SPIKY

G+1 +3 Consiglialo su Google

Nebojsa Bozanic et al. (2014), Scholarpedia, 9(12):32344. doi:10.4249/scholarpedia.32344 revision #146120 [link to/cite this article]

- **Nebojsa Bozanic**, Institute for complex systems (ISC), National research council (CNR), Sesto Fiorentino, Italy
- **Dr. Mario Mulansky**, Institute for Complex Systems (ISC), National Research Council (CNR), Sesto Fiorentino, Firenze, Italy
- **Thomas Kreuz**, Institute for complex systems (ISC), National research council (CNR), Sesto Fiorentino, Italy

SPIKY is a graphical user interface written in Matlab that facilitates the application of time-resolved measures of spike train synchrony to both simulated and real data. It contains several approaches to analyze spike train synchrony: the standard Peri-Stimulus Time Histogram (PSTH), the ISI-distance, SPIKE-distance, and SPIKE synchronization.

For a given data set SPIKY calculates the measures of choice and allows the user to switch between many different visualizations such as dissimilarity profiles, pairwise dissimilarity matrices, or hierarchical cluster trees. SPIKY also includes a spike train generator and an event detector which makes it capable of analyzing continuous data. Finally, the SPIKY-package (http://wwwold.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/SPIKY.html) includes complementary programs for the automatized analysis of a large number of datasets and for the evaluation of the statistical significance of the results.

Contents

Measures of spike train (dis)similarity

All measures implemented in SPIKY allow analyzing two or more neuronal spike trains on different spatial and temporal scales. The Peri-Stimulus Time Histogram is the only multivariate measure. The three bivariate measures are complementary to each other since each one addresses a different specific aspect of spike train similarity. While the ISI-distance quantifies local dissimilarities based on covariances of the neurons firing rate profiles, both the SPIKE-distance and SPIKE synchronization capture the relative timing of local spikes. The SPIKE-distance weights and normalizes the differences between nearest neighbor spikes, whereas event synchronization acts as a binary coincidence detector.

For all of the bivariate measures there exists a straightforward extension to the case of more than two spike trains (spike train number $N > 2$), the averaged bivariate distance. This average over all pairs of spike trains commutes with the average over time, so by first calculating the instantaneous average over all pairwise instantaneous values it is possible to achieve the same kind of time-resolved visualization as in the bivariate case.

ISI-distance

For more detailed information, please refer to ISI-distance.

The ISI-distance, proposed as a bivariate measure in Kreuz et al. (2007) (http://10.1016/j.jneumeth.2007.05.031) and extended to the multi spike train case in Kreuz et al. (2009) (http://10.1016/j.jneumeth.2009.06.039) was the first spike train distance directly defined as the temporal average of an instantaneous dissimilarity profile. Since the interspike intervals only change at the times of spikes, the dissimilarity profile is piecewise constant (with discontinuities at the spikes). It equals zero for identical interspike intervals in the two spike trains, and approaches one in regions where one spike train is much faster than the other.

SPIKE-distance

For more detailed information, please refer to SPIKE-distance.

The SPIKE-distance (Kreuz et al., 2013 (http://10.1152/jn.00873.2012)) is the centerpiece of SPIKY. In contrast to the ISI-distance, it considers the exact timing of the spikes. The dissimilarity profile is derived from a linear interpolation of piecewise constant quantities which makes it piecewise linear (with potential discontinuities at the spikes). Both the dissimilarity profile and the SPIKE-distance are bounded in the interval $[0, 1]$. The distance value of zero is obtained for identical spike trains only.

The SPIKE-distance comes with two further variants. The realtime SPIKE-distance can be calculated online (and in real-time) because it relies on past information only. The forward SPIKE-distance is 'inverse' to the profile of the realtime SPIKE-distance in the sense that it relies on forward information only. It can be used in order to evaluate the (causal) effect of certain spikes or of specific stimuli features on future spiking.

SPIKE synchronization

For more detailed information, please refer to SPIKE synchronization.

SPIKE synchronization is a much simplified version of event synchronization (Quian Quiroga et al., 2002 (http://10.1103/PhysRevE.66.041904) ; Kreuz et al., 2007b (http://10.1016/j.physd.2006.09.039)), a sophisticated coincidence detector. It quantifies the degree of synchrony from the relative number of quasi-simultaneous appearances of spikes. There are two variants which differ in the way a coincidence is defined. The main variant is parameter- and scale-free, since the coincidence window, i.e., the time lag up to which two spikes are considered to be synchronous, is adapted to the local spike rates. The other variant uses a coincidence window of fixed size. Both variants have a parameter that governs the temporal resolution. In contrast to the two spike train distances, SPIKE synchronization is oriented as a measure of similarity, i.e., it yields one if (and only if) all spikes coincide.

