
Check for updates

et al., 2017).

Here, we introduce a modification of the significance assessment of STPs by SPADE to explicitly account for the pattern duration as a third parameter to consider besides the pattern size and the occurrence count. We define the pattern duration as the time difference between the first and the last spikes of the pattern. We refer to the modified approach as 3d-SPADE. The novel method computes the significance threshold separately for patterns of different durations, enabling – compared to the previous 2d-version – a more reliable detection of patterns of longer duration and low occurrence count. The improved performance is demonstrated on synthetic data. We also introduce here the implementation of SPADE (both in its 2d and 3d versions) as a module of the Python-based Electrophysiological Data Analysis Toolbox (Elephant, RRID:SCR_003833)[2], which provides a large number of analysis methods for electrophysiological (spikes and LFP) data.

2. 2d-SPADE

Here we briefly introduce 2d-SPADE (below, "SPADE" for short), the original method that 3d-SPADE builds upon.

SPADE consists of three successive steps: (a) mining all different patterns present in the data at a prescribed high temporal resolution, and counting their repetitions, (b) testing the statistical significance of the detected patterns under the null-hypothesis that neurons do not coordinate at that temporal resolution, and (c) removing those identified patterns that can be considered as a by-product of chance overlaps of true pattern spikes and background activity. The first step addresses the problem of extracting spike patterns in parallel recordings, and solves it by employing a frequent pattern mining algorithm. The second step is solved by a Monte-Carlo approach, which generates surrogate data from the original data by spike dithering (Louis et al., 2010) to implement the stated null hypothesis. It thereby determines the significance level of patterns of a given size and occurrence count, and discards patterns which are too small or too infrequent. The third step performs a conditional statistical testing, which further discards patterns selected in the second step, but that can be considered as a result of chance overlap of background spikes and other retained patterns. The three steps are detailed below.

2.1. Frequent item set mining

In order to systematically detect patterns with a fine temporal precision (millisecond scale), it is necessary to deal with a number of challenges. First, the number of possible combinations of patterns is huge (infinite in continuous time). Second, one does not expect a pattern to repeat identically at recording precision (e.g. 30 kHz), due to a lower temporal resolution of the neuronal processes ($\geq$1 ms). Therefore, one needs to allow for, and be able to identify, repetitions of patterns with some degree of temporal imprecision (up to a few milliseconds) of the inter-spike lags. Third, the number of potential patterns, that can be produced by a population of N neurons, grows exponentially with the number of spikes and neurons involved in a pattern.

We address the first two challenges by discretizing the time into exclusive time intervals (bins) of a few milliseconds. The ideal bin size corresponds to the temporal scale that is relevant for the coordination of the neuronal activity. In Grün et al. (1999) and Pazienti et al. (2008) we found that the time scale for which neuronal correlation is maximal

[2] http://python-elephant.org.

time bin. In these set
the array. Mathematically, a pattern occurring
window W spanning w bins can be represented as a set $\{(b_1, n_1), \ldots, (b_k, n_k)\}$, where k is the number of spikes comprising a pattern, i.e. b_i is the analysis window's bin where the ith spike of the pattern falls, and n_i identifies the neuron that emitted that ith spike. The total occurrence count of a pattern is determined by shifting the analysis window to the right, and by counting the number of times the pattern is found across all windows. If a pattern occurrence is found, the comprising spikes are not re-used to find further occurrences of the same pattern, in order to avoid artificially increasing the pattern's occurrence count.

The number of potential patterns grows exponentially as a function of the number of neurons and bins. Even for short recordings from a moderate number of simultaneously recorded neurons, listing all possible patterns with their repetition counts becomes soon computationally infeasible. SPADE solves this problem by only considering patterns that (1) involve a minimum number of neurons (i.e. a parameter of the analysis, default is 2), (2) occur a minimum number of times (*frequent sets*), and (3) are not subsets of larger patterns occurring more often (*closed sets*). Patterns which do not meet these criteria are considered due to chance (or indistinguishable from chance patterns), without the need for further statistical assessment. *Frequent item set mining* (FIM, Agrawal et al., 1993) and *formal concept analysis* (FCA, Wille, 1982) provide efficient algorithms to search for such structures in the data. FIM and FCA cast the same problems in different formulations, leading to equivalent solutions (Quaglio et al., 2017; Yegenoglu et al., 2016). We stick here to FIM for consistency with our previous publications on the SPADE method (Torre et al., 2013, 2016).

Given parallel binarized spike trains of N neurons, an *item set* is defined as a set of spikes that form a pattern, and its *support* as the number of times that the item set occurs in the data (Picado-Muiño et al., 2013). We are interested in closed frequent item sets, that is, patterns that occur a predefined minimum number of times, and with a support that is larger than the support of any of their supersets. In order to capture the temporal extension of the patterns, we slide a temporal window of predefined duration w along the data, bin by bin. The window length w sets the maximum duration of patterns that SPADE finds (Fig. 1). In these settings, an item set is the set of spikes that occur within each window position. First, all item sets are collected. Then, FIM operates a tree search through these item sets requesting a minimum support (frequent item sets): the search starts from patterns of minimum size, and the tree is built such that its layers contain patterns of increasing size. The closed frequent item sets are obtained by successive pruning of the tree. The efficiency of the algorithm derives from the fact that each pattern is visited at most once, and that the data are stored in a compact data structure. For further details we refer to Torre et al. (2013) and Quaglio et al. (2017). The output is a list of closed frequent item sets (STP candidates) with their corresponding support (number of occurrences).

2.2. 2-Dimensional pattern spectrum filtering

The second step of the SPADE method consists in testing the statistical significance of the patterns that were detected by FIM. Due to the huge number of patterns that FIM detects in large data, testing each single pattern individually would lead to a considerable multiple testing problem. To overcome this issue, SPADE first tests pattern significance based on the pattern size z and the pattern occurrence count c only. Therefore patterns of the same *signature* (z, c) are pooled, and their count is later tested for significance. We call the matrix containing all

