\hat{Y}\to\hat{X}}(k) = &\left(\log p\left(\hat{X}_k^{\mu_x+1}, \hat{Y}_k^{\mu_y}\right) - \log p\left(\hat{X}_k^{\mu_x}, \hat{Y}_k^{\mu_y}\right)\right) \\ &- \left(\log p\left(\hat{X}_k^{\mu_x+1}\right) - \log p(\hat{X}_k^{\mu_x})\right)\end{aligned} \tag{15}$$

The symbolic local transfer entropy is such that

$$\widehat{T}_{\hat{Y}\to\hat{X}} = \sum_k p\left(\hat{X}_{k+\delta}, \hat{X}_k^{\mu_x}, \hat{Y}_k^{\mu_y}\right) \hat{t}_{\hat{Y}\to\hat{X}}(k) \tag{16}$$

Sum runs over all states, and δ represents a time shift in the history dependence. Also, *symbolic mutual information* $\hat{I}(X;Y)$ given as:

$$\hat{I}(X;Y) = H\left(\hat{X}^{\mu_x}\right) + H\left(\hat{Y}^{\mu_y}\right) - H(\hat{X}^{\mu_x}, \hat{Y}^{\mu_y}) \tag{17}$$

2.3. Algorithm for optimization of embedded parameters

Proper values of embedding parameters m and τ define a smooth discrete-time process able to reconstruct the underlying dynamics [2,21–23], and hence for an appropriate characterization of the structure of the time series [18,24–28] of the original dynamics. To estimate the embedding parameters m and τ, we implemented the algorithm presented in Ref. [2]. Here, τ is determined as the first minimum of the mutual information function versus the time lag [2]:

$$I\left(X; X_{+\tau}\right) = H\left(X\right) + H\left(X_{+\tau}\right) - H(X, X_{+\tau}) \tag{18}$$

Space dimension m is determined from a separate procedure using the false nearest neighbors. For that, given a state vector, $X_k^{\mu} = (x_{k-(m-1)\tau}, x_{k-(m-1)\tau+\tau}, \ldots, x_{k-\tau}, x_k)$, the nearest neighbor is

$$X_k^{\mu,NN} = \left(x_{k-(m-1)\tau}^{NN}, x_{k-(m-1)\tau+\tau}^{NN}, \ldots, x_{k-\tau}^{NN} x_k^{NN}\right) \tag{19}$$

Euclidean distance, between these two points in m-dimensional space, is given by

$$R_i^m = \left(\sum_{k=0}^{m-1} \left(x_{i+k\tau} - x_{i+k\tau}^{NN}\right)^2\right)^{1/2} \tag{20}$$

Distance between these two points in $(m+1)$-dimensional space is,

$$R_i^{m+1} = \left(\left(R_i^m\right)^2 + \left(x_{i+m\tau} - x_{i+m\tau}^{NN}\right)^2\right)^{1/2} \tag{21}$$