Graphical user interface

Overall, the workflow of SPIKY (Figure 1) is linearly directed from the input of spike train data to the output of results. However, the program is much more interactive than previous implementations of the measures.

Input

There are four different ways for importing data: one can make use of predefined examples, load data from a file or from the Matlab workspace, detect discrete events from continuous data, or employ the spike train generator.

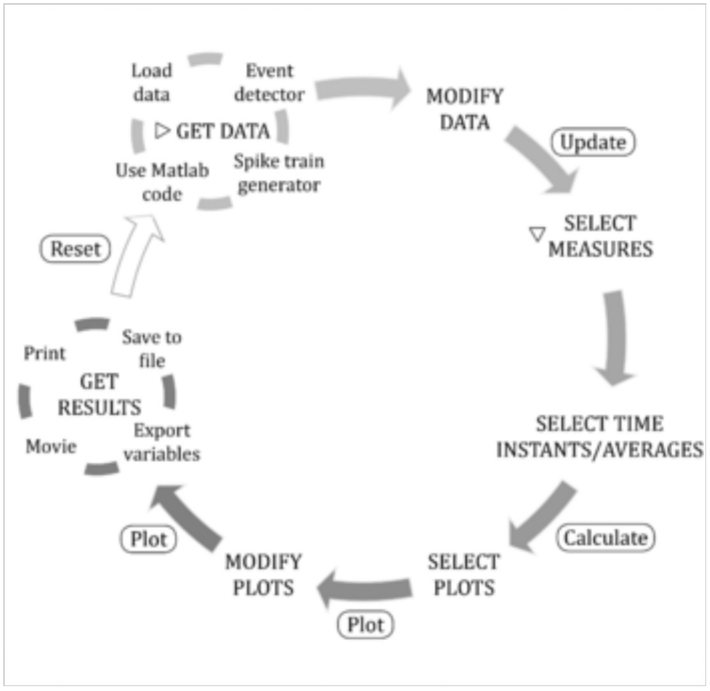

Figure 1: SPIKY's flowchart.

The first option is to select one of the predefined examples which are generated using Matlab-code. It is also possible to define new examples.

When loading spike train data from the Matlab workspace or from a file, two different file formats are accepted: '.txt' (ASCII) and'.mat' files. For the latter SPIKY currently allows three different kinds of input formats: cell arrays, rectangular matrices with zero padding, and time bin matrices with ones/zeros indicating presence/absence of a spike.

SPIKY also includes an event detector which can be used to transform continuous data into discrete events. There are many different possibilities of defining an event. A variety of standard events (such as threshold crossings and local maxima/minima) with a number of parameters are already included.

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

Figure layout

Once the full dataset is available, modification is still possible. The user can restrict the analysis to a specific subset, e.g., select a smaller time window and/or a subset of spike trains. It is also possible to impose some external structure on the raster plot (spike trains vs time). For that, SPIKY allows the definition of two types of time markers (e.g. thick/thin markers for specific events such as seizure onset/offset in epilepsy) and two types of spike train separators (e.g. a thick separator for neurons from the left vs. neurons from the right hemisphere and a thin separator for different regions within the two halves). The user can also define spike train groups. Depending on the setup these could be spike trains recorded in different brain regions or upon presentation of different stimuli. Figure 2 shows an example of a raster plot with annotations marking all these different elements.

SPIKY is designed so that it can directly generate figures suitable for publication. The user is given control over the appearance of every individual element (e.g. fonts, lines etc.) in each type of figure. Essential graph properties such as color, font size or line width can either be determined beforehand or they can be changed via context menus while the program is running. If a figure contains more than one subplot (besides the subplot containing the spike raster plot and/or the dissimilarity profiles, these are typically subplots with dissimilarity matrices and dendrograms), it is possible to change both their position and their size.

