SPIKY: A graphical user interface for monitoring spike train synchrony

Nebojsa Bozanic, Thomas Kreuz

Institute for complex systems, CNR, Sesto Fiorentino, Italy

Abstract

XXXXX Preliminary, to be edited in the end XXXXX Recently, the SPIKE-distance has been proposed as a measure of spike train synchrony which is both parameter-free and time-scale independent. Since it relies on instantaneous estimates of spike train dissimilarity, it is also time-resolved which makes it possible to track changes in instantaneous clustering, i.e., time-localized patterns of (dis)similarity among multiple spike trains. Further features include selective and triggered temporal averaging as well as the instantaneous comparison of spike train groups. Besides the regular SPIKE-distance, there also exists a causal variant which is defined such that the instantaneous values of dissimilarity rely on past information only so that time-resolved spike train synchrony can be estimated in real-time. Finally, here we introduce the future SPIKE-distance which can be used in triggered temporal averaging in order to evaluate the effect of certain spikes or of certain stimuli features on future spiking. In the first part of this report we address some of the computational aspects in the calculation and implementation of the SPIKE-distance while in the second part we present SPIKY, a graphical user interface which facilitates the application of the SPIKE-distance and all its variants to both simulated and real data.

1. Introduction

XXXXX Preliminary, to be edited in the end XXXXX

A wide variety of approaches to quantify the dissimilarity, or distance, between two spike trains has been suggested. Among these is the metric introduced in **?**, which evaluates the cost needed to transform one spike train into the other, using only certain elementary steps. Another metric proposed in **?**, measures the Euclidean distance between the two spike trains after convolution of the spikes with an exponential function. Both methods involve one parameter that sets the time scale. In contrast, two more recent bivariate approaches, the ISI- and the SPIKE-distance are time scale independent and self-adaptive (**??**). These two measures are complementary: The ISI-distance relies on the relative length of simultaneous interspike intervals and is thus well-designed to quantify similarities in the neurons' firing-rate profiles (**??**). The SPIKE-distance is based on differences between the spike times of the two spike trains and is therefore ideally suited to track synchrony that is mediated by spike timing (**??**).

Reliability includes clustering

XXXXX Computational aspects XXXXX

Email addresses: `strance@gmail.com` (Nebojsa Bozanic), `thomas.kreuz@cnr.it` (Thomas Kreuz).

2. Measures

2.1. *The SPIKE-distance*

The SPIKE-distance (see **?** for the original proposal and **?** for the definite version presented here) is extracted from differences between the spike times of the two spike trains. It relies on instantaneous values in the sense that in a first step the two sequences of discrete spike times are transformed into a continuous dissimilarity profiles $S(t)$. This dissimilarity profile is based on three piecewise constant quantities which for each neuron $n = 1, 2$ are assigned to every time instant between 0 and T (see Fig. 1). These are the time of the preceding spike

$$t_{\mathrm{P}}^{x}(t) = \max(t_i^n | t_i^n \leq t) \quad t_1^n \leq t \leq t_{M_n}^n, \tag{1}$$

the time of the following spike

$$t_{\mathrm{F}}^{n}(t) = \min(t_i^n | t_i^n > t) \quad t_1^n \leq t \leq t_{M_n}^n, \tag{2}$$

as well as the interspike interval

$$x_{\mathrm{ISI}}^{n}(t) = t_{\mathrm{F}}^{n}(t) - t_{\mathrm{P}}^{n}(t). \tag{3}$$

The ambiguity regarding the definition of the very first and the very last interspike interval is resolved by placing for each spike train auxiliary leading spikes at time $t = 0$ and auxiliary trailing spikes at time $t = T$. From these three quantities the dissimilarity profile is calculated in two steps: First for each spike the distance to the nearest spike in the

