‘*SpikeNburst*’ and ‘*Nicespike*’: Advanced Tools for Enhancing and Accelerating *In Vitro* High-Density Electrophysiology Analysis

Robert Wolff[1,*], Alessia Polito[1,2,*], Alessio Paolo Buccino[3], Michela Chiappalone[4], Valter Tucci[1,#]

[1] Genetics and Epigenetics of Behavior (GEB) Laboratory, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163 Genova, Italy

[2] Padova Neuroscience Center (PNC), University of Padova, Via Orus 2/B, 35129 Padova, Italy

[3] Allen Institute for Neural Dynamics, 615 Westlake Ave, Seattle, WA 98109, United States

[4] Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS), University of Genova, Via all’Opera Pia 13A, 16145 Genova, Italy

[#] Corresponding author: valter.tucci@iit.it

[*] These authors contributed equally.

Abstract

High-density multi-electrode arrays (HD-MEAs) have revolutionized electrophysiology by enabling the recording of neuronal activity with unprecedented spatial and temporal resolution. However, analysing these large-scale datasets poses significant challenges, including artefact removal, spike sorting, and accurate assessments of neuronal synchronization. Here, we present two Python-based tools, ‘*spikeNburst*’ and ‘*nicespike*’, designed to address these challenges and provide a scalable solution for the comprehensive analysis of HD-MEA recordings.

The *spikeNburst* tool integrates advanced methodologies for spike train filtering, burst and network burst detection, and synchronization analysis, and we implemented a full analysis pipeline in the *nicespike* tool, which includes GPU-accelerated spike sorting using template matching with Kilosort to accurately identify neuronal units spanning multiple electrodes. Together, these tools enable more precise analyses by mitigating redundancy and overestimation inherent in single-channel approaches. Their graphical user and command-line interfaces ensure accessibility for diverse user needs.

We validated the tools on neuronal culture recordings, demonstrating their ability to identify somatic and dendritic features of neuronal units, characterize bursting behaviour, and quantify synchronization at both unit and network levels. By addressing critical limitations of existing methods, *spikeNburst* and *nicespike* provide a robust, scalable, and user-friendly framework for HD-MEA data analysis, advancing our ability to study neural network dynamics and single-cell activity in detail.

Introduction

Continuous advancements in high-dimensional electrophysiological recording technologies, particularly *in vitro* and *ex vivo* systems, have greatly enhanced our capacity to study neuronal activity with unprecedented spatial and temporal resolution. These systems facilitate the acquisition of large-scale neuronal datasets by capturing activity across thousands of electrodes, allowing for detailed analyses of both network dynamics and single-neuron activity. Despite these advancements, significant challenges persist, including artefact filtration, neuron-specific spike sorting, and the accurate assessment of synchronization. Furthermore, the increasing complexity and redundancy of signals in datasets generated by high-density systems necessitate the development of robust, scalable analytical tools to address these limitations and optimize data processing.

We specifically examined a commercially available high-density multi-electrode array (HD-MEA) system (3Brain), which comprises 4,096 electrodes arranged in a square grid with an inter-electrode spacing of 60 µm. Under these conditions, neuronal activity can be concurrently detected across multiple electrodes, capturing the spatial extent of individual neuronal signals.

Existing software solutions, including the default tools integrated within standard analytical platforms, such as the `BrainWave` software (3Brain, 2025), primarily provide fundamental spike detection functionalities but are restricted to single-channel analyses. This limitation often results in overestimation of neuronal units and network activity due to redundant measurements of neurons interacting across multiple electrodes. Furthermore, while effective for systems with few electrodes, legacy tools such as `Spycode` (Bologna et al., 2010) are neither scalable nor compatible with contemporary HD-MEA datasets.

This study addresses these gaps by presenting two Python-based tools, ‘`spikeNburst`’ and ‘`nicespike`’, designed for the advanced analysis of HD-MEA recordings. These tools integrate state-of-the-art spike sorting, burst detection, and synchrony measures within an accessible and flexible framework. By leveraging advanced computational methods and a user-friendly interface, they enable more accurate and comprehensive investigations of neuronal activity, bridging a critical need in electrophysiological research .

In this paper, we describe the implementation, features, and application of these tools, demonstrating their efficacy in overcoming the limitations of existing methods while providing a scalable solution for large-scale electrophysiological datasets (Polito, 2024).

Advanced and easy-to-use spike sorting and analysis

We implemented an analysis pipeline for the analysis of raw electrophysiological recordings from HD-MEA systems. Raw recordings are processed by the `nicespike` tool that ultimately plots and exports the analysis results for each recording using the `spikeNburst` tool (Figure 1A). Both tools are implemented with a graphical user interface (GUI) and can be accessed programmatically as python modules.