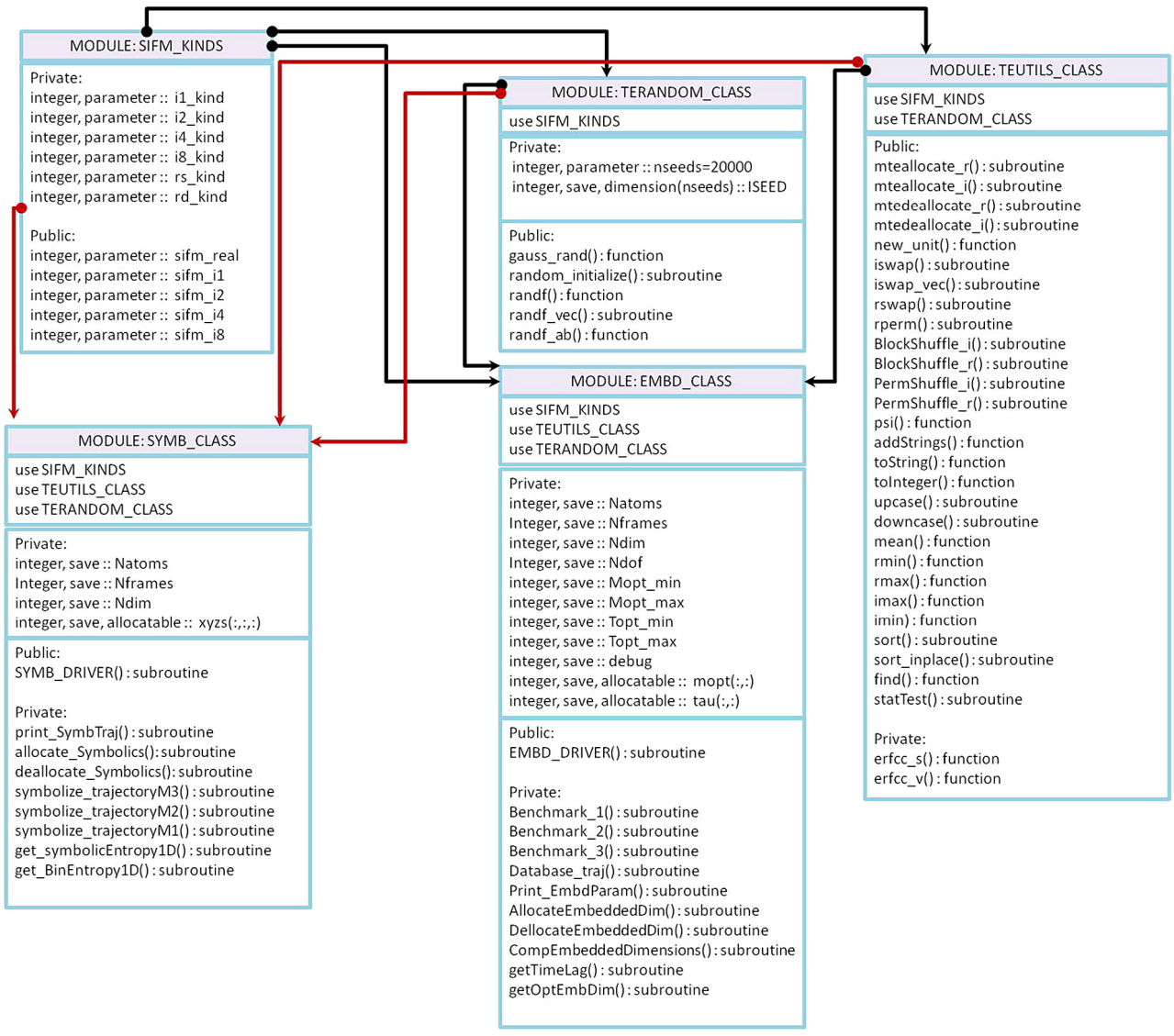

Fig. 2. UML diagram of the module EMBD_CLASS and SYMB_CLASS.

This distance is normalized against the length in m-dimensional space:

$$\gamma_i^m = \left(\frac{\left(R_i^{m+1}\right)^2 - \left(R_i^m\right)^2}{\left(R_i^m\right)^2} \right)^{1/2} = \frac{|x_{i+m\tau} - x_{i+m\tau}^{NN}|}{R_i^m} \tag{22}$$

γ_i^m is compared to a threshold value R_{tol}, which is determined a priori and recommended to be 15 [28]. We tested the algorithm for different values of R_{tol} in the range from 5 to 25 [2]. If γ_i^m exceeds R_{tol}, then X_i^{NN} is the false nearest neighbor of X_i in the m-dimensional space and f_{FNN}, the frequency of the false nearest neighbors, is increased by one. m increases until f_{FNN} approaches zero [2].

3. Software framework

3.1. Software architecture

A flowchart diagram of the software, shown in Fig. 1, shows the five modules, namely, EMBD_CLASS, SYMB_CLASS, TE_CLASS, LTE_CLASS, and MI_CLASS.

Embedded Parameters Module (*EMBD_CLASS*) contains the methods for computation of the embedded parameters. Fig. 2 shows a UML diagram. It inherits the functions from three auxiliary modules: SIFM_CLASS, TEUTILS_CLASS, and TERANDOM_CLASS.