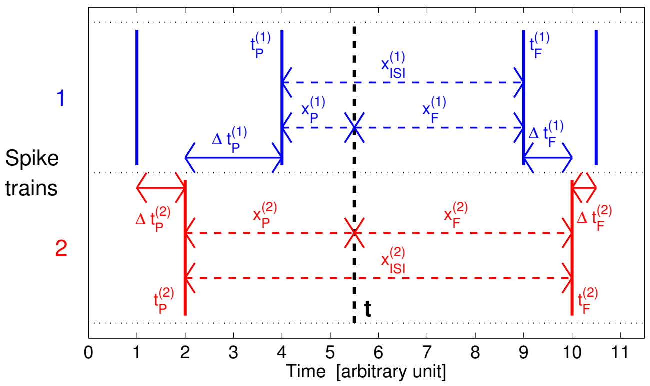

Fig. 1. SPIKE-distance. Illustration of the local quantities needed to define the dissimilarity profile $S(t)$ for an arbitrary time instant t.

other spike train is calculated, then for each time instant the relevant spike time differences are selected, weighted, and normalized. Here 'relevant' means local; each time instant is uniquely surrounded by four corner spikes: the preceding spike from the first spike train $t_{\rm P}^{(1)}$, the following spike from the first spike train $t_{\rm F}^{(1)}$, the preceding spike from the second spike train $t_{\rm P}^{(2)}$, and, finally, the following spike from the second spike train $t_{\rm F}^{(2)}$. Each of these corner spikes can be identified with a spike time difference, for example, for the previous spike of the first spike train

$$\Delta t_{\rm P}^{(1)} = \min_i(|t_{\rm P}^{(1)} - t_i^{(2)}|) \tag{4}$$

and analogously for $t_{\rm F}^{(1)}$, $t_{\rm P}^{(2)}$, and $t_{\rm F}^{(2)}$ (see Fig. 1). For each spike train separately a locally weighted average is employed such that the differences for the closer spike dominate; the weighting factors depend on

$$x_{\rm P}^{(n)}(t) = t - t_{\rm P}^{(n)}(t) \tag{5}$$

and

$$x_{\rm F}^{(n)}(t) = t_{\rm F}^{(n)}(t) - t, \tag{6}$$

the intervals to the previous and the following spikes for each neuron $n = 1, 2$. The local weighting for the spike time differences of the first spike train reads

$$S_1(t) = \frac{\Delta t_{\rm P}^{(1)} x_{\rm F}^{(1)} + \Delta t_{\rm F}^{(1)} x_{\rm P}^{(1)}}{x_{\rm ISI}^{(1)}} \tag{7}$$

and analogously $S_2(t)$ is obtained for the second spike train. Averaging over the two spike train contributions and normalizing by the mean interspike interval yields

$$S''(t) = \frac{S_1(t) + S_2(t)}{2\langle x_{\rm ISI}^{(n)} \rangle_n}. \tag{8}$$

This quantity weights the spike time differences for each spike train according to the relative distance of the corner spike from the time instant under investigation. This way relative distances within each spike train are taken care of, while relative distances between spike trains are not. In order to get these ratios straight and to account for differences in firing rate, in a last step the two contributions from the two spike trains are locally weighted by their instantaneous interspike intervals. This leads to the definition of the dissimilarity profile

$$S(t) = \frac{S_1(t) x_{\rm ISI}^{(2)} + S_2(t) x_{\rm ISI}^{(1)}}{2\langle x_{\rm ISI}^{(n)} \rangle_n^2}. \tag{9}$$

The SPIKE-distance is defined as the temporal average of this dissimilarity profile

$$D_S = \frac{1}{T} \int_{t=0}^{T} dt S(t). \tag{10}$$

The dissimilarity profile $S(t)$ and the SPIKE-distance D_S as its average are bounded in the interval $[0, 1]$. The distance value $D_S = 0$ is obtained for identical spike trains only. Since the dissimilarity profile is obtained from a linear interpolation of three piecewise constant quantities it is piecewise linear.