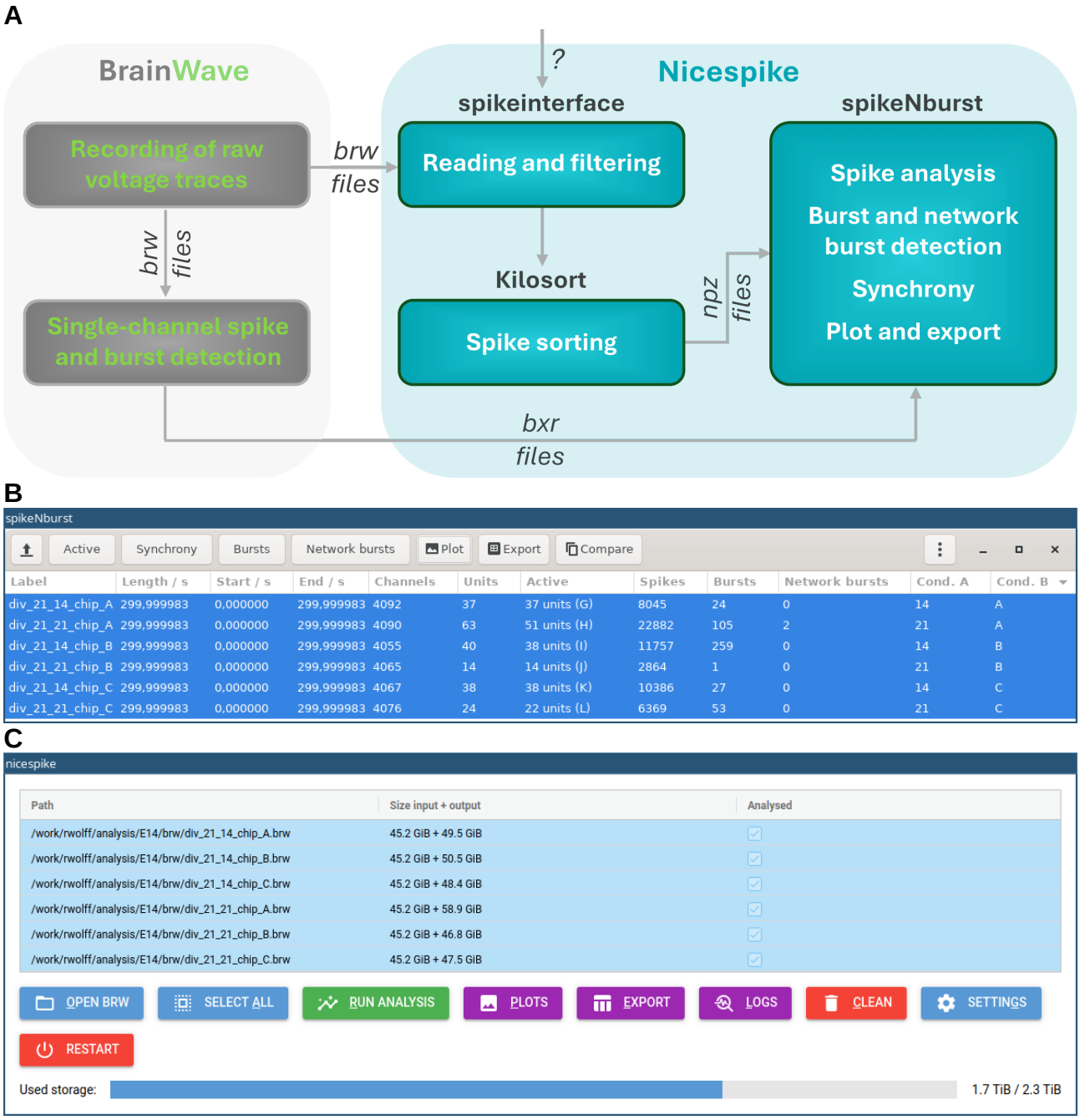


Figure 1: (A) Scheme of the analysis tools with the flow of the data. The arrow with '?' indicates possible extensions to analyse recordings from different HD-MEA systems. (B) `spikeNburst`'s GUI showing example recordings loaded and analysed. (C) `nicespike`'s GUI showing the same example recordings.

The `spikeNburst` tool was developed to provide accessible and advanced analysis of spike

trains. It supports input from both bxr files generated by the BrainWave software (3Brain) and custom npz files (NumPy data archive format, used internally by nicespike). The tool incorporates multiple analytical methods. First, neuronal units can be filtered using the inter-spike-interval (ISI) violation method (Hill et al., 2011). Second, methods for burst and network burst detection are implemented, ranging from basic approaches (Chiappalone et al., 2005) to advanced algorithms (Pasquale et al., 2010), enabling detailed investigation of bursting behaviour. Third, a variety of unit synchrony measures are available, including spike distance (Kreuz et al., 2013), the spike-time tiling coefficient (STTC) (Cutts and Eglen, 2014) and phase synchrony (Li et al., 2007; Patel et al., 2014, 2012), implemented using the burst_sync package (Eisenman et al., 2015). Parameters calculated by spikeNburst can be easily exported as tabular data for further analysis and visualization, and basic plotting functionality is included. The spikeNburst tool can be utilized both as a python module and a graphical user interface (GUI) (Figure 1B, Supplemental S1).

We implemented an analysis pipeline including spike sorting with template matching and subsequent analysis within the nicespike tool. We used the spikeinterface package (Buccino et al., 2020) for accessing the brw files. For this, we co-authored the brw file reading in the python-neo tool (Garcia et al., 2014). Firstly, the raw voltage traces of electrophysiology recordings are filtered. We implemented an automatic channel removal for broken or very noisy channels that removes signals if they have constant consecutive values for at least 10 samples or if their standard deviation (SD) is > 1.7 times the median of SD of all channels. Broken channels, determined from previous recordings, are removed, and the raw traces are band-pass-filtered from 300 to 6,000 Hz. Then, spike sorting is done using Kilosort, running on GPU (Pachitariu et al., 2016). The spike-sorted unit spike traces are then exported to custom npz files that are further analysed by the spikeNburst tool. The nicespike tool can be used as a python module or with a GUI (Figure 1C, Supplemental S2).

Neurons are recorded from multiple electrodes

We recorded raw voltage traces from wild-type (E14) cells at two (DIV 21+14) and three (DIV 21+21) weeks after the end of the differentiation (Methods). We analysed the raw recordings with the described tools using two types of spike detection: 1. the single-channel spike detection from BrainWave, and 2. spike-template matching with Kilosort within the nicespike tool. Subsequently, we performed spike, burst and synchrony measurement using spikeNburst. We observed that the mean firing rate was over-estimated by the single-channel spike detection by one order of magnitude (Figure 2A).

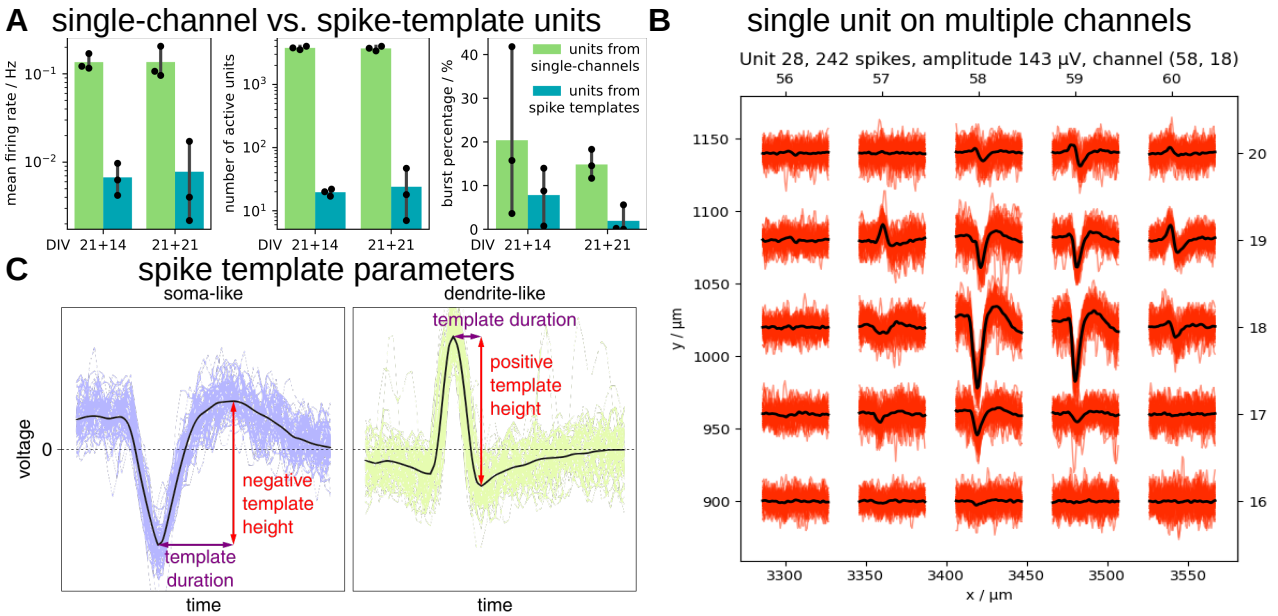


Figure 2: Comparison between single-channel spike detection (`BrainWave`) and advanced unit sorting based on spike template sorting (`Kilosort`). (A) Parameters of recordings in three recordings at DIV 21+14 and 21+21 using units from single channels and spike template sorting. (B) Parameters of recordings in three recordings at DIV 21+14 and 21+21 using units from single channels and spike template sorting. An example unit that has a matched template with somatic (negative voltage peak) and dendritic (positive voltage peak) parts spanning over at least 200 µm. (C) Estimate of spike template height and duration for soma-like and dendrite-like units.