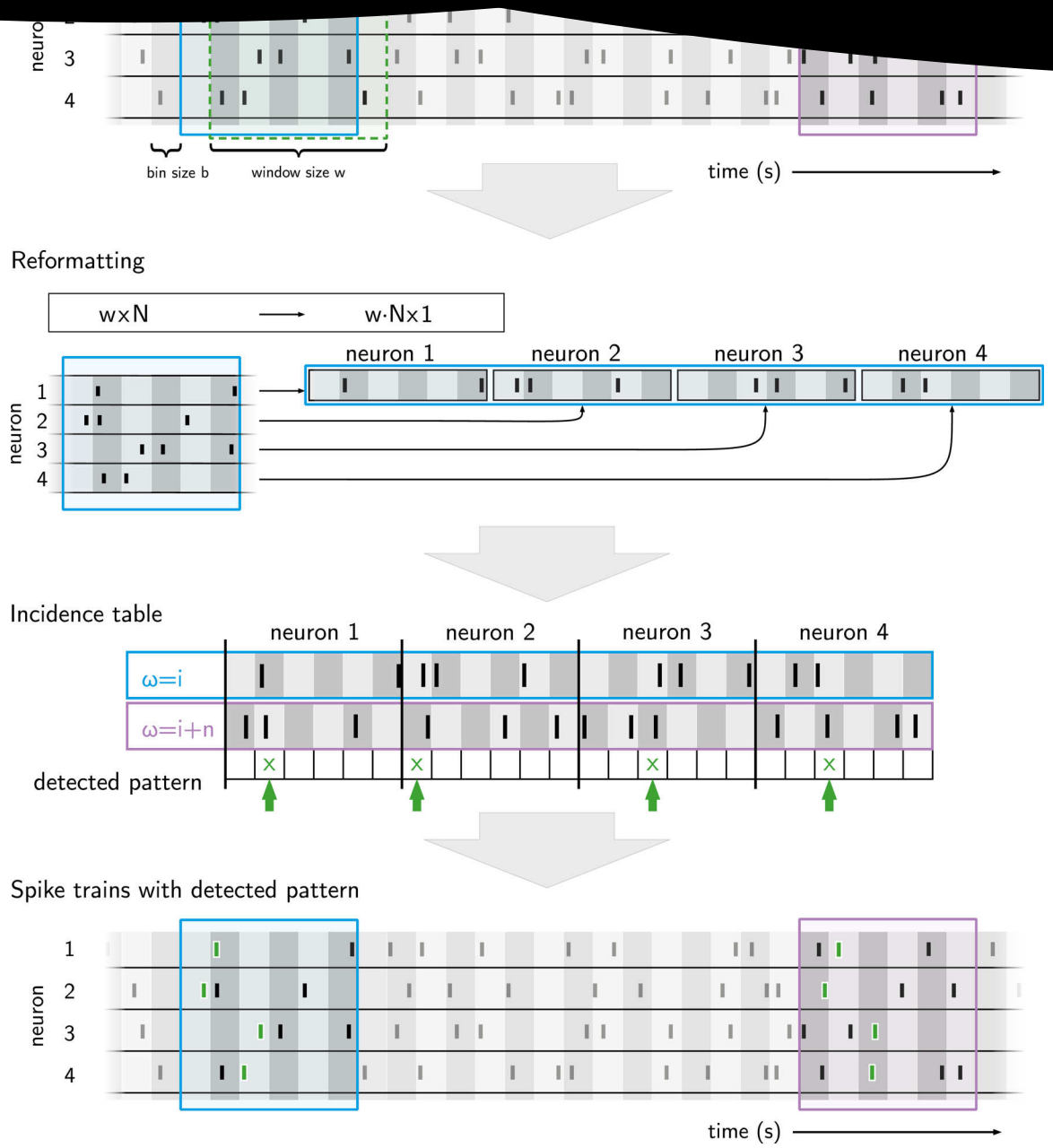

Fig. 1. Pattern extraction. Top: example of spike trains (each line represents a spike) of $N = 4$ neurons recorded in parallel. The data are binned in bins of b ms, and indicated by grey shaded vertical areas. A window of duration w is slid from bin to bin, containing spikes for pattern detection of maximal duration w. The second panel from top shows the reformatting of a window of dimensions $w \times N$ into a vector of dimensions $w{\cdot}N \times 1$. This is done for each window position (not shown here). Third panel from top (indicence table): example showing the vector representations of the windows at positions $\omega = i$ and $\omega = i + n$. Green crosses mark bins that contain spikes in both vectors. These indicate a pattern occurring identically two times. Fourth panel: same illustration as the first panel, with the detected pattern (by FIM) highlighted in green.

counts of all signatures (z, c) *pattern spectrum*. The p-value of each signature is determined by a Monte-Carlo approach that generates surrogate data from the original data by spike dithering (Date et al., 1999; Louis et al., 2010). Spike dithering randomly perturbs the exact spike timing of each spike of the original data, thereby destroying potentially existing patterns while preserving other properties of the data, such as (co-)varying neuronal firing rates. The p-value of each signature is determined by performing FIM on each of the many surrogate data sets and by computing the fraction of surrogates that contain patterns with that signature. The p-value of each signature (z, c) is stored as the element $P_{z,c}$ of the p-value matrix P, a matrix with the same dimensions as the pattern spectrum. All closed frequent patterns found in the original data having statistically significant signatures are retained, while the rest is discarded. This step, called *pattern spectrum filtering* (PSF),

are present, however, the step is prone to also classify as significant those patterns that are due to chance overlap of the real patterns with background spiking activity. These chance patterns have indeed a larger size and/or occurrence count than expected under independence (i.e. complete absence of significant patterns). To discard these false positives, a further conditional testing step is performed, called pattern set reduction (PSR). PSR evaluates the conditional significance of each pattern given other patterns that partially overlap with it. For example, if a pattern A of size z_A is a superset of pattern B of size z_B, which occurs $c_B > c_A$ times, PSR tests whether the additional occurrences of B alone are still significant under the null hypothesis. It also tests whether the pattern $A \setminus B$, which repeats z_A times, is significant. Depending on the results of this conditional testing, either one of the two patterns is discarded, or both are kept as spatio-temporal patterns (STPs) (Torre et al., 2013; Quaglio et al., 2017).

3. 3d-SPADE

Calibration on synthetic data has shown that SPADE is statistically robust, and efficiently distinguishes between chance patterns and statistically significant STPs (Quaglio et al., 2017). Yet, the same calibration also revealed that patterns with a longer duration are more likely to be detected as false positives compared to shorter patterns that occur the same number of times. This problem arises due to the fact that 2d-SPADE does not account for the pattern duration, i.e. the temporal extent of the pattern. One pattern composed of z synchronous spikes and a second one composed of z spikes spanning several bins, both occurring c times, are assigned by 2d-SPADE the same signature (z, c) and thus the same p-value. However, patterns of longer durations have a higher probability to occur by chance, due to the larger number of possible spike combinations that can be formed during that longer time span, and should therefore be assigned a lower significance value (higher p-value). As a minimal example, let us consider two different pattern durations: 1 or 2 bins. Two neurons A and B can produce only one spike pattern of size $z = 2$ in a single-bin: the synchronous pattern. There exist, instead, two possible STPs of size $z = 2$ that extend over a window of two bins: a spike from A followed by a spike from B, or vice versa. In this scenario, and under independence and firing rate homogeneity, patterns of signature $(2, c)$ extending over two time bins are therefore 2 times as likely than a pattern extending over one bin to happen by chance, and thus are correspondingly less significant. For longer (more bins) and larger (more spikes) patterns this discrepancy is even higher, given the larger number of combinations being possible. In Section 4.1 we quantify this statement by deriving an approximate calculation of how the pattern duration affects the pattern p-value under simplifying assumptions regarding the spike trains' statistics.