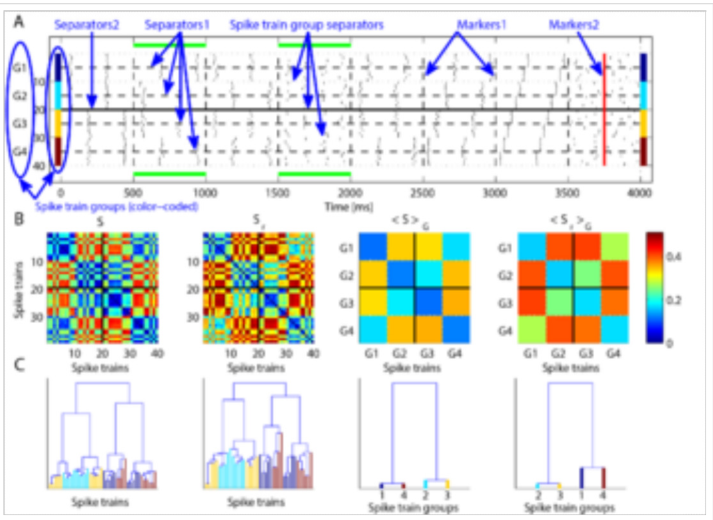

Figure 2: SPIKY's annotated screenshot. A. Artificially generated spike trains. B. Dissimilarity matrices obtained by averaging over two separate time intervals for both the regular and the real-time SPIKE-distance as well as their averages over subgroups of spike trains (denoted by <.>G). C. Corresponding dendrograms.

Output

In SPIKY the spike trains and the overall results of an analysis will automatically be stored in an output structure with one field for each measure. In addition, from within SPIKY it is also possible to extract the results of a specific analysis (for example profiles, matrices, or dendrograms) to the Matlab workspace for further processing.

Dissimilarity profiles are not equidistantly sampled, rather they are stored memory-efficiently with 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 difficult to process, SPIKY includes functions for computing averages over dissimilarity profiles, selective averages over time intervals, and triggered averages over time instants.

Finally, SPIKY also offers the opportunity to save each figure as a postscript-file and to turn a sequence of images into an 'avi'-movie.

Complementary programs

Dataset queue

SPIKY was mainly designed to facilitate the detailed analysis of a single dataset (consisting of one group of spike trains). It enables the user to switch between different representations and to zoom in on both spatial and temporal features of interest. However, SPIKY is less convenient for the analysis of many different datasets, for example when the statistics of a certain quantity such as an average over a specific time interval should be evaluated over all available datasets (e.g. over all trials of a stimulus setup or for recordings of all subjects etc.). For these purposes the SPIKY-package contains a complementary automatized program called *SPIKY_loop*. While not providing a graphical user interface, this program still contains the full functionality of SPIKY and gives access to time instants, selective and triggered averages as well as averages over spike train groups.

By combining these two programs, it is possible to first use SPIKY for a rather exploratory but detailed analysis of a limited number of individual datasets and then use SPIKY_loop and its output structure to verify whether any effect discovered on the example dataset is consistently present within all of the datasets. Both programs are able to deal with datasets containing more than hundred spike trains and overall more than a million spikes.

Statistical significance

An important aspect when using distance measures is the statistical significance of the obtained values, e.g. if an observed change in time is indeed due to some change in the spiking dynamics or just a random fluctuation. One approach to this problem is the use of spike train surrogates (Kass et al., 2005 (http://10.1152/jn.00648.2004) ; Grün et al., 2009 (http://10.1152/jn.00093.2008) ; Louis et al., 2010 (http://10.3389/fncom.2010.00127)). The idea is to compare the results 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* designed to evaluate 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 resampling). Other ways to calculate rate functions (e.g. based on kernels with different bandwidths) will be added in future releases.

System requirements & Availability

All source codes are written in Matlab (MathWorks Inc, Natick, MA, USA) with the most time-consuming loops coded in MEX-files. Consequently, SPIKY is not stand-alone but requires Matlab to run. A zip-package containing all the necessary files can be accessed for free on the SPIKY download page (http://www.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/SPIKY.html) . This package also contains a folder with documentation (such as a FAQ-file and an introduction to all individual elements of SPIKY and all individual files of the SPIKY-package). Further information and many demonstrations (both images and movies) can be found on the download page and on the SPIKY Facebook-page (https://www.facebook.com/SPIKYgui) . Finally, the movies are also available on the SPIKY Youtube-channel (https://www.youtube.com/user/SPIKYgui1) .