Duplicates are expected for single-channel spike detection, because the activity of the same neuron may spread on multiple channels. By contrast, the sorting by spike-template matching characterized the neuron's spatial footprint, identifying the highest spiking activity of a single unit captured by a central channel and the traces of spike activity on the surrounding channels (Figure 2B). Also, the number of active units was over-estimated by the single-channel spike detection by two orders of magnitude (Figure 2A), and more bursts are detected for the units from single-channels than from spike-template matching (Figure 2A).

We classified units to be soma-like or dendrite-like depending on the spike template shape of the central channel that has the highest voltage difference (Figure 2B–C). Spike templates where the absolute minimum voltage is greater than the maximum voltage are classified as soma-like with an associated negative template height, while dendrite-like units have a positive template height (Bakkum et al., 2019).

Automated plotting and exporting with spikeNburst and nicespike

The analysis pipelines implemented both in `spikeNburst` and in `nicespike` include automated plotting and exporting. For each spike-sorted unit the templates are plotted for the central channel and surrounding channels within `nicespike` (Figure 2B). The plotting with `spikeNburst` includes histograms and spatial plots of many measured parameters (Figure 3).

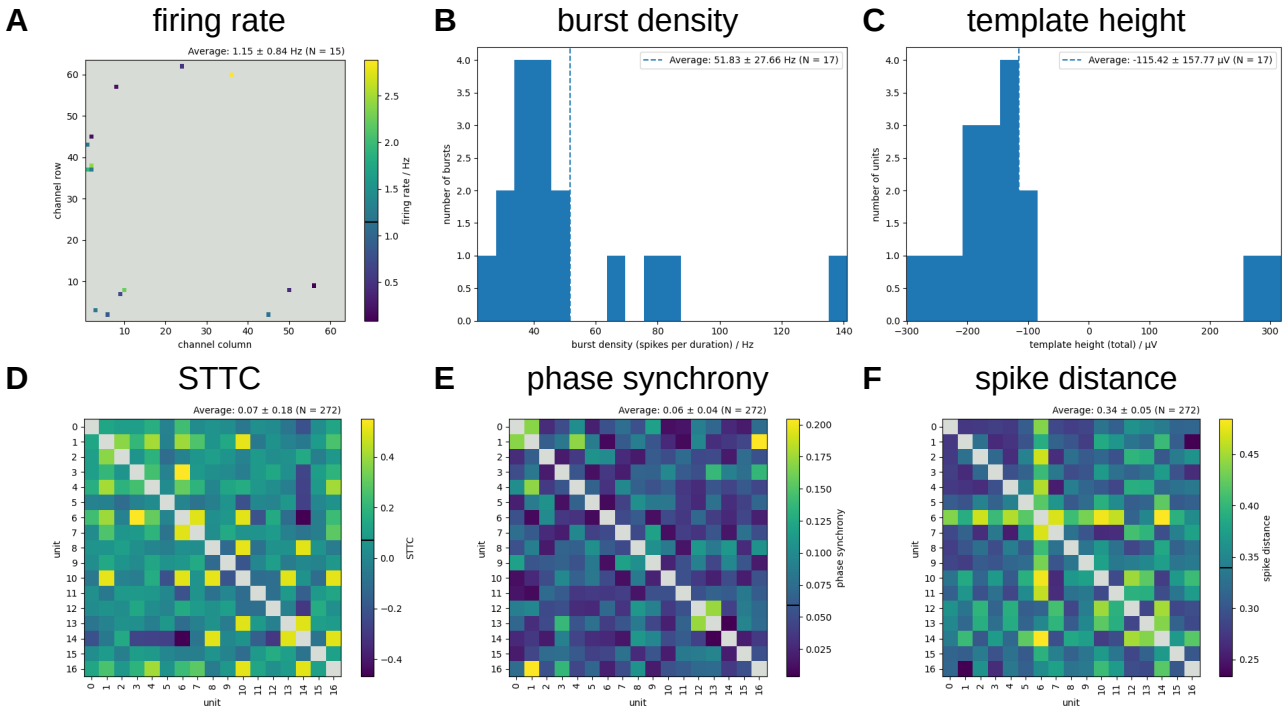


Figure 3: Characterization of one recording after spike template sorting using plots produced by `spikeNburst`. (A) Firing rate of units at their locations. (B) Histogram of burst density for 17 bursts. (C) Histogram of unit template height with 1 dendrite-like and 16 soma-like units. (D–F) Various synchrony measures for pairs of units: STTC (D), phase synchrony (E) and spike distance (F).

The firing rate of units depending on their location shows that few units have high firing rate of up to around 3 Hz (Figure 3A). The bursts have a mean burst density of 52 ± 28 spikes per burst (Figure 3B). Most found units were classified as soma-like and only one is found to be dendrite-like with a template height of about 300 µV (Figure 3C). Different synchrony measures have been calculated (Figure 3D–F). They have different meanings: the STTC and the phase synchrony are 0 for no synchrony and 1 for high synchrony, while the spike distance is 0 for highest synchrony and greater 0 for low synchrony. In particular, the phase synchrony and the spike distance show anti-correlation as expected. Overall, the example recording does not have high synchrony.

Conclusion

To address current challenges in HD-MEA technologies, we have developed `spikeNburst` and `nicespike`, which provide robust, scalable, and user-friendly solutions for the comprehensive analysis of HD-MEA recordings. The `spikeNburst` tool introduces advanced methodologies for spike train filtering, burst detection, and synchrony analysis, integrating established algorithms and metrics within a flexible framework. The `nicespike` tool complements these functionalities with GPU-accelerated spike sorting, leveraging template matching to accurately delineate neuronal units across multiple electrodes. Together, these tools overcome the limitations of single-channel analyses, minimizing overestimation of neuronal units and providing a more accurate representation of network activity.

Our analysis pipeline, combining `spikeNburst` and `nicespike`, was evaluated on datasets recorded from neuronal cultures, demonstrating its effectiveness in identifying somatic and dendritic features of single neurons and quantifying synchrony and burst dynamics at both unit and

network levels. The integration of command-line and graphical interfaces ensures accessibility for diverse research needs, from high-throughput batch analyses to individual experiment visualization. The current implementation was proven to work with electrophysiological recordings from one specific manufacturer, but because it is open-source software and it depends on spikeinterface that is able to read many different recording types, it is trivial to extend the functionality to other HD-MEA systems.