Furthermore, also the window length chosen for the pattern detection has an impact on the p-value assigned to patterns of the same signature (z, c). A shorter window contains fewer chance patterns under independence, thus yielding a smaller p-value.

To address the issues stated above, 3d-SPADE tests patterns with the same signature (z, c) for significance separately for each pattern duration d. It does so by replacing the signature (z, c) by the triplet (z, c, d). The corresponding pattern spectrum is now a 3-dimensional tensor with pattern duration along the third dimension (see Fig. 2, left). Also the p-value spectrum is extended to a 3-dimensional structure as is the PSF. The p-value of each 3-dimensional signature is derived independently using the same surrogate-based procedure as in the 2-dimensional version (Fig. 2, right). This approach multiplies the number of tests to be performed by the number of possible pattern durations, i.e. by the

of false positives [illegible]
(see Fig. 3).

4. Method validation

4.1. Comparison of 2d- and 3d-SPADE

Here we first investigate the statistical performance of 3d-SPADE in comparison to the original 2d-SPADE method. This test aims to demonstrate the impact of considering the pattern duration on the final assessment of the pattern's significance. To this end we apply both methods to simulated data sets comprising of $N = 100$ parallel spike trains, modeled as independent Poisson processes of constant firing rate with $\lambda = 15$ Hz and a total duration $T = 10$ s. The data are enriched with spike patterns of different durations to demonstrate the danger of false negatives in case of 2d-SPADE. We inject patterns of size $z = 3$ and occurring $c = 4$ times each, but having different durations $d = 0$ ms, 2 ms, 6 ms, 8 ms, 12 ms. Fig. 3, top shows an example raster display of a data segment containing three of the five injected patterns. The analysis window length is varied as $w = 1, 4, 7, 10, 13$ bins, with a binsize $b = 1$ ms. The green crosses in Fig. 3, middle panel, show the number of injected patterns. Only patterns of duration $d \leq w$ can be detected by FIM. For increasing window size, more and more injected patterns should be detected by FIM and should be identified as significant patterns. For 3d-SPADE this is clearly the case: all the patterns injected are detected (Fig. 3, middle panel, right: all black bars coincide with green crosses). However, in case of 2d-SPADE, no patterns are detected as significant, i.e. as STPs, for window sizes of $w = 10$ and $w = 13$.

The explanation can be found in the lower panel of Fig. 3, which shows for illustration purposes the p-values obtained using 2d-SPADE (left) and 3d-SPADE (right) for patterns of different duration d (horizontal axis), occurrence count c (vertical axis), and with fixed size $z = 3$. For each window length w, p-values are shown for patterns of different duration (x-axis) and occurrence counts for each respective signature. In both cases signatures are resolved also for the duration, but only in 3d-SPADE (right) p-values are duration-dependent. On the left, the p-values are computed for pooled signatures of different durations and are thus identical for the different durations (e.g. for each w, and c). This resulted as significant for pattern durations of $d = 1$ ms, 4 ms, 7 ms, but not for $d = 10$ ms, 13 ms. Therefore no STPs are detected for $d = 10$ ms, 13 ms. In contrast, for 3d-SPADE, the p-values vary for each pattern duration d for a given window size w, and thus all patterns of different durations are detected as STPs. Specifically, the p-values of patterns of durations $d = 0$ ms, 2 ms, 6 ms, 8 ms, 12 ms for $c = 4$ are $\simeq$0.000,0.001,0.002,0.003,0.005, respectively.

In order to understand the dependence of the p-value on the pattern duration, we provide a rough analytical derivation here. Under the assumptions that the data are stationary and all neurons have the same constant firing rates λ, we can derive an approximation of the p-value of each pattern signature (z, c, d) under the null hypothesis of independence. A pattern involving z specific neurons has probability $p = (b \cdot \lambda)^z$ to occur at a specific time, with b the bin size. The probability $\mathbb{P}(X \geq c)$ that the same pattern with a duration d occurs at least c times within the total duration T of the data is given by a cumulative binomial distribution with parameters p and $n = \frac{T-d}{b}$ (the total number of window positions). The p-value of signature (z, c, d) is then given by $p_{z,c,d} = \mathbb{P}(X \geq c) \cdot \frac{N!}{(N-z)!} \cdot (\frac{d}{b})$, where $\frac{N!}{(N-z)!} \cdot (\frac{d}{b})$ is the number of all possible combinations of patterns with z spikes and duration d. Using these equations we can compute the p-values for the five patterns of different

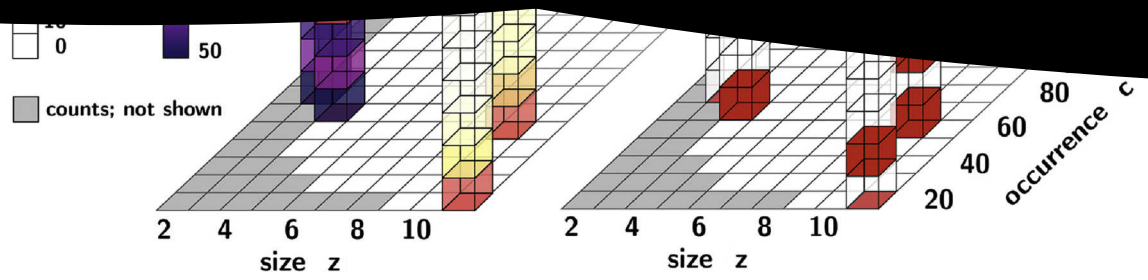

Fig. 2. 3d-pattern spectrum and PSF spectrum. The graph on the left illustrates a 3d-pattern spectrum, which counts for each size z, occurrence c and duration d (z-axis) the number of patterns occurring (color bar) with that signature. Thus a count of 2 may for example result from 2 patterns of identical signature {3 neurons, 25 occurrences, 51 ms duration}, but of different inter spike delay times and/or neuron compositions, e.g. {neuron id 1,2,3; $\Delta t_1 = 30$ ms, $\Delta t_2 = 21$ ms} and {neuron id 1, 3, 4; $\Delta t_1 = 15$ ms, $\Delta t_2 = 36$ ms}. For better visibility, the counts for low size and low occurrence counts (later resulting as non-significant; gray) are omitted. The graph on the right is the PSF spectrum after significance evaluation, containing only two types of voxel content: significant (red; containing STPs) and non-significant (light gray or transparent). Note that within a column for entries of same size and same occurrence, different durations may be significant or not.