Symbolize Trajectory Module (*SYMB_CLASS*) contains the methods used for creating the symbolic trajectories. Fig. 2 presents a UML diagram showing the auxiliary modules inherited by the SYMB_CLASS.

Symbolic Transfer Entropy Module (*TE_CLASS*) contains the procedures for calculations of the symbolic transfer entropy (Fig. 4). Fig. 3 presents the UML diagram of TE_CLASS, where NAME_CLASS is TE_CLASS. Besides, we show the auxiliary modules.

Symbolic Local Transfer Entropy Module (*LTE_CLASS*) contains the procedures for calculations of the symbolic local transfer entropy (see Fig. 4). In Fig. 3, we show the UML diagram of the LTE_CLASS module (with NAME_CLASS as LTE_CLASS), and other modules called by LTE_CLASS.

Symbolic Mutual Information Module (*MI_CLASS*) contains the procedures for calculations of the symbolic mutual information (Fig. 4); Fig. 3 shows a UML diagram of MI_CLASS in which

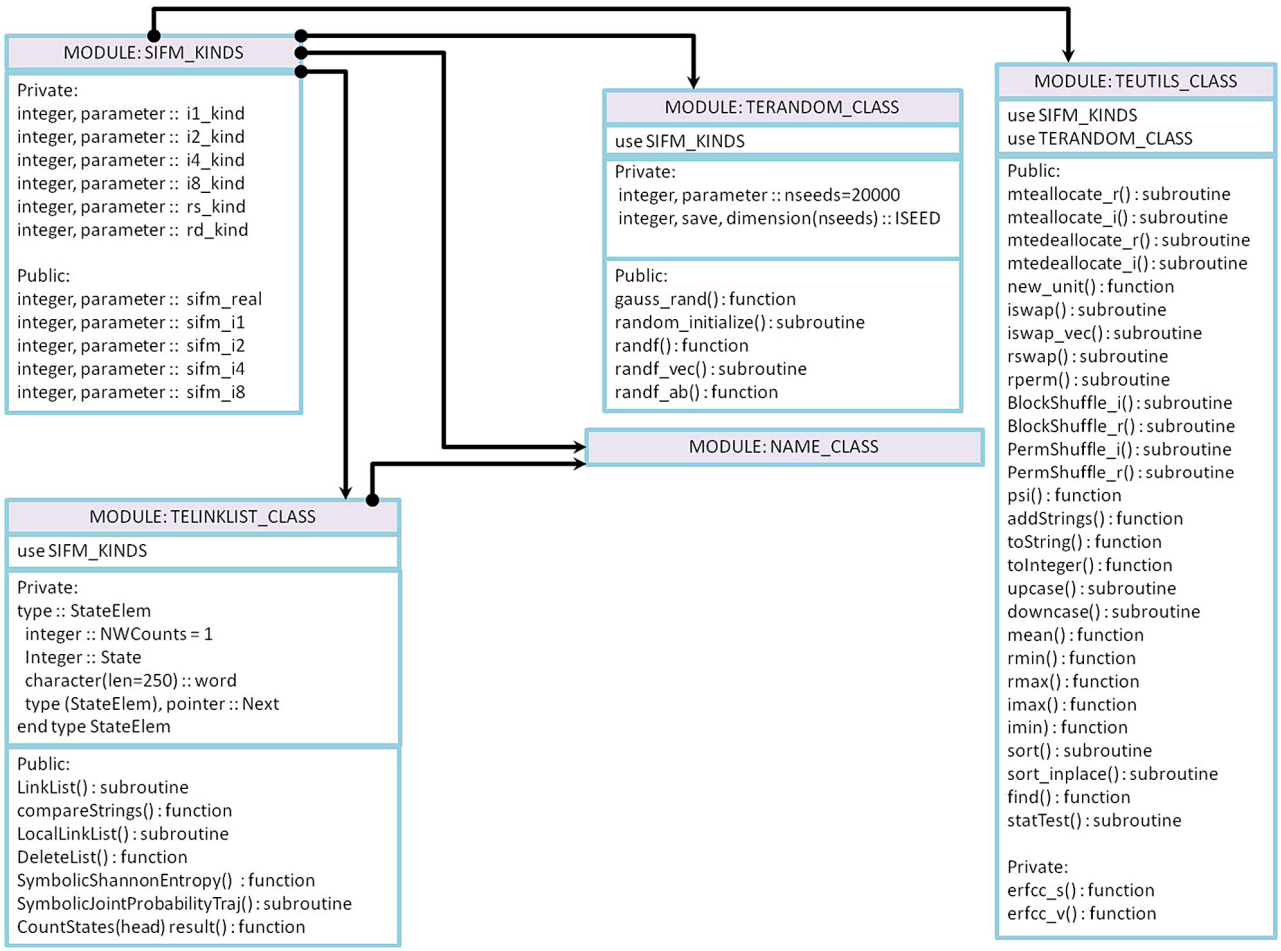

Fig. 3. UML diagram of an arbitrary module NAME_CLASS, which can be TE_CLASS, LTE_CLASS, or MI_CLASS.

NAME_CLASS represents MI_CLASS. We also indicate the auxiliary modules called by MI_CLASS.

Auxiliary Class Modules: There are four auxiliary modules: SIFM_KINDS, TEUTILS_CLASS, TERANDOM_CLASS, and TE_LINKLIST_CLASS (see Fig. 3).

3.2. Software functionalities and implementation

Portability SIFM is written in Fortran 90 object-oriented style, making it very portable in all operating systems.

Computational Precision: The precision of the computation can be adjusted by the user using KIND variables, described in SIFM_KINDS, for example, by adjusting the variables *sifm_real*, *sifm_i1*, *sifm_i2*, *sifm_i4*, or *sifm_i8*. Besides, SIFM uses fast intrinsic routines of FORTRAN for manipulating the operations with strings or converting integer symbols into string symbols.

Memory Management: Software is written in a modular fashion, making it faster to compile and better memory management. All the dynamical arrays are declared as either allocatable arrays or pointers, providing efficient management of the memory.