In summary, `spikeNburst` and `nicespike` represent a step forward in the field of *in vitro* and ex vivo electrophysiology, enabling researchers to efficiently analyse complex HD-MEA datasets and uncover detailed neuronal dynamics. By addressing critical gaps in current analytical approaches, these tools lay the groundwork for future innovations in the study of neural systems and their applications in neuroscience and bioengineering.

Methods

Sample preparation

We propagated mESC line E14Tg2a (E14) on laminin-coated dishes (Sigma) in DMEM (Dulbecco's Modified Eagle Medium, Invitrogen) supplemented with 15% ESC-certified FBS (Invitrogen), 0.1 mM nonessential amino acids (Invitrogen), 1 mM sodium pyruvate (Invitrogen), 0.1 mM b-mercaptoethanol (Sigma), 50 U/ml penicillin/streptomycin (Invitrogen), and 103 U/ml LIF (ESGRO Millipore) as previously described (Bouschet et al., 2017). All ESC lines were regularly controlled to exclude the presence of mycoplasma using the MycoSPY kit (Duotech/Biontex).

In vitro 2D corticogenesis was performed by adapting a protocol previously described (Gaspard et al., 2009). The day before the differentiation, the ESC line was plated at low density on laminin-coated dishes. The day after, the cells start to be cultured in chemically defined default medium DDM supplemented with B27-without vitamin A (Invitrogen). From day 0 to 8 of culture, the DDM medium was supplemented with the dorsomorphin analog DMH1-HCl at 1 µM (Tocris) instead of cyclopamine. At 12 days of 2D in vitro corticogenesis, E14 neuroprogenitor cells were plated at high density on 0.1 mg/ml PEI (Sigma)-33 µg/ml laminin (Sigma) coated Accura (3Brain) chips according to the manufacturer's instructions (3Brain), differentiated until 21 days (DIV 21), and maintained in neurobasal medium for 14 (DIV 21+14) and 21 (DIV 21+21) days for MEA recordings. The cells were cultured and differentiated at 37 °C under an atmosphere of 5% CO2.

Electrophysiological recordings with HD-MEA

We performed spontaneous electrophysiological activity recordings of the derived cultures using an HD-MEA system (BioCAM DupleX with `BrainWave`, version 4 or 5 software by 3Brain). The recording chips comprise 4,096 micro-electrodes with an active area of 3.8 mm × 3.8 mm. After 2 minutes to acclimate the culture in the system, 5 minutes of recording for each sample was performed at two (DIV 21+14) and three (DIV 21+21) weeks after the end of the differentiation. Raw voltage traces are recorded using the default `BrainWave` software, which generates `brw` files (HDF5-based format containing raw data). These files can be further processed within `BrainWave` to perform single-channel spike and burst detection, resulting in `bxr` files (another HDF5-based format containing analysis data). The `bxr` files can subsequently be analysed using the `spikeNburst` tool for advanced analyses. Alternatively, raw voltage traces from the `brw` files can be processed using the `nicespike` tool, which integrates filtering, spike sorting, and

subsequent analysis with the `spikeNburst` tool.

SpikeNburst

The `spikeNburst` package is developed for `python` ≥ 3.8. With `python` 3.12.6, for reading and exporting the modules `h5py` (3.11.0), `odfpy` (1.4.2), `openpyxl` (3.1.5) are used. The analysis and plotting is done using the `matplotlib` (3.9.2), `numpy` (2.0.1), `pandas` (2.2.2), and `scipy` (1.14.1) modules. The GUI is implemented using `GTK` 3.0 interfaced with `pygobject` (3.48.2). The synchrony measures are calculated using the python module `burst_sync` (2.0.1) (Eisenman et al., 2015) with its dependency on `cython` (3.0.11). It is able to run on `bxr` files in version 2.x from `BrainWave` 3–4 (3Brain, 2021) and custom `npz` files created from running `spikeinterface` within `nicespike`.

Nicespike

The `nicespike` package is developed for `python` ≥ 3.8. With `python` 3.11.7, it runs with the modules `nicegui` (1.4.23), `numpy` (1.26.4), `spikeinterface` (0.100.0), `python-neo` (0.13.1), and `spikeNburst` (0.2.0). It is able to run on `brw` files in versions 3.x from `BrainWave` 3–4 (3Brain, 2021) and 4.x from `BrainWave` 5 (3Brain, 2023).

Code availability

The source code is available at codeberg.org/spiky/spikenburst and codeberg.org/spiky/nicespike.

References

3Brain, 2025. BrainWave 5 [WWW Document]. https://www.3brain.com/products/software/brainwave5 (accessed 01/02/2025)

3Brain, 2023. File Format Documentation For BRW v4.x, BXR v3.x and BCMP v1.x, Version 1.1.3. https://gin.g-node.org/NeuralEnsemble/ephy_testing_data/src/master/biocam/documentation_brw_4.x_bxr_3.x_bcmp_1.x_in_brainwave_5.x_v1.1.3.pdf

3Brain, 2021. Documentation for BRW v. 3.x and BXR v. 2.x File Formats, Version 1.2.0. https://gin.g-node.org/NeuralEnsemble/ephy_testing_data/src/master/biocam/documentation_brw_3.x_bxr_2.x_in_brainwave_3.x_and_4.x_v1.2.0.pdf

Bakkum, D.J., Obien, M.E.J., Radivojevic, M., Jäckel, D., Frey, U., Takahashi, H., Hierlemann, A., 2019. The Axon Initial Segment is the Dominant Contributor to the Neuron's Extracellular Electrical Potential Landscape. Adv. Biosyst. 3, 1800308. https://doi.org/10.1002/adbi.201800308

Bologna, L.L., Pasquale, V., Garofalo, M., Gandolfo, M., Baljon, P.L., Maccione, A., Martinoia, S., Chiappalone, M., 2010. Investigating neuronal activity by SPYCODE multi-channel data analyzer. Neural Netw. 23, 685–697. https://doi.org/10.1016/j.neunet.2010.05.002

Bouschet, T., Dubois, E., Reynès, C., Kota, S.K., Rialle, S., Maupetit-Méhouas, S., Pezet, M., Le Digarcher, A., Nidelet, S., Demolombe, V., Cavelier, P., Meusnier, C., Maurizy, C., Sabatier, R., Feil, R., Arnaud, P., Journot, L., Varrault, A., 2017. In Vitro Corticogenesis from Embryonic Stem Cells Recapitulates the In Vivo Epigenetic Control of Imprinted Gene Expression. Cereb. Cortex 27, 2418–2433. https://doi.org/10.1093/cercor/bhw102

Buccino, A.P., Hurwitz, C.L., Garcia, S., Magland, J., Siegle, J.H., Hurwitz, R., Hennig, M.H., 2020. SpikeInterface, a unified framework for spike sorting. eLife 9, e61834. https://doi.org/10.7554/eLife.61834

Chiappalone, M., Novellino, A., Vajda, I., Vato, A., Martinoia, S., van Pelt, J., 2005. Burst detection algorithms for the analysis of spatio-temporal patterns in cortical networks of neurons. Neurocomputing, Computational Neuroscience: Trends in Research 2005 65–66,