There exists a straightforward extension to the case of more than two spike trains (number of spike trains $N > 2$), the averaged bivariate distance. This average over all pairs of neurons commutes with the average over time, so it is possible to achieve the same kind of time-resolved visualization as in the bivariate case by first calculating the instantaneous average, e.g., $S^{\rm a}(t)$ over all pairwise instantaneous values $S^{mn}(t)$,

$$S^{\rm a}(t) = \frac{1}{N(N-1)/2} \sum_{n=1}^{N-1} \sum_{m=n+1}^{N} S^{mn}(t) \tag{11}$$

2.2. *Realtime SPIKE-distance*

The realtime SPIKE-distance D_{S_r} is a modification of the SPIKE-distance with the key difference that the corresponding time profile $S_r(t)$ can be calculated online because it relies on past information only. From the perspective of an online measure, the information provided by the following spikes, both their position and the length of the interspike interval, is not yet available. Like the regular (improved) SPIKE-distance D_S, this causal variant is also based on local spike time differences but now only two corner spikes are available, and the spikes of comparison are restricted to past spikes, e.g., for the preceding spike of the first spike train

$$\Delta t_{\rm P}^{(1)} = \min_i(|t_{\rm P}^{(1)} - t_i^{(2)}|), t_i < t. \tag{12}$$

Since there are no following spikes available, there is no local weighting, and since there is no interspike interval, the normalization is achieved by dividing the average corner spike difference by twice the average time interval to the preceding spikes (Eq. 5). This yields a causal indicator of local spike train dissimilarity:

$$S_r(t) = \frac{\Delta t_{\rm P}^{(1)} + \Delta t_{\rm P}^{(2)}}{4\langle x_{\rm P}^{(n)} \rangle_n}. \tag{13}$$

2.3. *Future SPIKE-distance*

The future SPIKE-distance D_{S_f} can be used in triggered temporal averaging in order to evaluate the effect of certain spikes or of certain stimuli features on future spiking. It is the inverse measure to the realtime SPIKE-distance but instead of relying on past information only it relies on future information only. Again for each time instant there are just two corner spikes and the potential nearest spikes in the other spike train are future spikes only. Thus the spike time difference for the following spike of the first spike train reads

$$\Delta t_{\mathrm{F}}^{(1)} = \min_i(|t_{\mathrm{F}}^{(1)} - t_i^{(2)}|), t_i > t, \tag{14}$$

and accordingly for the following spike of the second spike train. In analogy to Eq. 13, an indicator of local spike train dissimilarity is obtained as follows:

$$S_f(t) = \frac{\Delta t_{\mathrm{F}}^{(1)} + \Delta t_{\mathrm{F}}^{(2)}}{4\langle x_{\mathrm{F}}^{(n)} \rangle_n}. \tag{15}$$

2.4. *The ISI-distance*

While the dissimilarity profile of the SPIKE-distance is extracted from differences between the spike times of the two spike trains, the dissimilarity profile of the ISI-distance (**??**) is calculated as the instantaneous ratio between the interspike intervals $x_{\mathrm{ISI}}^{(1)}$ and $x_{\mathrm{ISI}}^{(2)}$ (Eq. 3) according to:

$$I(t) = \begin{cases} x_{\mathrm{ISI}}^{(1)}(t)/x_{\mathrm{ISI}}^{(2)}(t) - 1 & \text{if } \ x_{\mathrm{ISI}}^{(1)}(t) \leq x_{\mathrm{ISI}}^{(2)}(t) \\ -(x_{\mathrm{ISI}}^{(2)}(t)/x_{\mathrm{ISI}}^{(1)}(t) - 1) & \text{otherwise.} \end{cases} \tag{16}$$

This ISI-ratio equals 0 for identical ISI in the two spike trains, and approaches -1 and 1, respectively, if the first or the second spike train is much faster than the other. Since the ISI-values only change at the times of spikes, the dissimilarity profile is piecewise constant. For the ISI-distance the temporal averaging analogous to Eq. 10 is performed on the absolute value of the ISI-ratio, thus both kinds of deviations are treated equally. Since the ISI-distance relies on the instantaneous ISI-values and thus requires knowledge about the following spikes, no causal realtime extension is possible.