Data Management: Each module produces its data output, used either for further analysis or by other modules as an input. That is, in particular, important when the program is interrupted or crashed due to errors, and then a new restart is set up from the last point. SIFM supports different kinds of input data formats, including binary files, reducing the size of those files. SIFM endorses the majority of the trajectory file formats produced by other software, making it applicable to many other fields.

Parallelization: SIFM is highly parallelized using Message Passing Protocols. In this version, we parallelized the computation of symbolic trajectories using the Monte Carlo and the calculation of symbolic TE, LTE, and MI. That is in particularly useful when performing symbolic analysis of more than one pair of time series, let say of n pairs, $(X_1(t), Y_1(t)), (X_2(t), Y_2(t)), \ldots, (X_n(t), Y_n(t))$. Since the computations of pairs are independent, we can distribute these computations among P processors. Every processor is performing m calculations, $m = P/n$. P is such that P/n is an integer to ensure the equal workload among the processors. Fig. 5(A) shows the average CPU time (in seconds) for the computation of embedded parameters for each degree of freedom as a function of P and length of the time series. The averages carry out over different dynamical systems. It shows that CPU time decreases proportionally with the number of processors, and as expected, it also increases with the length of the time series. Typically, it takes 2.3 s to compute both m and τ for a time series of size 20 000 frames using 8 CPUs.

Fig. 5(B–C) presents the speedup of the parallel computations for symbolic transfer entropy and symbolic mutual information versus the number of processors. For the sake of comparisons, the ideal expected speed up is also shown. We found that up to four processors, the estimated speedup is the same as the ideal one. Further increase in the number of processors resulted in a deviation from this linearity due to overhead time on the