Alternative Python package: PySpike

PySpike is a Python library that provides implementations of the ISI-distance, SPIKE-distance and SPIKE-Synchronization, as well as some functions for loading, generating and plotting spike trains. In contrast to SPIKY, it does not include a graphical user interface (GUI) and hence requires some programming experience. PySpike is an open source library and freely available on Github (https://github.com/mariomulansky/PySpike) and PyPi (https://pypi.python.org/pypi/pyspike) . It is independent of Matlab and only needs Python (http://www.python.org) to run. The core functions of PySpike are implemented in Cython (http://cython.org) to ensure optimal performance. PySpike comes with a tutorial-like documentation (http://mariomulansky.github.io/PySpike) as a starting point for new users and a reference (http://mariomulansky.github.io/PySpike/pyspike.html) for experienced users.

References

- Grün S (2009). Data-Driven Significance Estimation for Precise Spike Correlation. J Neurophysiol, 101:1126-40.

- Kass RS, Ventura V, Brown EN (2005). Statistical issues in the analysis of neuronal data. J Neurophysiol, 94:8-25.
- Kreuz T, Haas JS, Morelli A, Abarbanel HDI, Politi A (2007). Measuring spike train synchrony. J Neurosci Methods 165:151–161.
- Kreuz T, Mormann F, Andrzejak RG, Kraskov A, Lehnertz K, Grassberger P (2007b). Measuring synchronization in coupled model systems: A comparison of different approaches. Phys D 225:29–42.
- Kreuz T, Chicharro D, Andrzejak RG, Haas JS, Abarbanel HDI (2009). Measuring multiple spike train synchrony. J Neurosci Methods 183:287–299.
- Kreuz T, Chicharro D, Greschner M, Andrzejak RG (2011). Time-resolved and time-scale adaptive measures of spike train synchrony. J Neurosci Methods 195:92–106.
- Kreuz T, Chicharro D, Houghton C, Andrzejak RG, Mormann F (2013). Monitoring spike train synchrony. J Neurophysiol 109:1457-72.
- Louis S, Gerstein GL, Grün S, Diesmann M (2010). Surrogate spike train generation through dithering in operational time. Front Comp Neurosci, 4:127.
- Quian Quiroga R, Kreuz T, Grassberger P (2002). Event synchronization: a simple and fast method to measure synchronicity and time delay patterns. Phys Rev E 66:041904.

Internal references

- James Meiss (2007) Dynamical systems. Scholarpedia, 2(2):1629.
- Arkady Pikovsky and Michael Rosenblum (2007) Synchronization. Scholarpedia, 2(12):1459.
- David Golomb (2007) Neuronal synchrony measures. Scholarpedia, 2(1):1347.
- Jose Pedro Segundo (2010) Spike train and point processes. Scholarpedia, 5(7):5729.
- Thomas Kreuz (2011) Measures of neuronal signal synchrony. Scholarpedia, 6(12):11922.
- Thomas Kreuz (2011) Measures of spike train synchrony. Scholarpedia, 6(10):11934.
- Thomas Kreuz (2012) SPIKE-distance. Scholarpedia, 7(12):30652.
- Conor Houghton (2013) Population measures of spike train synchrony. Scholarpedia, 8(10):30635.

External Links

- SPIKY download page (http://www.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/SPIKY.html)
- SPIKY Facebook page (https://www.facebook.com/SPIKYgui)
- SPIKY Youtube-channel (https://www.youtube.com/user/SPIKYgui1)
- Movie (Youtube): General introduction to SPIKY (https://www.youtube.com/watch?v=g0r3LI7eBS0)
- Movie (Youtube): How to use the spike train generator in SPIKY (https://www.youtube.com/watch?v=Eo-b54zZsbc)
- Movie (Youtube): How to use SPIKY to sort spike trains according to their similarity (https://www.youtube.com/watch?v=y9iqYD4UtYE)
- PySpike Github repository (https://github.com/mariomulansky/PySpike)
- Author's website (Thomas Kreuz) (http://www.fi.isc.cnr.it/users/thomas.kreuz/)
- Author's website (Mario Mulansky) (http://www.mariomulansky.de)

Sponsored by: Conor Houghton, Department of Computer Science, University of Bristol, England
Reviewed by (http://www.scholarpedia.org/w/index.php?title=SPIKY&oldid=0) : Conor Houghton, Department of Computer Science, University of Bristol, England

Reviewed by (http://www.scholarpedia.org/w/index.php?title=SPIKY&oldid=145812) : Dr. daniel chicharro, Istituto Italiano di Tecnologia, genova, Italy

Accepted on: 2014-12-23 22:10:35 GMT (http://www.scholarpedia.org/w/index.php?title=SPIKY&oldid=146014)

Categories: Computational neuroscience | Dynamical Systems | Network Dynamics | Signal Analysis | Spiking Networks | Synchronization