2.5. *Computational aspects*

2.5.1. *Edge effect*

In previous versions of the SPIKE-distance the ambiguity regarding the definition of the initial (final) distance to the preceding (following) spike as well as the very first and the very last interspike intervals was resolved by adding to each spike train an auxiliary leading spike at time $t = 0$ and an auxiliary trailing spike at time $t = T$. This lead to spurious synchrony at the edges where by construction the dissimilarity profile reached the zero value. Here we follow a suggestion by Conor Houghton (personal communication) and at least partly correct this edge effect. We describe the correction for the beginning of the recording, it is an analogous mirror image at the end of the recording.

We count the auxiliary spikes as normal spikes which can be nearest neighbor to other spikes. But instead of calculating their spike time distance (which is always zero) we use the spike time difference of the first real spike. For the first interspike interval we know that it is at least the distance to the first spike $t_1 - t_0 = t_1$ but it could be longer. So to take the local firing rate (or its inverse) into consideration we set

$$x_{\mathrm{ISI}}(0) = \max(t_1, t_2 - t_1). \tag{17}$$

where we use the length of the first known interspike interval $t_2 - t_1$ as an upper limit of the inverse firing rate. This way we get at least a crude estimate of how much longer the first interspike interval could be.

2.5.2. *Sampling*

Earlier versions of the codes for calculating the ISI- and the SPIKE-distance relied on sampled dissimilarity profiles. Typically the precision was set to the sampling interval of the neuronal recording. Since the dissimilarity profile has to be calculated and stored for each pair of spike trains, this resulted, for each measure, in a matrix of order 'number of sampled time instants' $\times$ 'number of spike train pairs' (i.e., $\#(t_s) \times N(N-1)/2$). For small sampling intervals and a large number of recorded spike trains this lead to memory problems. In SPIKY we use an optimized and more memory-efficient way of storing the data where we make use of the fact that the dissimilarity profile $I(t)$ of the ISI-distance is piecewise constant and the dissimilarity profile $S(t)$ of the SPIKE-distance is piecewise linear with each interval running from one spike of the pooled spike train to the next one. Thus for each such interval (and for each pair of spike trains) we have to store only one value for the ISI-distance and two values for the SPIKE-distance, one at the beginning and one at the end of the interval. Typically the storage space required will be much smaller than for dissimilarity profiles sampled with a reasonable precision.

For both dissimilarity profiles there are instantaneous jumps at the times of the spikes since this is where the lengths of the interspike intervals and the identity of the previous and the following spikes change abruptly. In contrast to the calculation based on sampling we get the exact result since each spike is both the previous and the next spike and there is no need anymore to 'cut the corners' of the dissimilarity profiles as had to be done for the sampled dissimilarity profiles. The dissimilarity profiles $S_r(t)$ and $S_f(t)$ of the real-time and the future SPIKE-distances are hyperbolic but for these measures the exact result can be obtained by piecewise integration over all intervals of the pooled spike train.

2.6. *Representations*

The ISI- and the SPIKE-distance combine a variety of properties that make them well suited for applications to real data. In particular, they are conceptually simple, computationally efficient, and easy to visualize in a time-resolved manner. By taking into account only the preceding and the following spike in each spike train, these distances rely on local information only. They are also time-scale-adaptive since the information used is not contained within a window of fixed size but rather within a time frame whose size depends on the local rate of each spike train.

Moreover, the sensitivity to spike timing and the instantaneous reliability achieved by the SPIKE-distance opens up many new possibilities in multi-neuron spike train analysis (**?**). These build upon the fact that there are several levels of information reduction.

2.6.1. *Full matrix and cross sections*

The starting point is the most detailed representation in which one instantaneous value is obtained for each pair of spike trains (see Eq. 9). For multi-neuron data, this results in a matrix of size 'number of unique spikes in the pooled spike train' $\times$ 'number of spike train pairs' ($\times 2$ for the SPIKE-distance, see Section 2.5.2).