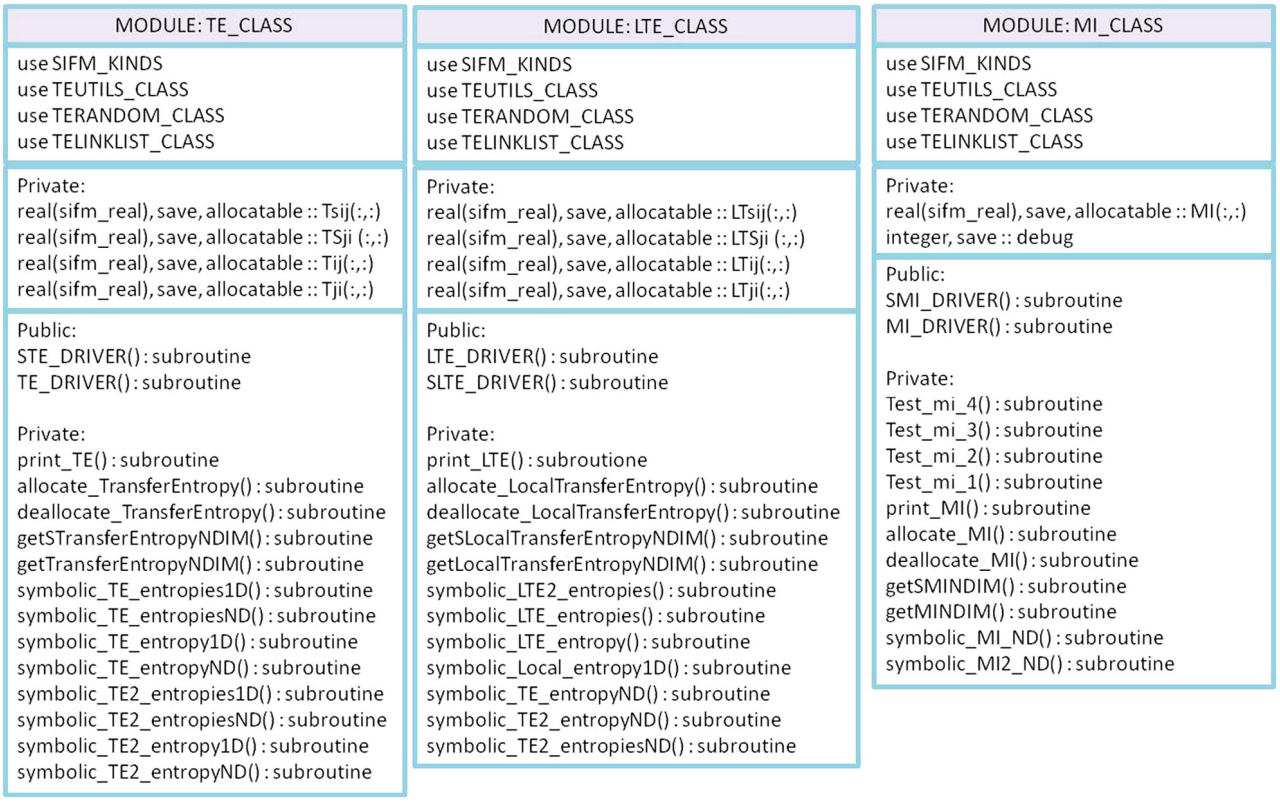

Fig. 4. UML diagram of the modules TE_CLASS, LTE_CLASS, and MI_CLASS.