653–662. https://doi.org/10.1016/j.neucom.2004.10.094
Cutts, C.S., Eglen, S.J., 2014. Detecting pairwise correlations in spike trains: an objective comparison of methods and application to the study of retinal waves. J. Neurosci. Off. J. Soc. Neurosci. 34, 14288–14303. https://doi.org/10.1523/JNEUROSCI.2767-14.2014
Eisenman, L.N., Emnett, C.M., Mohan, J., Zorumski, C.F., Mennerick, S., 2015. Quantification of bursting and synchrony in cultured hippocampal neurons. J. Neurophysiol. 114, 1059–1071. https://doi.org/10.1152/jn.00079.2015
Garcia, S., Guarino, D., Jaillet, F., Jennings, T.R., Pröpper, R., Rautenberg, P.L., Rodgers, C., Sobolev, A., Wachtler, T., Yger, P., Davison, A.P., 2014. Neo: an object model for handling electrophysiology data in multiple formats. Front. Neuroinformatics 8. https://doi.org/10.3389/fninf.2014.00010
Gaspard, N., Bouschet, T., Herpoel, A., Naeije, G., van den Ameele, J., Vanderhaeghen, P., 2009. Generation of cortical neurons from mouse embryonic stem cells. Nat. Protoc. 4, 1454–1463. https://doi.org/10.1038/nprot.2009.157
Hill, D.N., Mehta, S.B., Kleinfeld, D., 2011. Quality Metrics to Accompany Spike Sorting of Extracellular Signals. J. Neurosci. 31, 8699–8705. https://doi.org/10.1523/jneurosci.0971-11.2011
Kreuz, T., Chicharro, D., Houghton, C., Andrzejak, R.G., Mormann, F., 2013. Monitoring spike train synchrony. J. Neurophysiol. 109, 1457–1472. https://doi.org/10.1152/jn.00873.2012
Li, X., Cui, D., Jiruska, P., Fox, J.E., Yao, X., Jefferys, J.G.R., 2007. Synchronization Measurement of Multiple Neuronal Populations. J. Neurophysiol. 98, 3341–3348. https://doi.org/10.1152/jn.00977.2007
Pachitariu, M., Steinmetz, N.A., Kadir, S.N., Carandini, M., Harris, K.D., 2016. Fast and accurate spike sorting of high-channel count probes with KiloSort, in: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc. https://papers.nips.cc/paper/2016/hash/1145a30ff80745b56fb0cecf65305017-Abstract.html
Pasquale, V., Martinoia, S., Chiappalone, M., 2010. A self-adapting approach for the detection of bursts and network bursts in neuronal cultures. J. Comput. Neurosci. 29, 213–229. https://doi.org/10.1007/s10827-009-0175-1
Patel, T.P., Ventre, S.C., Geddes-Klein, D., Singh, P.K., Meaney, D.F., 2014. Single-Neuron NMDA Receptor Phenotype Influences Neuronal Rewiring and Reintegration following Traumatic Injury. J. Neurosci. 34, 4200–4213. https://doi.org/10.1523/JNEUROSCI.4172-13.2014
Patel, T.P., Ventre, S.C., Meaney, D.F., 2012. Dynamic Changes in Neural Circuit Topology Following Mild Mechanical Injury In Vitro. Ann. Biomed. Eng. 40, 23–36. https://doi.org/10.1007/s10439-011-0390-6
Polito, A., 2024. The role of maternal and paternal genomes in neuronal networks (PhD thesis). Università degli studi di Padova, Padova, Italy. https://hdl.handle.net/11577/3516146

Supplemental S1 – Manual spikeNburst (version 0.2)

Installation

The `spikeNburst` tool (https://codeberg.org/spiky/spikenburst) can be installed on Linux or WSL (for Windows). The recommended way to install the dependencies is to create a new Anaconda environment with:

```
conda create -n spikenburst cython h5py librsvg matplotlib
    numpy openpyxl pandas pip python scipy conda-forge::gtk3
    conda-forge::odfpy conda-forge::pygobject
conda activate spikenburst
```

Then, `spikeNburst` can be installed with:

```
pip install --extra-index-url
    https://codeberg.org/api/packages/spiky/pypi/simple/
    spikenburst[full]
```

Alternative ways to install `spikeNburst` are documented in the README.

Usage of the python module

There are two options how you can use this tool. The first one is the `spikeNburst` python module which can be used inside python code with

```
import spikenburst

sa = spikenburst.SpikeAnalysis('npz/div_21_21_chip_A.npz',
    use_spike_sorted_units=True, min_firing_rate=0.05)
sttc = sa.sttc(dt=0.5)
phase_synchrony = sa.phase_synchrony()
spike_distance = sa.spike_distance(n_bin=120)
ba = sa.burst_detection('pasquale')
nba = ba.network_burst_detection('pasquale', min_n_entity=5)
sa.plot('ouput/div_21_21_chip_A/s_')
ba.plot('ouput/div_21_21_chip_A/b_')
nba.plot('ouput/div_21_21_chip_A/n_')
nba.export('ouput/div_21_21_chip_A.ods')
```

to run the spike analysis, calculate different synchrony measures, detect bursts and network bursts, and finally, plot and export the analysis results.

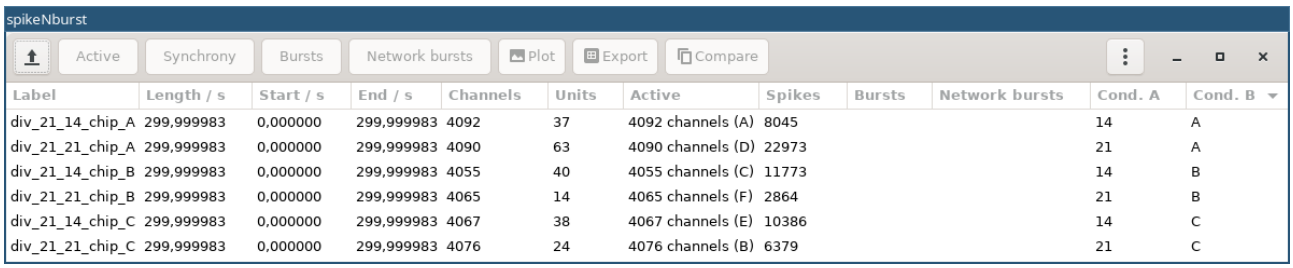
Usage of the graphical user interface

Start GUI and load input data

Start the graphical user interface of `spikeNburst` by

```
spikenburst [input]
```

with optionally input files or folders given as input. Alternatively, input files can be opened clicking on the top left button in the appearing window. The appearing window lists the loaded recordings with labels and some basic parameters (Figure 4). Conditions A and B are determined from the file names if possible (in the example below DIV and chip).