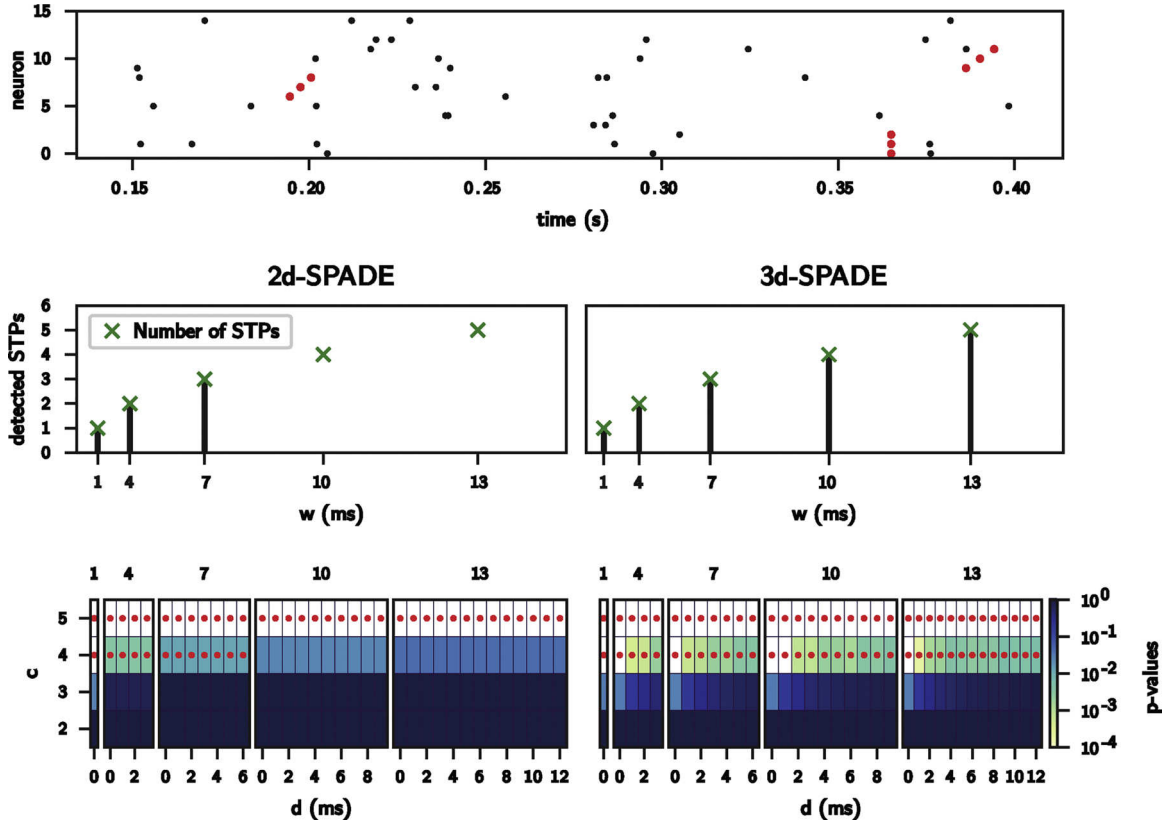

Fig. 3. Comparison of 2d- and 3d-SPADE with respect to pattern duration. Top panel: raster plot of parallel spike trains (y-axis). Background spikes are shown in black (independent Poisson processes of constant firing rate of $\lambda = 15$ Hz and a total duration $T = 10$ s), and 3 injected patterns in red. Only 15 out of 100 parallel spike trains are shown for a duration of 350 ms, and only 3 out of 6 possible patterns of size $z = 3$, of different durations ($d = 0$ ms, 2 ms, 6 ms, 8 ms, 12 ms), each injected $c = 4$ times. Middle and lower panels: 2d-SPADE results on the left, and 3d-SPADE on the right. In the middle panel, the histograms show the number of detected significant patterns (STPs) when using different analysis window length (here: $w = 1, 4, 7, 10, 13$ bins of 1 ms along the x-axis). The green crosses show the total number of injected patterns. 2d-SPADE does not detect all injected STPs, while 3d-SPADE does. Bottom panel: p-value spectra for each method, shown for pattern size $z = 3$ and different number of occurrences c (y-axis) and pattern durations d (x-axis) for different analysis window sizes w (as titles at the top of the panels). The p-values are indicated by the color scale, the lighter the smaller the p-value (color bar on the right). Significant signatures are marked with a red dot. Analysis parameters are chosen as `binsize` = 1 ms, `alpha` = 0.05, `dither` = 15 ms, `n_surr` = 5000 surrogates. The complete code to reproduce this figure can be found at https://github.com/INM-6/SPADE_applications/tree/master/multiple_pattern_durations/code. To run the complete analysis (from generation of the artificial data up to plotting the figure) execute `Snakefile` discussed in Section 5.2.

we derive the p-value spectrum in such a case by use of the bootstrap approximation using surrogate data, as explained above.

It is worth highlighting the difference in meaning between the window length w and the pattern duration d. The first defines the maximal temporal duration of a pattern to be detected: any pattern of temporal duration $d \leq w$ is detected by FIM, and thus can then be tested for significance. The window duration w is an input parameter of the method. The pattern duration is, in contrast, an intrinsic property of the data which is not known for experimental data and thus results from the analysis.

To summarize, for 2d-SPADE longer patterns may not be detected as significant, since the total number of chance patterns up to w is large such that a particular pattern of a particular duration is drowned. In contrast, when the pattern duration d is considered as an additional feature in the pattern signature (3d-SPADE), also longer patterns are adequately detected.

4.2. Validation of 3d-SPADE

Next, we investigate the performance of 3d-SPADE for different firing rate profiles, in terms of false positives and false negatives. We define here false positives (FPs) as falsely detected significant patterns that were not identical to the ones that were injected, but either patterns composed purely of background spikes or patterns composed of injected pattern spikes combined with other spikes. False negative (FN) patterns are defined as injected patterns not detected as STPs. For this purpose, we again generate artificial data for which we know the ground truth.

For all data sets, we generate parallel spike trains of a certain background rate into which we inject repeating patterns. The injected patterns are composed of a sequence of z spikes (varied from 3 to 10) (into the first z out of N neurons) and inserted them c times (as well varied from 3 to 10). Thus, for each background rate profile (i.e. a data set) we have 64 different data sets in respect to the pattern size and occurrences. In order to possibly compare the results shown here for 3d-SPADE to the 2d-SPADE results presented in Quaglio et al. (2017) we insert the identical patterns, i.e. with temporal lags of $l = 5ms$ between each of the pattern spikes, as in the former study. The inserted patterns are inserted within the duration of the data ($T = 1$ s).