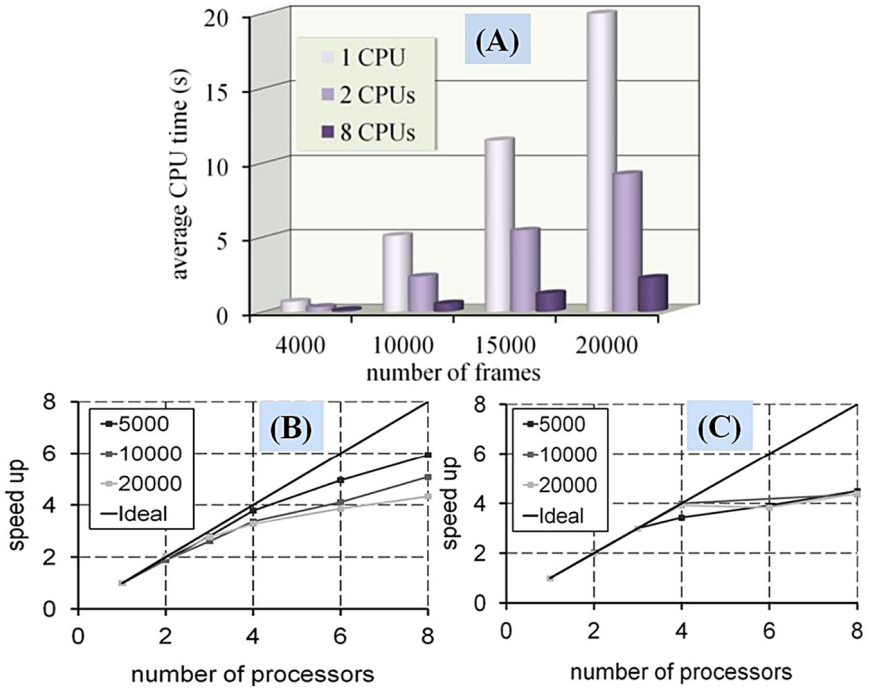

Fig. 5. (A) Average CPU time (in seconds) for the computation of embedded parameters (m, τ) per time series versus the length of the time series and number of processors. Speed up of (B) Symbolic Transfer Entropy and (C) Symbolic Mutual Information calculations versus the number of processors for the different lengths of time series.

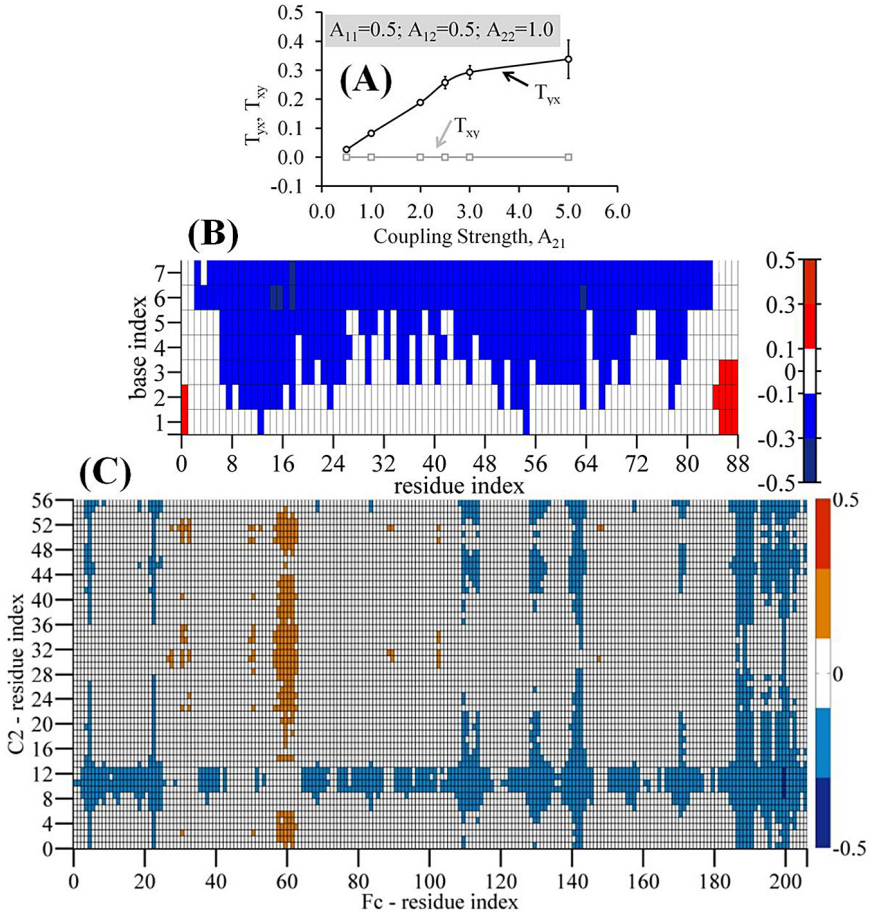

Fig. 6. (A) Symbolic transfer entropies, T_{xy} and T_{yx} versus the coupling strength A_{21} for fixed choices of other parameters. Directional Symbolic Transfer Entropy for (B) protein–RNA complex system, and (C) C2–Fc complex system at the interface.

master node. Note that these computations are performed on the i7 computer architecture.