Label	Length / s	Start / s	End / s	Channels	Units	Active	Spikes	Bursts	Network bursts	Cond. A	Cond. B
div_21_14_chip_A	299,999983	0,000000	299,999983	4092	37	4092 channels (A)	8045			14	A
div_21_21_chip_A	299,999983	0,000000	299,999983	4090	63	4090 channels (D)	22973			21	A
div_21_14_chip_B	299,999983	0,000000	299,999983	4055	40	4055 channels (C)	11773			14	B
div_21_21_chip_B	299,999983	0,000000	299,999983	4065	14	4065 channels (F)	2864			21	B
div_21_14_chip_C	299,999983	0,000000	299,999983	4067	38	4067 channels (E)	10386			14	C
div_21_21_chip_C	299,999983	0,000000	299,999983	4076	24	4076 channels (B)	6379			21	C

Figure 4: Interface of `spikeNburst` after loading the input recordings.

Determine active units, count spikes

Before any analysis can be done, recording(s) must be selected from the list by mouse click. After clicking on the button ‘Active’, which is now not greyed out any more, a window shows up, in which different settings for the active units detection can be set, as which units to use, minimum firing rate, etc. (Figure 5A). Mouse hovering over the parameter settings raise pop-ups with further explanation (Figure 5B). After clicking on ‘OK’ the columns of ‘Active’ and ‘Spikes’ are updated to include only the chosen active units or channels.

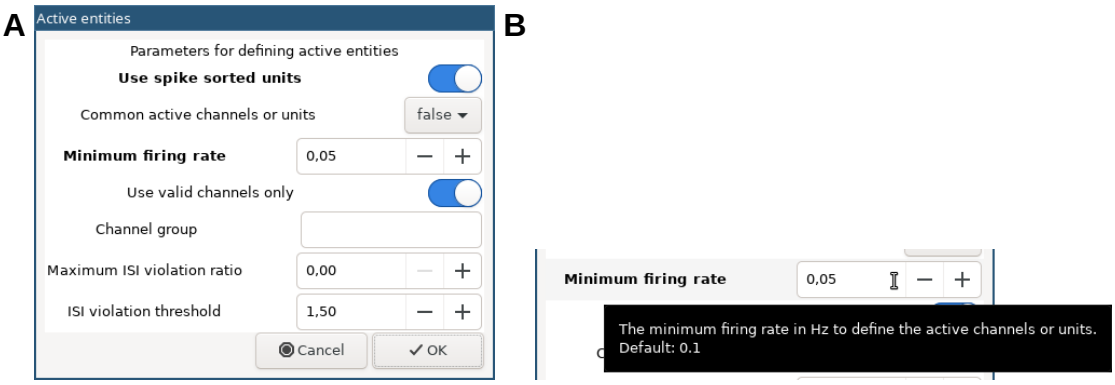


Figure 5: (A) active channel or unit settings. (B) parameter description on mouse hovering.

Calculate synchrony measures

Clicking on the button ‘Synchrony’ leads to a dialog box where the method and input parameters must be selected (Figure 6).

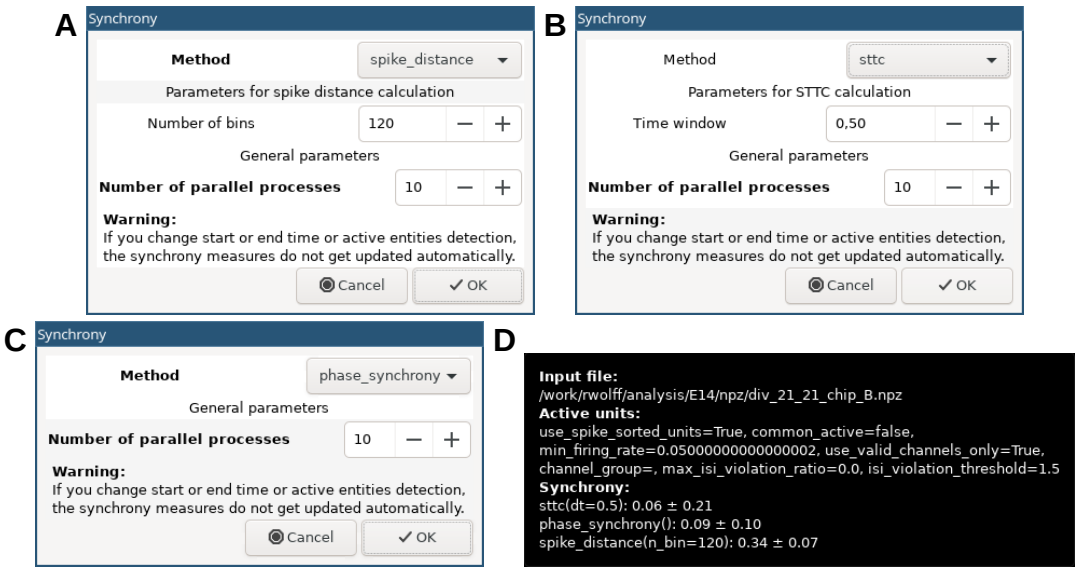


Figure 6: Synchrony settings for (A) spike distance, (B) STTC, (C) phase synchrony. (D) hint on mouse hovering over recording in main window.

After clicking ‘OK’, the selected synchrony measure is calculated. It can be verified by hovering over a recording that shows the mean and standard deviation of the calculated synchrony measures.

Calculate bursts and network bursts

Bursts are detected by clicking on the button ‘Bursts’ and subsequently network bursts are detected by clicking on the button ‘Network bursts’ (Figure 7). Different methods can be used for detection with dedicated parameter settings.

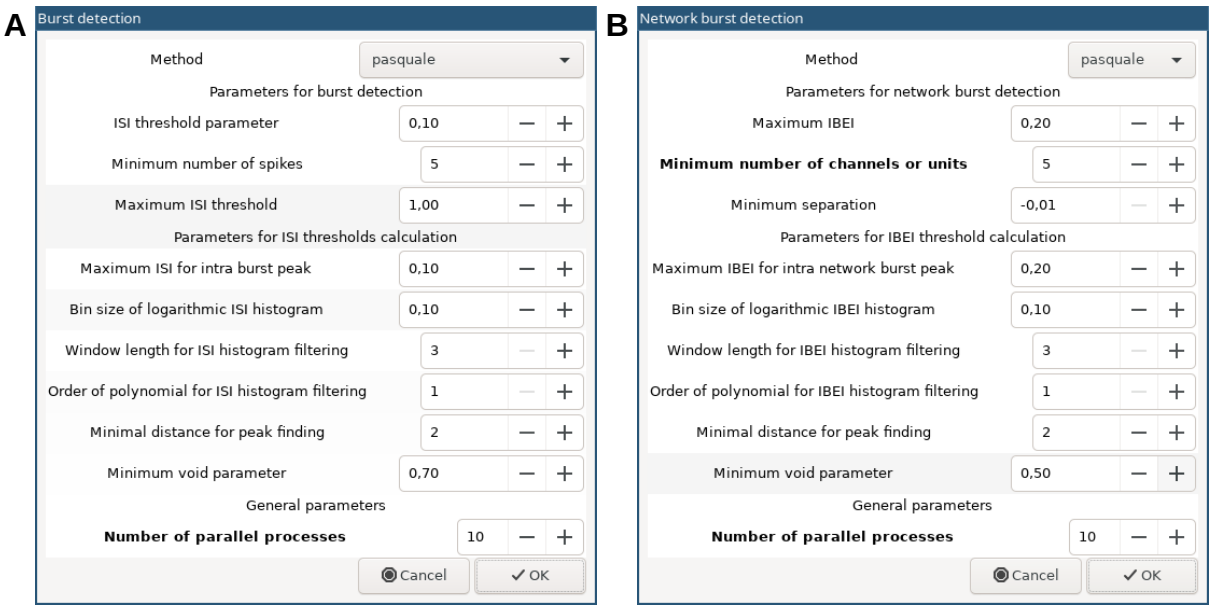


Figure 7: (A) burst and (B) network burst settings.