All data sets consist of $N = 100$ parallel Poisson spike trains, but they differ in the background firing rates. In the first data set, all neurons have a stationary firing rate of $\lambda = 25$ Hz. The second data set is composed of background firing rates that exhibit a sudden coherent rate jump from 10 Hz to 60 Hz in all neurons for a period of 100 ms. The third case comprises a heterogeneous data set where each neuron has a stationary rate but each of a different rate. The firing rates vary across the neurons from 5 Hz up to 25 Hz in equidistant steps. In the last data set, 5 groups of 20 neurons each exhibit short and simultaneous rate jumps (from 14 Hz to 100 Hz) for an interval of 5ms, sequentially with a delay of 5ms, from the first to the last (5th) group. The four data sets correspond to the four columns of Fig. 4 (stationary, coherence, heterogeneity, propagation, respectively). The first row of Fig. 4 shows sketches of the respective rate profiles. The second row shows for each data set an example raster plot. The rows below contain matrices of false positives (third row), false negatives (fourth row), and the maximum of the two for each signature (z, c, d), for varying z and c. The data are analyzed with 3d-SPADE, using bin size of $b = 1$ ms, a sliding window of $w = 50$ ms, and a significance level of $\alpha = 0.01$, with the Holmes–Bonferroni correction.

For deriving the FP rate (third row) and the FN rate (fourth row) the a matrix. For coher
case leading to FPs (Louis et al., 2010), the
Indeed, having a larger maximal firing rate (60 Hz), even only for a short time (100 ms), increases the number of chance occurrences of patterns and thereby requires larger number of patterns to occur to become significant. Thereby less FPs occur. This, on the other hand, causes slightly larger number of FNs.

Rate propagation (Fig. 4, right column) does not lead to an enlarged level of FPs as compared to the other data sets, although these short coherent rate increases augment the probability to get chance synchrony, and sequences of those. In fact, the surrogate method for generation the null hypothesis is well compensating for that.

On the other hand, the FN rate of the detection of the injected patterns is also generally low (between 0.001 and 1%). That means that most of the patterns that occur more than 4 times are correctly identified, i.e. with low FN rates.

In Quaglio et al. (2017) we studied the identical scenario for 2d-SPADE. Interestingly, the results of both analyses correspond strongly: both show a low FP rate and a low FN rate even for strongly co-varying background rates. Since in these data we only studied patterns of one specific duration, the disadvantages of 2d-SPADE discussed above (high FN rate for longer patterns) do not come into play here. In addition, the statistical performance agrees well for both methods, indicating that the Holmes–Bonferroni multiple testing correction is well chosen.

Finally, we observed a similarly good performance (high number of FN for low number of occurrences) for patterns of size 2, which we were able to observe in experimental data (Torre et al., 2016): this is not shown here to facilitate the comparison to Figs. 6 and 7 of Quaglio et al. (2017).

5. Software implementation

The SPADE method (2d and 3d) is available in the Electrophysiology Analysis Toolkit[3] (Elephant, RRID:SCR_003833), an open source Python library providing methods for the analysis of electrophysiological data (e.g., parallel spike data, local field potentials) that includes user documentation and an extensive repertoire of unit tests of the analysis methods. Elephant builds on the Neo library[4] (RRID:SCR_000634) (Garcia et al., 2014) which provides a generic data representation for electrophysiological data and supports a large number of file formats, thus facilitating the practical application of SPADE. For the usage of SPADE we refer to the Elephant documentation[5]. The `spade` module contains the main function `spade()` that performs the analysis. It takes as arguments a list of parallel spike train objects and a set of analysis parameters (e.g. the bin size, the length of the analysis window length, the significance threshold, 2d or 3d version of the method). The function returns the list of found patterns (either all the mined patterns, or only the significant patterns) and, optionally, the p-value spectrum and the list of non-significant signatures. Each element of the list of found patterns is a dictionary of all features identifying a pattern (neurons involved, number of occurrences, occurrence times, temporal lags between the pattern spikes and p-values).

To perform the pattern mining, by default SPADE employs a C++ implementation[6] of the FP-growth algorithm for FIM developed by

[3] http://python-elephant.org.
[4] http://neuralensemble.org/neo.
[5] https://elephant.readthedocs.io/en/latest/reference/spade.html.
[6] http://www.borgelt.net/pyfim.html.