From this matrix, it is possible to extract any desired information. By selecting a pair of spike trains, one obtains the bivariate dissimilarity profile $S(t)$ for this pair of spike trains. Selecting a time instant t_s and using linear interpolation between the stored values, if necessary, instead yields an instantaneous matrix of pairwise spike train dissimilarities $S_{mn}(t_s)$. This matrix can be used to divide the spike trains into instantaneous clusters, that is, groups of spike trains with low intra-group and high inter-group dissimilarity.

2.6.2. *Spatial and temporal averaging*

Another way to reduce the information of the dissimilarity matrix is averaging. There are two possibilities that commute: the spatial average over spike train pairs and the temporal average.

The local average over spike train pairs yields a dissimilarity profile for the whole population. Temporal averaging over certain (continuous or noncontinuous) intervals on the other hand leads to a bivariate distance matrix (in real data, these intervals could be chosen to correspond to different external conditions such as normal vs. pathological, asleep vs. awake, target vs. non-target stimulus, or presence/absence of a certain channel blocker). Finally, in both cases, application of the respective remaining average results in one distance value that describes the overall level of synchrony for a group of spike trains over a given time interval.

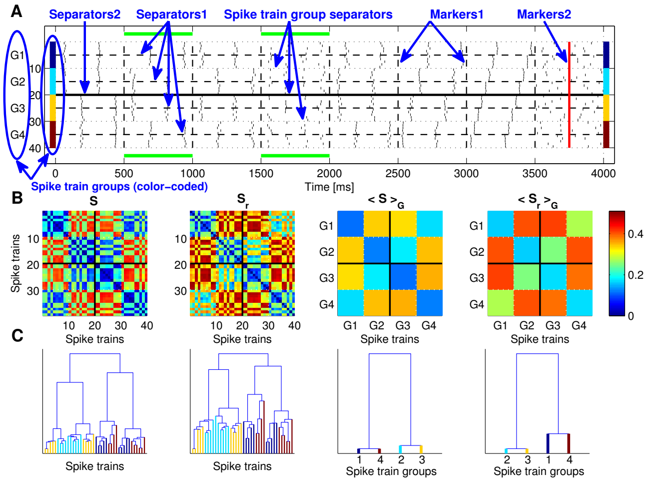

Fig. 2. Annotated screenshot from a movie. A. Artificially generated spike trains. B. Dissimilarity matrices obtained by averaging over two separate time intervals for both the regular and the real-time SPIKE-distance as well as their averages over subgroups of spike trains (denoted by $<\ >_G$). C. Corresponding dendrograms.

2.6.3. *Triggered averaging*

The fact that there are no limits to the temporal resolution allows further analyses such as internally or externally triggered temporal averaging. Here, the matrices are averaged over certain trigger time instants only. The idea is to check whether this triggered temporal average is significantly different from the global average since this would indicate that something peculiar is happening at these trigger instants. These trigger times can either be obtained from internal conditions (such as the spike times of a certain spike train) or from external influences (such as the occurrence of certain features in a stimulus). In real multi-neuron data, internal triggering might help to uncover the connectivity in neural networks or to detect converging or diverging patterns of firing propagation. External triggering might be helpful in addressing questions of neuronal coding, for example, it could be used to evaluate the influence of localized stimulus features on the reliability of real neurons under repeated stimulation.

A last possibility is spatial averaging such that the spike trains are manually assigned to subgroups, and a block matrix (and the corresponding dendrogram) is obtained by averaging over the respective submatrices of the original dissimilarity matrix. In applications to real data, these groups could be different neuronal populations or responses to different stimuli, depending on whether the spike trains were recorded simultaneously or successively.