Once the burst or network burst detection is done, the dedicated ‘Bursts’ or ‘Network bursts’ columns are updated in the main window.

Plot and export single-recording parameters

By clicking on the button ‘Plot’ single-recording parameters are plotted in the chosen image format (`png`, `pdf`, etc., Figure 8A). Additionally, the file path to a plot settings file can be added that can be used for advanced settings of axis limits, etc. By clicking on the button ‘Export’, the analysis of each recording will be exported to the chosen file format (`ods`, `xlsx`, `pkl`, Figure 8B).

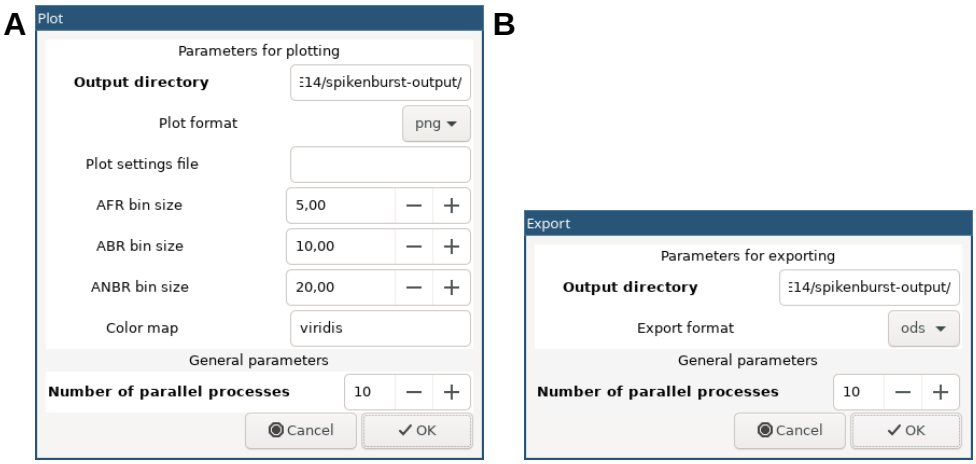


Figure 8: (A) plot and (B) export settings.

Compare multiple experiments

By clicking on the button ‘Compare’, the results of multiple recordings are compared by plotting

some basic plots in the specified output directory (Figure 9). This feature is still under development and may change in future.

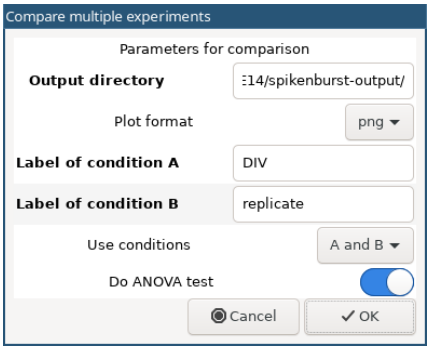


Figure 9: Comparison settings.

Supplemental S2 – Manual nicespike (version 0.2)

Installation

The `nicespike` tool (https://codeberg.org/spiky/nicespike) needs a GPU that is able to run CUDA on Linux or WSL. The recommended way is to run `nicespike` in docker. Firstly clone the docker branch of the git repository with:

```
git clone -b docker --single-branch
    https://codeberg.org/spiky/nicespike.git
cd nicespike
```

Then edit the file `docker-compose.yml` according to your wishes and build, compose and run the docker image with:

```
docker compose up --detach
```

Alternatively, `nicespike` can be installed in a python virtual environment with:

```
python -m venv venv_nicespike
source venv_nicespike/bin/activate
pip install --extra-index-url
    https://codeberg.org/api/packages/mahlzahn/pypi/simple/
    nicespike
nicespike
```

Usage of the graphical user interface

Start GUI and load input data

Once `nicespike` is running, the application is (by default) accessible at http://localhost:8080. Input recordings are selected by clicking on ‘Open brw’ and choosing the corresponding `brw` files or folders with these files. Once they are loaded, the window lists the input file paths in column ‘Path’, the size of the input files and 0 B for the output, because no analysis has been done, yet, in column ‘Size input + output’, and the empty checkboxes in the column ‘Analysed’ (Figure 10). The window also shows a line of buttons for the different actions and an indicator of used storage in the folder that is used for storing analysis data (by default `/data`).

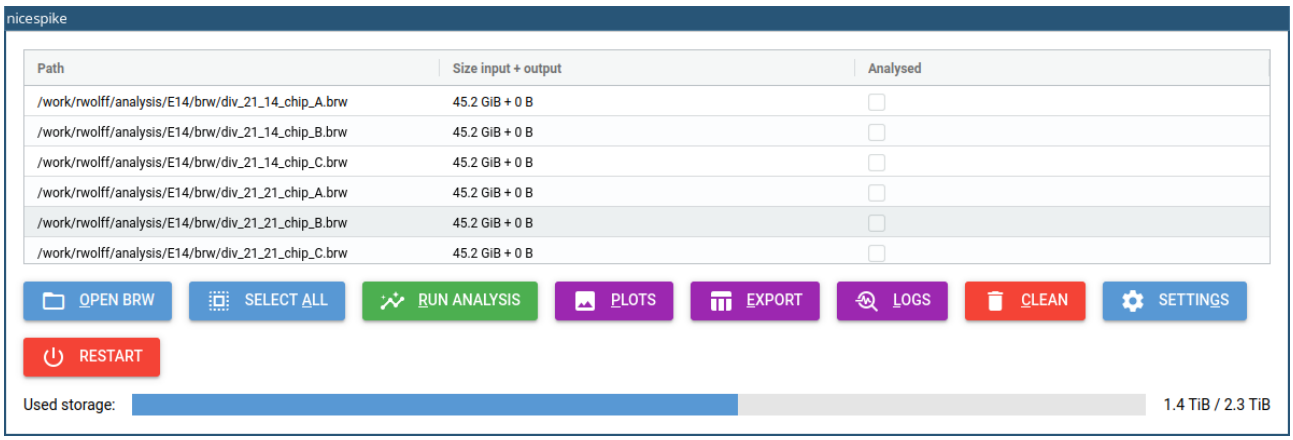


Figure 10: Interface of `nicespike` after loading the input recordings.