4. Illustrative examples

Benchmark 1 is two coupled noisy time series as:

$$\begin{cases} x(t) = A_{11}x(t-\delta) + A_{12}\mathcal{N}(0,1) \\ y(t) = A_{21}x(t-\delta) + A_{22}\mathcal{N}(0,1) \end{cases} \tag{23}$$

In Eq. (23), A_{11}, A_{12}, A_{21}, and A_{22} are constants, and $\mathcal{N}(0,1)$ is a random number following the normal distribution with mean zero and variance one. A_{21} is the strength of coupling between X and Y, and A_{22} the strength of the external noise on Y. δ is one. Fig. 6(A) shows the symbolic transfer entropy versus the coupling strength A_{21}. Fig. 6(A) also indicates the values of other parameters.

Benchmark 2 corresponds to the C2-Fc complex biomolecule, where C2 is a fragment of protein G, and Fc is a domain of human IgG protein. C2 fragment has 56 amino acids, and Fc has 206 amino acids. We performed molecular dynamics simulations for 30 ns. The first ten ns are omitted from the analysis, and only the last 20 ns are used for calculations. We printed out the configurations every two ps, thus, in total, 10 000 snapshots were used for analyzes. Fig. 6(C) shows the directional symbolic transfer entropies, $D_{i\to j}$, as a color map between C2 and Fc. The directional symbolic transfer entropy between time series X and Y is calculated as: $D_{X\to Y} = \widehat{T}_{X\to Y} - \widehat{T}_{Y\to X}$. The scale of the values is shown in color bars plotted next to the graph. Our results identify the driving (source), characterized by a positive value of $D_{i\to j}$ and responding (sink) residues, characterized by a negative value of $D_{i\to j}$. That is, if $D_{i\to j} > 0$, then residue i drives j, otherwise j drives i.

The next test system is a complex biomolecular system, representing Protein–RNA interactions. Protein is composed of 88 amino acids, and RNA is composed of 6 bases. We performed 15 ns of molecular dynamics simulation of the complex. The first five ns is considered as equilibration, and only the last ten ns is used for calculations. We printed out the configurations every two ps, and thus, 5000 snapshots were saved. Fig. 6(B) presents the directional symbolic transfer entropy between protein and RNA. We can identify the residues driving the fluctuation motions and those responding to these fluctuations. We identify different clusters of residues acting as the source of the changes: the first cluster includes the residues with the index from 1 to 2, and the second cluster consists of the residues with the index from 7 to 12. There also exist three other small groups driving the motions of the base 1 and 2, such as the cluster made up of the residues between 36 and 40, between 65 and 69, and this including the residues between 75 and 87.

5. Impact

Impact includes the use of SIFM as a toolkit for measurement of information flow on the dynamical systems, for instance, between coupled dynamical systems (Benchmark 1) or between various components of the same dynamical system (Benchmark 2). The SIFM uses the ML tool to encode the full dynamics of a system to a lower-dimensional space, and an embedded parameters technique to reconstruct the dynamics of the original system,

making the toolkit robust from the computation viewpoint. Furthermore, the use of a symbolic data analysis approach provides a robust and accurate estimate of information flow measurements.

Also, we believe that SIFM can be used to associate transfer entropy with energy transfer at the interfaces of complex biomolecular systems, as in Refs. [14,29]. Thus, we can be able to characterize the thermal conductance at the interfaces relevant to biological processes enabling *in-silico* rational peptide/protein engineering technologies for biomolecules interfaces.

6. Conclusions

We presented a new computer program for computation of symbolic analysis of time series representing random processes of dynamical systems. Besides, this program can compute symbolic transfer entropy, mutual information, and local transfer entropy.

Some of the functionalities of this computer program added during the implementation include portability, adjustable computational precision, efficient memory management, useful and practical data management system, and parallelization of computations using MPI protocols.