3. SPIKY

SPIKY is a graphical user interface (GUI) for monitoring synchrony between artificially simulated or experimentally recorded neuronal spike trains. It is based on two recently proposed methods to measure spike train (dis)similarity, the ISI- and the SPIKE-distance. SPIKY is a free software package programmed by Thomas Kreuz and Nebojsa

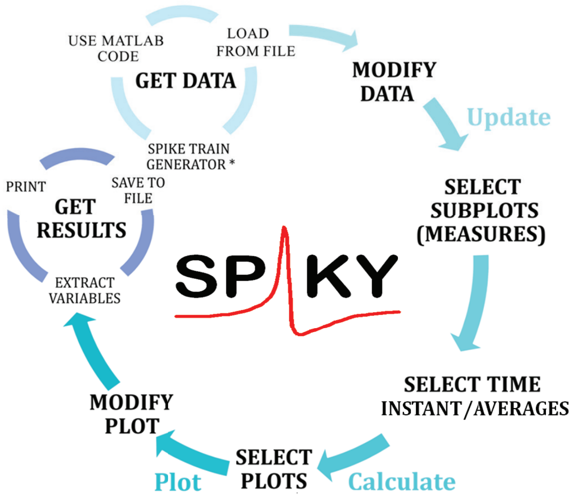

Fig. 3. SPIKY-Flowchart

Bozanic. All source codes are written in Matlab (MathWorks Inc, MA, USA) with the most time-consuming loops coded in MEX-files (in our case these are subroutines written in C). It is not stand-alone but requires Matlab to run

XXXXX Update, now stand-alone? XXXXX.

3.1. *Structure of SPIKY*

First extract the zip-package (all files should be in one directory named SPIKY), then within Matlab go to this directory or add it to the path (addpath). Compile the MEX-files using 'SPIKY_compile_MEX', finally run SPIKY.

Typically the suggested element for the next user action is marked by a bold font. The most complete initial example is the entry 'Clustering' in the Data listbox. We suggest to follow this example through till the end advancing from panel to panel by pressing the highlighted button. With the same example in a second step you can start to change some parameters and see the consequences. Later, if you do not want to set all the parameters each time when you start SPIKY with a new dataset have a look at the file 'SPIKY_f_user_interface' and try to understand how for this example the spike train and the parameter values have been created. Hopefully you should then be able to do similar things with your own datasets. For quick information about the individual elements of the GUI activate the Hints-checkbox in the Options-Menu and then hover with the mouse cursor above the elements of interest and you will see short hints. An overview of all this information can be found in the file SPIKY-Elements.doc.

3.2. *Input*

There are three different possibilities to input spike train data into SPIKY.

The first option is to select one of the predefined examples which are generated using Matlab-code. Initially these are the examples used in (**?**) but one can also define new examples.

The second option is to load spike train data from a file. Two different file formats are allowed, '.mat' and '.txt' (ASCII) files. For the mat-files SPIKY currently allows three different kinds of input formats (further formats can be added on demand).

- cell arrays (ca) with just the spike times. This is the preferred format used by SPIKY since it is most memory efficient. The two other formats will internally be converted into this format.
- regular matrices with each row being a spike train and zero padding (zp) in case the spike numbers are different.
- matrices representing time bins where each zero/one (01) indicates the absence/presence of a spike

If case of a mat-file SPIKY looks for a variable called 'spikes', if it cannot find it you have the chance to select the variable name (or field name) which contains the spikes via an input mask which provides a hierarchical structure tree of the variable structure. In the text format spike times should be written as a matrix with each row being one spike train. The SPIKY-package contains one example file for all four formats ('testdata_ca.mat', 'testdata_zp.mat', 'testdata_01.mat' and 'testdata.txt').

The third option is to create new spike train data via the spike train generator. After setting some defining variables (number of spike trains, start and end time, sampling interval) you can build your spike trains by using predefined spike train patterns (such as periodic, splay, uniform or Poisson) and/or by manually adding, shifting and deleting individual spikes or groups of spikes.