Configure analysis settings

The analysis and plot settings are configured by clicking on the button ‘Settings’. A dialog box with six tabs appears, where general, unit filtering (‘Spikes’), synchrony calculation, burst and network

burst detection and plotting settings are adjusted (Figure 11).

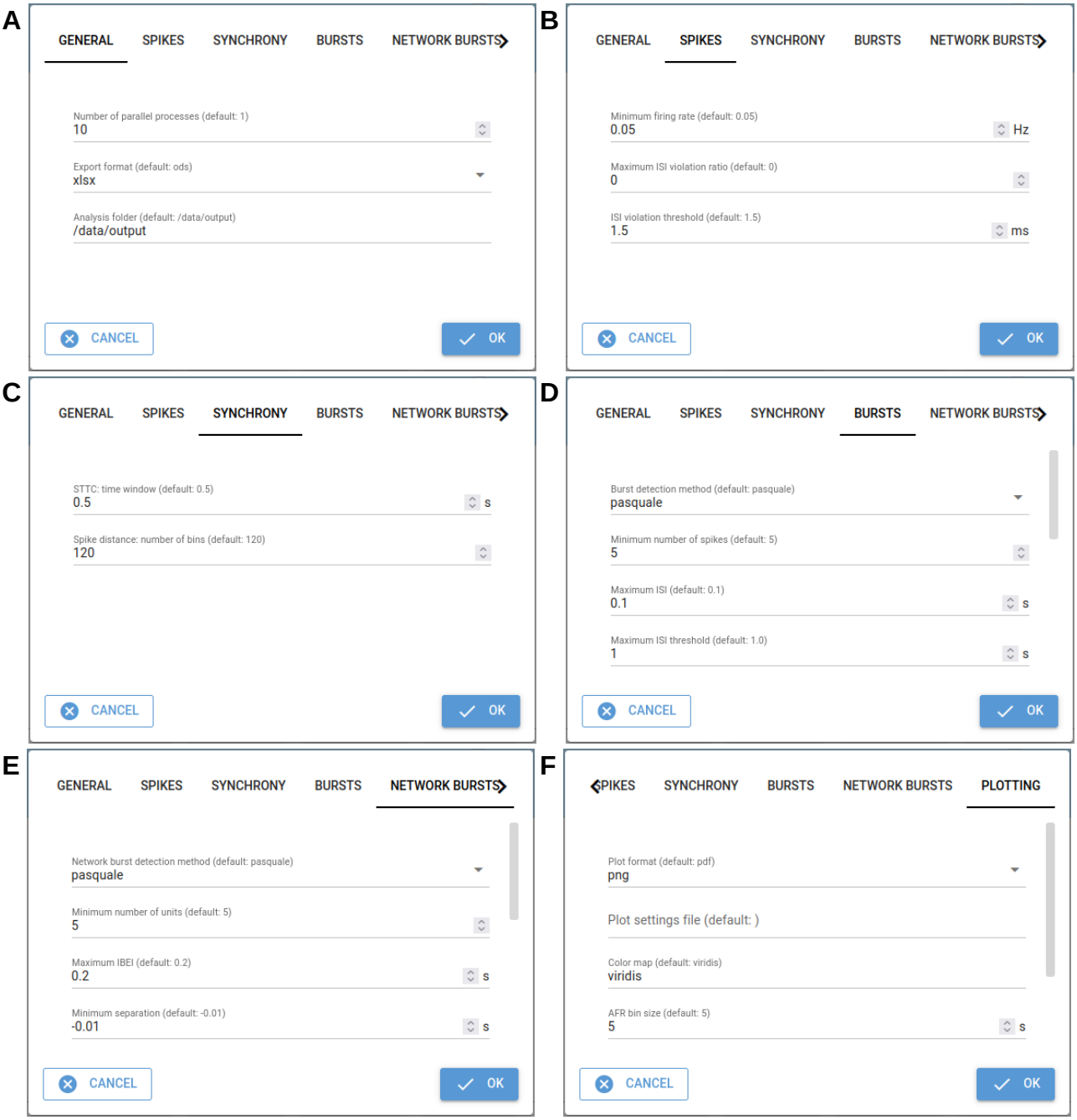


Figure 11: Dialog boxes for (A) general, (B) unit filtering ‘Spikes’), (C) synchrony calculation, (D) burst and (E) network burst detection and (F) plotting settings.

Analyse recordings

Before any analysis can be done, recording(s) must be selected from the list by mouse click or by clicking on button ‘Select all’. Make sure that the settings for the analysis are correct. If not you will need to rerun the whole analysis. You can adjust the settings by clicking on the button ‘Settings’. Once the analysis is running recording by recording, it can be only cancelled by clicking on the button ‘Cancel’. After confirmation of the cancellation the container is killed and restarted.

Plot and export single-recording parameters

Plotting of selected recordings is performed after clicking on the button ‘Plots’. Tables or python-pickled files are exported upon clicking on the button ‘Export’. To obtain the log files that have

been saved during the running of analysis, plotting and exporting, the button ‘Logs’ is clicked. In all three cases the data is gathered and packed in a zip file that then may be downloaded.

Remove analysis data from disk

After clicking on the button ‘Clean’ a dialog box appears with three options (Figure 12).

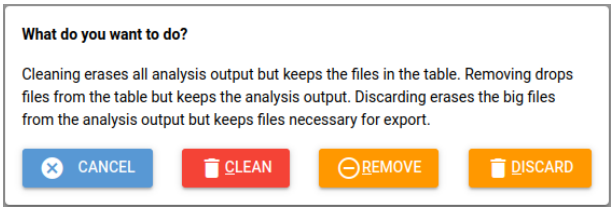


Figure 12: Dialog box for cleaning, removing or discarding.

1. ‘Clean’ selected recordings: erases all analysis outputs but keeps the files in the table,
2. ‘Remove’ selected recordings: drops files from the table but keeps the analysis outputs, and
3. ‘Discard’ selected recordings: erases big temporary files from the analysis but keeps necessary files for exporting.

Usage of the python module

The `nicespike` tool can be also accessed as a python library. With

```
from nicespike import settings, set_value, SpikeSorting

set_value(settings['General']['Number of parallel
    processes'], 8)
set_value(settings['Spikes']['Minimum firing rate'], 0.05)
set_value(settings['Network bursts']['Minimum number of
    units'], 5)
set_value(settings['Synchrony']['STTC: time window'], 0.5)

sorting = SpikeSorting(path='brw/div_21_21_chip_A.brw',
    output_dir='output/div_21_21_chip_A', settings=settings,
    verbose=True)
sorting.read_and_filter()
sorting.run_kilosort2()
sorting.extract_waveforms()
sorting.export_npz()
sorting.spikenburst_analysis()
sorting.spikeinterface_plot()
sorting.spikenburst_plot()
sorting.spikenburst_export()
```

the settings are adjusted, the raw voltage traces are read and filtered, the spikes are sorted, waveforms extracted, exported to `npz` and then further analysed by `spikeNburst` before plotted and exported. All results are saved to the given output directory.