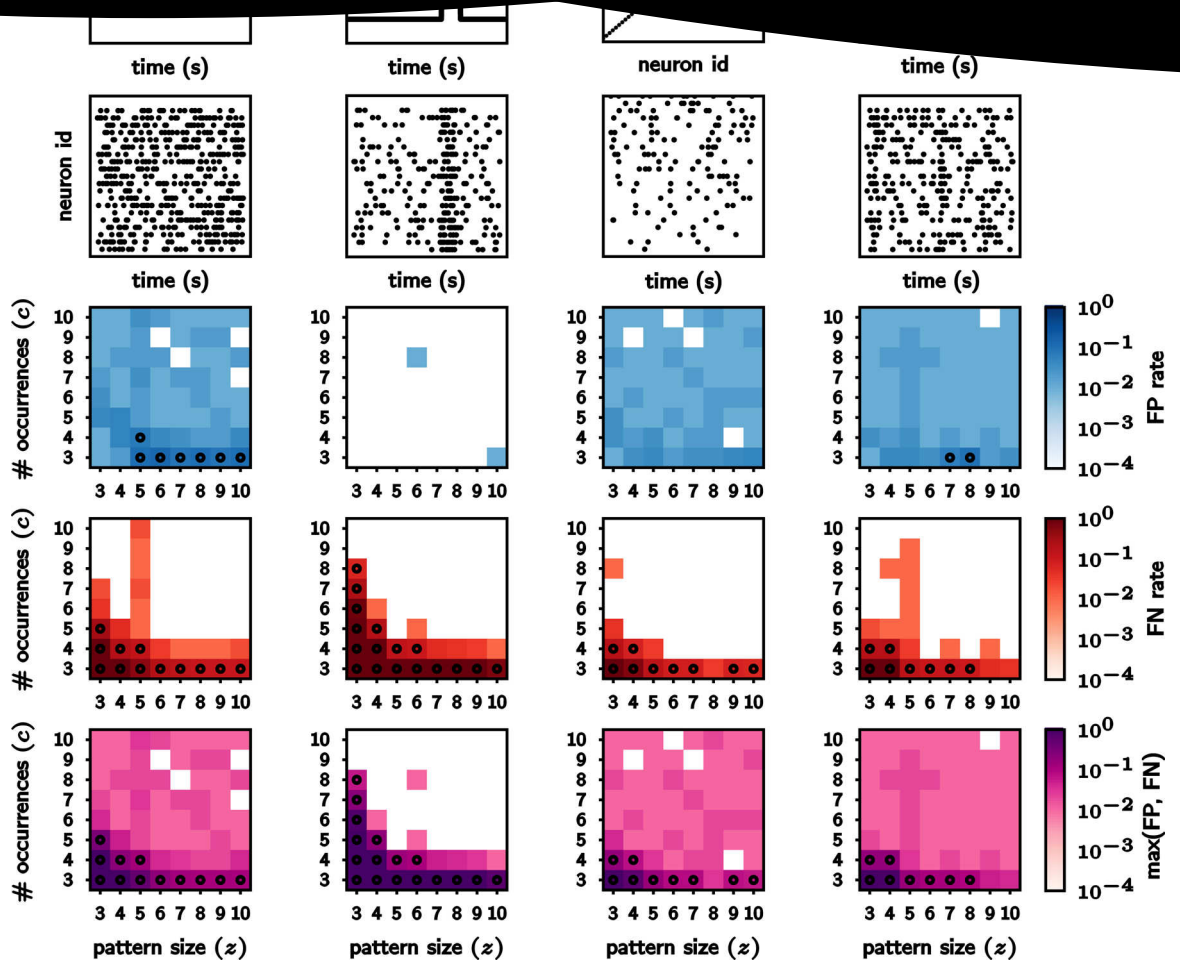

Fig. 4. Performance of 3d-SPADE in terms of FN and FP rates four types of artificial data sets. Top row: rate models for the background activity. From left to right: stationary constant firing rate (25 Hz) for all neurons, coherent rate changes of all neurons, rate heterogeneity across neurons, and rate jump propagation across successive groups of neurons. Second row: corresponding raster plots. Third and fourth row: FP and FN rates, respectively, obtained by injecting 50 ms-long patterns of fixed size z (horizontal axis) and number of occurrences c (vertical axis) into the background data. Bottom: maximum of FP and FN rate per signature. Black circles indicate entries in the matrices where the FP or FN rate is over 0.05 (arbitrary threshold). The code to reproduce this figure can be found at https://github.com/INM-6/SPADE_applications/tree/master/validation_FPFN/code. To run the complete analysis (from generation of the artificial data, performing the SPADE analysis up to plotting this figure) execute the `Snakefile` contained therein.

Borgelt (2012). Alternatively, a Python-based implementation of the FCA algorithm for FIM can be selected as the mining algorithm (Fast FCA by Lindig (2000)), which can be used in case of machine incompatibility of the external FIM C++ module. The implementation of SPADE also supports parallel computer environments by executing the script calling the `spade()` function using Open MPI[7]. Then `spade()` distributes the pattern extraction from the surrogate data for computing the p-value spectrum on all available cores.

The listing in Fig. 5 shows an example Python code demonstrating the use of the `spade()` function from Elephant in order to run the analysis with a data set of parallel spike trains. In this minimal example we analyze 10 independent Poisson spike trains, generated by employing the Elephant library.

5.1. Profiling of the computational performance

For profiling the SPADE code we register the compute times for some components of SPADE of which we know that they are time intensive. Also we aim to test if there is an increase of the compute time due to the use of 3d-SPADE. We show the compute times of the different implementations of the FIM algorithm (FP-growth in C++ and Fast-FCA in Python), but also for the complete SPADE analysis, i.e. 2d- and 3d-SPADE, each with the two FIM implementations. For these six cases we profiled the compute time under variation of three typical data parameters, i.e. firing rates, total duration of the data, number of parallel spike trains. For doing that we analyze simulated independent Poisson spike trains (no patterns injected) of different parameter constellations:

- Variation of the firing rates λ from 15 Hz to 75 Hz for all $N = 100$ spike trains of duration of $T = 3$ s (Fig. 6, left column).
- Variation of the duration T from 3 s to 15 s of $N = 100$ parallel spike

[7] https://www.open-mpi.org.

```
    rate=10*pq.Hz,  t_stop=10*pq.s) for i in range(10)]

# Mining patterns with SPADE using a binsize of 1 ms and a window length of 1
# bin (i.e., detecting only synchronous patterns).
SPADE_results = spade.spade(
        data=sts, binsize=1*pq.ms, winlen=1, dither=5*pq.ms, n_surr=1000,
   psr_param=[0,0,3],
        output_format='patterns')

# Printing SPADE output
print(SPADE_results)
```

Fig. 5. Listing of the SPADE call using the toolbox. Minimal example of Python code to generate independent Poisson processes using Elephant, import the `spade()` function from Elephant, set the relevant parameters and analyze the generated data set.