3.3. *Output*

From within SPIKY it is possible to extract the spike trains and the results of the analyses (measure profiles, matrices, dendrograms) to the Matlab workspace for further processing. When one clicks the right mouse button on the element whose data one wishes to extract results will be stored in variables such as 'SPIKY_spikes', 'SPIKY_profile_X_1', 'SPIKY_profile_Y_1', 'SPIKY_profile_name_1' as well as 'SPIKY_matrix_1' and 'SPIKY_matrix_name_1'. In addition, the results obtained during an analysis will automatically be stored in the output structure 'SPIKY_results' which will have one field for each measure selected. Depending on the parameter selection within SPIKY, for each measure the structure can contains the following subfields which largely correspond to the different representations identified in Section 2.6:

– SPIKY_results.¡Measure¿.name: Name of selected measures (helps to identify the order within all other variables)
– SPIKY_results.¡Measure¿.distance: Level of dissimilarity over all spike trains and the whole interval. This is just one value, obtained by averaging over both spike trains and time
– SPIKY_results.¡Measure¿.matrix: Pairwise distance matrices, obtained by averaging over time
– SPIKY_results.¡Measure¿.x: Time-values of overall dissimilarity profile
– SPIKY_results.¡Measure¿.y: Overall dissimilarity profile obtained by averaging over spike train pairs

Note that the dissimilarity profiles are not equidistantly sampled. Rather they are stored as memory-efficiently as possible which means just one value for each interval of the pooled spike train for the ISI- and two values for the SPIKE-distance. Since this format can be more difficult to process, the functions 'SPIKY_f_selective_averaging', 'SPIKY_f_triggered_averaging', and 'SPIKY_f_average_pi' are provided in order to compute the selective average over time intervals, the triggered over time instants, or the average over many dissimilarity profiles, respectively. Furthermore, for the ISI-distance the function 'SPIKY_f_pico.m' can be used to obtain the average value as well as the x- and y-vectors for plotting.

Besides the standard way to work with Matlab-figures SPIKY also offers the opportunity to save each figure as a postscript-file. Finally, it is possible to save a sequence of images as an 'avi'-movie.

3.4. *Figure-Layout*

SPIKY was designed in a way that allows to directly generate figures suitable for publication. To this aim the user is given control over the appearance of every individual element (e.g. fonts, lines etc.) in each type of figure. There are two ways to determine essential properties such as color, font size or line width. Most conveniently, one can use the file 'SPIKY_f_user_interface' to define the standard values for all the parameters that describe the principal layout of the figure. But it is also possible to change elements in the active figure while the program is already running. To do so the user has to simply click the right mouse button on the element to be changed. A context menu will appear which lets the user either edit either the properties of individual elements or of all elements of a certain type. This also includes the string property of any font (title, x- and y-labels etc.).

If a figure contains more than one subplot (besides the combined subplot containing the spike rasterplot and dissimilarity profiles these are typically subplots with dissimilarity matrices and dendrograms), it is also possible to change their position and size. To do so just move the cursor to the respective axis (either just left or just below the subplot) and click the right mouse button. Now one can edit all position variables by hand or change the x-position, the y-position, the width and the height individually. In case there are several dissimilarity matrices / dendrograms one can do this either for an individual matrix / dendrogram or for all of them at the same time.

3.5. *GUI vs. loop*

How can I get the statistics of a certain quantity over a large number of datasets?

For this purpose you can use the program SPIKY_loop which is complementary to the graphical user interface SPIKY'. Both programs can be used to calculate time-resolved spike train distances (ISI and SPIKE) between two (or more) spike trains. However, whereas SPIKY was mainly designed to facilitate the detailed analysis of one dataset, SPIKY_loop is meant to be used in order to compare the 'SPIKY_results' for many different datasets (e.g. in some kind of loop). Note that the new program 'SPIKY_loop' uses the full functionality of SPIKY (access to time instants, selective and triggered averages as well as averages over spike train groups).

3.6. *Spike train surrogates and significance*

An important question that has not yet been asked is the one of statistical significance. Given a certain value of the SPIKE-distance how can one judge whether it reflects a significant decrease or increase in spike train synchrony and does not just lie within the range of values obtained for random fluctuations. One way to address this question is the use of spike train surrogates (**???**). The idea is to compare the results obtained for the original dataset versus the results obtained for spike train surrogates generated from that dataset. If the value obtained for the original lies outside the range of values for the surrogates this value can be assumed to be significant to a level defined by the number of surrogates used (e.g. $\alpha = 0.05$ for 19 surrogates or $\alpha = 0.001$ for 999 surrogates).