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] Rice J. Mathematical statistics and data analysis. 2nd ed.. Belmont: Duxbury Press; 1995.
[2] Kamberaj H, van der Vaart A. Extracting the causality of correlated motions from molecular. Biophys J 2009;97:1747–55.
[3] Granger J. Investigating causal relations by econometric models and cross-spectral methods. Acta Phys Polon B 1969;37:424–38.
[4] Schreiber T. Measuring information transfer. Phys Rev Lett 2000;85:461–4.
[5] Gencaga D, Knuth KH, Rossow WB. A recipe for the estimation of information flow in a dynamical system. Entropy 2015;17:438–70.
[6] Gourévitch B, Eggermont J. Evaluating information transfer between auditory cortical neurons. J Neurophysiol 2007;97:2533–43.
[7] Staniek M, Lehnertz K. Symbolic transfer entropy. Phys Rev Lett 2008;100. 158101.
[8] Marschinski R, Kantz H. Analyzing the information flow between financial time series. Eur Phys J B 2002;30:275–81.
[9] Karhunen K. Uber lineare methoden in der wahrscheinlichkeitsrechnung. Ann Acad Sci Fenn A1 1947;37:1–79.
[10] Balsera M, Wriggers W, Oono Y, Schulten K. Principal component analysis and long time protein dynamics. J Phys Chem 1996;100(7):2567–72.
[11] Kamberaj H. A theoretical model for the collective motion of proteins by means of principal component analysis. Open Phys 2011;9(1):96–109.
[12] Wehmeyer C, Noe F. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. J Chem Phys 2018;148:241703–9.
[13] McCulloch W, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 1943;5(4):115–33.
[14] Kamberaj H. Molecular dynamics simulations in statistical physics: Theory and applications. 1st ed.. Springer; 2020.
[15] Bossomaier T, Barnett L, Harré M, Lizier JT. An introduction to transfer entropy information flow in complex systems. Springer; 2016.
[16] Grassberger P, Procaccia I. Measuring the strangeness of strange attractors. Physica D 1983;9(1–2):189–208.
[17] Lizier Joseph T, Prokopenko Mikhail, Zomaya Albert Y. Local information transfer as a spatiotemporal filter for complex systems. Phys Rev E 2008;77(026110).
[18] Sauer T, Yorke J, Casdagli M. Embedology. J Stat Phys 1991;65:579–616.
[19] Cover Thomas M, Thomas Joy A. Elements of information theory. New Jersey, Hoboken: John Wiley & Sons, Inc.; 2006.
[20] Bonanno C, Mega M. Toward a dynamical model for prime numbers. Chaos Solitons Fractals 2004;20:107–18.
[21] Lizier J. JIDT: An information-theoretic toolkit for studying the dynamics of complex systems. Front Robot AI 2014;1:11.
[22] Moore D, Valentini G, Walker S, Levin M. Inform: A toolkit for information-theoretic analysis of complex systems. In: IEEE symposium series on computational intelligence (SSCI). 2017.
[23] Moore D, Valentini GWSI, Levin M. Inform: Efficient information-theoretic analysis of collective behaviors. Front Robot AI 2018;5:17.
[24] Cellucci C, Albano A, Rapp P. Comparative study of embedding methods. Phys Rev E 2003;67:066210–3.
[25] Kennel M, Brown R, Abarbanel H. Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys Rev A 1992;45:3403–11.
[26] Abarbanel H, Kennel M. Local false nearest neighbors and dynamical dimensions from observed chaotic data. Phys Rev E 1993;47:3057–68.
[27] Noakes L. The takens embedding theorem. Internat J Bifur Chaos Appl Sci Engrg 1991;1:867–72.
[28] Cellucci C, Albano A, Rapp P. Statistical validation of mutual information calculations: Comparison of alternative numerical algorithms. Phys Rev E 2005;71(066208-14).
[29] Leitner D, Pandey H, Reid K. Energy transport across interfaces in biomolecular systems. J Phys Chem B 2019;123:9507–24.