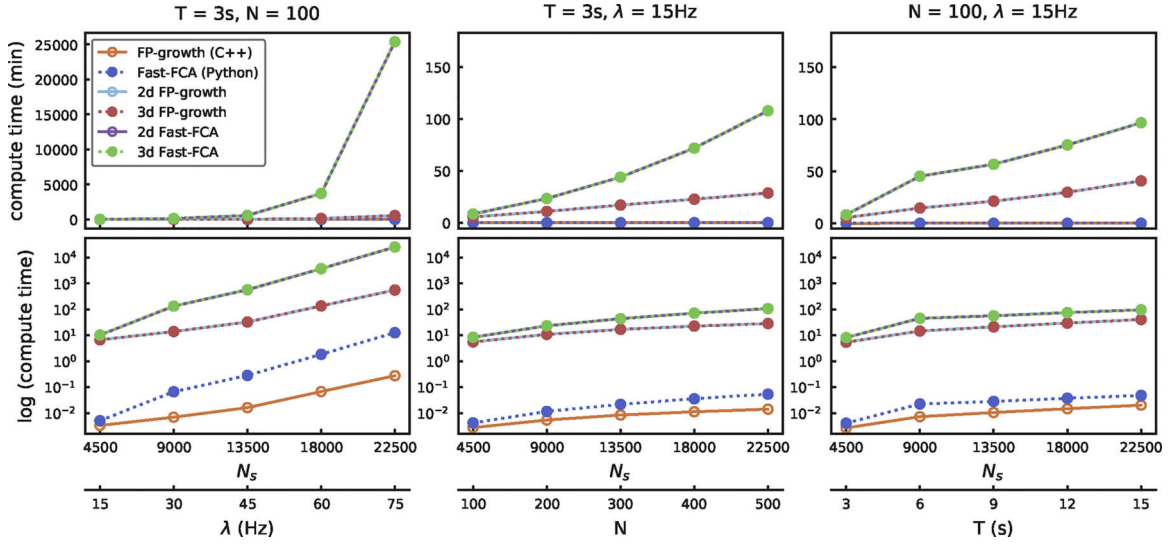

Fig. 6. Profiling results for different components of SPADE. Run times as a function of the number of spikes N_s by varying (as shown in the respective second x-axis): (1) the firing rates λ of the neurons (left panel, fixed number of neurons $N = 100$ and duration $T = 3$ s of the data sets), (2) the duration T of the data (central panel, constant rate $\lambda = 15$ Hz and $N = 100$) and (3) the number of parallel spike trains N (right panel, $\lambda = 15$ Hz and $T = 3$ s). The profiling times for pattern mining using the Python (FCA) implementation are shown in blue and for the C++ (FP-growth) implementation in orange. The profiling times of the PSF using FCA is shown for the 2d in purple and the 3d in green, lying on top of each other. Similarly, the profiling times of the PSF using FP-growth for the 2d in light blue and the 3d in red, again lying on top of each other. At the bottom, the same results are represented with logarithmic time axis. The complete code to reproduce this figure can be found at https://github.com/INM-6/SPADE_applications/tree/master/validation_FPFN/code. In order to run the complete profiling (from generation of the artificial data, performing the SPADE analysis to plotting the figure) one has to execute the `Snakefile` contained therein.

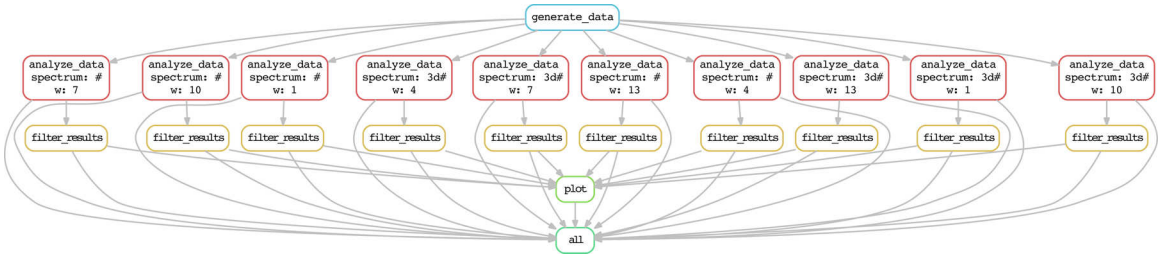

Fig. 7. Graph of the workflow used to generate Fig. 3 using Snakemake. Diagram as generated by Snakemake from the corresponding initial run of the workflow. Each node of the graph corresponds to one compute step (corresponding to a Python script and a set of parameters). Top node: generation of an artificial data set ('generate_data'). Second row: SPADE analysis ('analyze_data'), called 10 times with different parameters ('#' for 2d-SPADE, '3d#' for 3d-SPADE; window lengths 'w 1', 'w 4', 'w 7', 'w 10', 'w 13'). Third row: PSF and PSR analysis ('filter_results'). Fourth row: results are merged and the figure is generated ('plot'). Bottom row: 'all' saves all results.

sets varied in the same range from N_s = 4500 to 22,500 (Fig. 6, top *x*-axis). The variation of the respective other parameter is shown in the lower *x*-axis.

We find, that the Python implementation for pattern extraction (using FCA) is significantly slower than the C++ implementation (using FP-Growth) (Fig. 6, bottom, blue and orange, respectively). The bootstrap procedure used for PSF accounts for most of the compute time, since here the mining step is repeated as often as the requested number of surrogates (here 2000). However, this step can be trivially parallelized by the Elephant implementation of SPADE as explained above.

The difference of compute times of 2d-SPADE and 3d-SPADE are negligible (see Fig. 6, 2d and 3d versions for FP-growth (light blue and red) and for Fast-FCA (purple and green)) – the plots lie on top of each other. This is due to the fact that for both cases all patterns are extracted from the surrogates, and then they are grouped into the 2d- or 3d-spectrum. Thus, for both methods the mining of all closed frequent item sets from the surrogate data requires the same compute time. The time required for grouping into the 2d- or 3d-matrix is in both cases linear with respect to the number of mined sets.

All these observations hold for the three different data sets, where we varied alternatively the firing rates, the duration and the number of neurons. Noticeably, while varying the number of neurons or the duration of the data sets the computational time grows comparably (Fig. 6, middle column and right column, respectively). However, on the other hand, increasing the firing rate requires a much longer compute time to run SPADE, in particular for the Fast-FCA (Python) implementation. This due to the fact that in each of the sliding windows there are more spikes, and thus enhance the compute time for extracting repeated patterns.

5.2. Workflow description

For generation of the artificial data, its analysis and the generation of the figures of this manuscript, we used the workflow management system Snakemake[8]. The implementation of a Snakemake workflow enables us to automatically run and distribute the processes on the available cores for the complete analysis (for the given range of parameters to scan), and to combine the results afterwards into a single plot. See Fig. 7 as an illustration of an example workflow (shown as a directed acyclic graph (DAG)) used to generate Fig. 3. The data generation step is shown at the top of the graph ('generate data'). The individual analysis runs per parameter ('analyze_data spectrum') are then distributed by the Snakemake system to 10 parallel processes, 5 of which compute the 2d-pattern spectrum (argument for calling the SPADE function indicated as '#') and 5 others using 3d-pattern spectrum (argument indicated as '3d#'). The 5 runs for the 2d and the 3d version compute this step additionally for different window lengths w (indicated as 'w' and a number corresponding to its value for that run). After that follows for each line of computation the PSF and PSR step ('filter_results'), and finally all results are merged for the generation of the result figure ('plot'). In a last step, all results (of the pattern spectrum, PSF and PSR filtering and the merging steps) are collected and saved ('all').

The code necessary to reproduce the applications presented here are publicly available online.[9]

[8] https://snakemake.readthedocs.io.

[9] https://github.com/INM-6/SPADE_applications.

number of neurons
and newly also its temporal duration. This new
also patterns with long durations robustly, which would be lost if long and short patterns are evaluated together as done in 2d-SPADE.

Due to the increase of one dimension of the signature in 3d-SPADE, the multiple testing problem is enhanced. However, by using the Holmes-Bonferroni correction instead of FDR (as in 2d-SPADE) the performance with respect to the FN and FP rates is still acceptable (below 5%), also in case of rate profiles that tend to lead to false detections. Interestingly, the performance in terms of compute time of 3d- in comparison to 2d-SPADE is not enhanced, but is still solely determined by the implementation (C++ vs. Python) of the FIM analysis, since FIM has to be applied numerous times in order to compute the null hypothesis through surrogates. The compute time increases roughly with the number of spikes in the data set, but has slightly different behavior for increasing firing rates, total time of the data set, or for larger number of parallel spike trains. Using the C++ implementation of FIM, at least 500 parallel neurons can be analyzed for spatio-temporal patterns. The compute time can be considerably decreased by parallelization – the more cores the faster.