The SPIKY-package contains a program 'Spiky_loop_surro' which was designed to look at significance. So far it includes four different types of spike train surrogates. They differ in the properties that are preserved and maintain either the individual spike numbers (obtained by shuffling the spikes), the individual interspike interval distribution (obtained by shuffling the interspike intervals), the pooled spike train (obtained by shuffling spikes among the spike trains) or the peri-stimulus time histogram (PSTH) (obtained by drawing from the probability density function (PDF) of the data).

3.7. *Comparison with other implementations*

Python-Implementation of SPIKE-distance courtesy of Jeremy Fix:

`http://jeremy.fix.free.fr/Softwares/spike.html`

Python-Implementation of the pairwise ISI-distance courtesy of Michael Chary:

https://pypi.python.org/pypi/ISIpy/1.0.1

Răzvan Florian:

https://github.com/modulus-metric/spike-train-metrics (?)

HRLAnalysis^{TM} (?)

XXX Nebojsa figure (Performance comparison) XXX

3.8. *Access to SPIKY*

SPIKY is distributed under a BSD licence (). A zip-package containing all the necessary files can be accessed for free on the download page (http://www.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/SPIKY.html). This package also contains a folder with lots of documentation (such as a FAQ-file). Further information and some demonstrations (both images and movies) can be found on the download page and on the SPIKY Facebook-page (https://www.facebook.com/SPIKYgui). Both of these pages will be used to announce updates and distribute the latest information about new features. The Facebook-page also provides the user with an opportunity to provide feedback and ask questions. Finally, the movies can also be viewed on the SPIKY Youtube-channel (https://www.youtube.com/channel/UCgSz0YQ5lWdVF0_Z1FNN0Bw).

4. Discussion

So far the SPIKE-distance has been applied in the following papers:

(????)

XXXXX Update at the end XXXXX

4.1. *Summary*

4.2. *Limitations*

The calculation of the SPIKE-distance consists of three steps: First for each spike the distance to the nearest spike in all the other spike trains is calculated. Successively, for each time instant and each pair of spike trains, the distances of the four corner spikes are first locally weighted and then normalized. These latter steps involve matrices of the order 'number of time instants' 'number of spike train pairs', which for very long datasets with many spike trains can lead to memory problems. The solution to this problem is to make the calculation sequential, i.e., to cut the recording interval into smaller segments, and to perform the averaging over all pairs of spike trains for each segment separately. In the end the dissimilarity profiles for the different segments (already averaged over pairs of spike trains) are concatenated, and its temporal average yields the distance value for the whole recording interval. If such a sequential calculation is required the user is continuously informed about the progress via a waitbar.

4.3. *Outlook*

Stand-Alone, Python/C++

Acknowledgements

NB and TK acknowledge funding support from the European Commission through the Marie Curie Initial Training Network 'Neural Engineering Transformative Technologies (NETT)', project 289146. TK also acknowledges the Italian Ministry of Foreign Affairs regarding the activity of the Joint Italian-Israeli Laboratory on Neuroscience.

TK thanks Marcus Kaiser for hosting him at the University of Newcastle, UK.

We thank Emily Caporello, Daniel Chicharro, Tim Gentner, Conor Houghton, Jutta Kretzberg, Stefano Luccioli, Florian Mormann, Leon Paz, Friederice Pirschel, Alessandro Torcini, Jonathan Victor for useful discussions.

We also thank Thomas Alderson, Black Square, Mayte Bonilla Quintana, Hamid Charkhkar, Didier Desaintjan, Mario DiPoppa, Mahboubeh Etemadi, Marion Najac, Matthew Phillips, Eugenio Piasini, Robert Rein, Rodrigo Salazar, Michael Schaub, Eitan Schechtman, Matthew Williams, Yunguo Yu for advice and user feedback.