To support reproducibility of research the software codes and their respective Snakemake workflows are available for all results shown here.

Conflict of interest

None declared.

Acknowledgements

We thank Dr. Michael Denker and Alper Yegenoglu for the support in integrating SPADE into Elephant. Funding was received from European Union's Horizon 2020 Framework Programme for Research and Innovation under Specific Grant Agreement No. 785907 (Human Brain Project SGA2), Deutsche Forschungsgemeinschaft Grants GR 1753/4-2 and DE 2175/2-1 of the Priority Program (SPP 1665) and RTG2416 MultiSenses-MultiScales.

References

Abeles, M., 1982. Role of the cortical neuron: integrator or coincidence detector? Isr. J. Med. Sci. 18 (1), 83–92.

Agrawal, R., Imieliński, T., Swami, A., 1993. Mining association rules between sets of items in large databases. In: ACM. ACM Sigmod Record, vol. 22. pp. 207–216.

Bender, V.A., Bender, K.J., Brasier, D.J., Feldman, D.E., 2006. Two coincidence detectors for spike timing-dependent plasticity in somatosensory cortex. J. Neurosci. 26 (16), 4166–4177.

Benjamini, Y., Hochberg, Y., 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Stat. Soc. Ser. B Methodol. 289–300.

Bienenstock, E., 1995. A model of neocortex. Network: Comput. Neural Syst. 6 (2), 179–224.

Borgelt, C., 2012. Frequent item set mining. Wiley Interdisciplinary Reviews (WIREs): Data Mining and Knowledge Discovery, vol. 2. J. Wiley & Sons, Chichester, United Kingdom, pp. 437–456. https://doi.org/10.1002/widm.1074.

Brette, R., 2015. Philosophy of the spike: rate-based vs. spike-based theories of the brain. Front. Syst. Neurosci. 9, 151.

Buzsáki, G., 2004. Large-scale recording of neuronal ensembles. Nat. Neurosci. 7 (5), 446.

Date, A., Bienenstock, E., Geman, S., 1999. A statistical technique for detection of fine temporal structure in multi neuronal spike trains. Soc. Neurosci. Abstr. 25, 1441.

Fino, E., Paille, V., Cui, Y., Morera-Herreras, T., Deniau, J.-M., Venance, L., 2010. Distinct coincidence detectors govern the corticostriatal spike timing-dependent plasticity. J. Physiol. 588 (16), 3045–3062.

Garcia, S., Guarino, D., Jaillet, F., Jennings, T.R., Pröpper, R., Rautenberg, P.L., Rodgers, C., Sobolev, A., Wachtler, T., Yger, P., et al., 2014. Neo: an object model for handling electrophysiology data in multiple formats. Front. Neuroinform. 8, 10.

65–70.
Izhikevich, E.M., 2006. Polychronization: computation with spikes. Neural Comput. 18 (2), 245–282.
Kilavik, B.E., Roux, S., Ponce-Alvarez, A., Confais, J., Grün, S., Riehle, A., 2009. Long-term modifications in motor cortical dynamics induced by intensive practice. J. Neurosci. 29 (40), 12653–12663.
König, P., 1994. A method for the quantification of synchrony and oscillatory properties of neuronal activity. J. Neurosci. Methods 54 (1), 31–37.
Lindig, C., 2000. Fast concept analysis. Working with Conceptual Structures – Contributions to ICCS 2000, 152–161.
Louis, S., Gerstein, G.L., Grün, S., Diesmann, M., 2010. Surrogate spike train generation through dithering in operational time. Front. Comput. Neurosci. 4 (127).
Nádasdy, Z., Hirase, H., Czurkó, A., Csicsvari, J., Buzsáki, G., 1999. Replay and time compression of recurring spike sequences in the hippocampus. J. Neurosci. 19 (21), 9497–9507.
Pazienti, A., Maldonado, P.E., Diesmann, M., Grün, S., 2008. Effectiveness of systematic spike dithering depends on the precision of cortical synchronization. Brain Res. 1225, 39–46.
Picado-Muiño, D., Borgelt, C., Berger, D., Gerstein, G.L., Grün, S., 2013. Finding neural assemblies with frequent item set mining. Front. Neuroinform. 7, 9.
Prut, Y., Vaadia, E., Bergman, H., Haalman, I., Slovin, H., Abeles, M., 1998. Spatiotemporal structure of cortical activity: properties and behavioral relevance. J. Neurophysiol. 79 (6), 2857–2874.
Quaglio, P., Yegenoglu, A., Torre, E., Endres, D.M., Grün, S., 2017. Detection and evaluation of spatio-temporal spike patterns in massively parallel spike train data with spade. Front. Comput. Neurosci. 11, 41.
Riehle, A., Grün, S., Diesmann, M., Aertsen, A., 1997. Spike synchronization and rate
Shimazaki, H., Amari, S.,
varying higher-order spike correlation for
Comput. Biol. 8 (3), e1002385.
Steinmetz, N.A., Koch, C., Harris, K.D., Carandini, M., 2018. Challenges and opportunities for large-scale electrophysiology with neuropixels probes. Curr. Opin. Neurobiol. 50, 92–100.
Swadlow, H.A., 1988. Efferent neurons and suspected interneurons in binocular visual cortex of the awake rabbit: receptive fields and binocular properties. J. Neurophysiol. 59 (4), 1162–1187.
Swadlow, H.A., 1994. Efferent neurons and suspected interneurons in motor cortex of the awake rabbit: axonal properties, sensory receptive fields, and subthreshold synaptic inputs. J. Neurophysiol. 71 (2), 437–453.
Swadlow, H.A., 1998. Neocortical efferent neurons with very slowly conducting axons: strategies for reliable antidromic identification. J. Neurosci. Methods 79, 131–141.
Torre, E., Picado-Mui no, D., Denker, M., Borgelt, C., Grün, S., 2013. Statistical evaluation of synchronous spike patterns extracted by frequent item set mining. Front. Comput. Neurosci. 7, 132.
Torre, E., Quaglio, P., Denker, M., Brochier, T., Riehle, A., Grün, S., 2016. Synchronous spike patterns in macaque motor cortex during an instructed-delay reach-to-grasp task. J. Neurosci. 36 (32), 8329–8340.
Wille, R., 1982. Restructuring lattice theory: an approach based on hierarchies of concepts. Ordered Sets. Springer, pp. 445–470.
Yegenoglu, A., Quaglio, P., Torre, E., Grün, S., Endres, D., 2016. Exploring the usefulness of formal concept analysis for robust detection of spatio-temporal spike patterns in massively parallel spike trains. In: International Conference on Conceptual Structures. Springer. pp. 3–16.