CureQ MEA-analysis library user guide

12/08/2024

For additional questions, mail cureq-ft@hva.nl

Contents

Introduction

This file contains the instructions for the CureQ Microelectrode Array analysis library. It provides an easy way to perform MEA-analysis, from raw data to descriptive features. This algorithm is available in the form of an open-source python library. Users will have the capability to set their own parameters within the Python library; nevertheless, many of these parameters will come with pre-established values supported by extensive literature.
Additionally, users will have access to a GUI that interacts with the library functions and provides the user with interactive graphs. This will make MEA-analysis more accessible for people with little to no coding experience.
This analysis pipeline was made on behalf of CureQ. The Dutch research consortium CureQ is a collaboration of multiple universities, hospitals, applied universities, foundations and more, that aims to further understand the mechanics behind Huntington's Disease, provide more insight into the disease progression, better predict the age of onset and possibly come up with new treatment options (*Home-CureQ – CureQ*, n.d.).

Installation

Library installation

The CureQ library is available on the PyPi website, and can be installed using both pip and conda. The full code is available on GitHub. Installation instructions can be found on both websites.

Graphical user interface installation

The CureQ MEA-analysis tool is a Graphical User Interface (GUI) designed around the CureQ microelectrode array (MEA) python library. It allows for the user to easily interact with the library functions, as well as providing some extra functionality. The tool can be acquired in several different ways;

- The code for the GUI can be found at https://github.com/CureQ/MEA-analysis-tool. With an up-to-date python installation, and with all the required libraries installed, the user can start the GUI by executing the file: “CureQ_MEA-analysis_tool.py”
  This solution requires less storage space, but requires more programming knowledge.
- Additionally, an executable file can be found in the same repository. This file includes all the source-code, as well as a python installation and all the required libraries. After this file has been downloaded, the user can execute the file to start the GUI without any necessary installation steps. However, this will only work on Windows machines
  This solution requires more storage space (~500MB) but is easier to install.

Analysis

Methodology

Filtering

Typically, the first step when analysing raw voltage data is to filter the signal. The main goal of a filter is to improve the signal-to-noise ratio. A high pass filter will separate the slowly changing local field potential from the recorded spikes. A low-pass filter will remove unwanted noise from various sources above a certain frequency (Bullmann et al., 2019; Crosse et al., 2021; De Cheveigné & Nelken, 2019; Obien et al., 2015).

The optimal filter for this type of data is a Butterworth bandpass filter, a widely utilized filter type in neurophysiological research. This filter is maximally flat in the passband (Burgess, 2019), meaning the target signal will not be significantly altered. Secondly, the Butterworth filter is a causal filter, meaning that the way the filter behaves, is only affected by past, and not future data (De Cheveigné & Nelken, 2019).

Butterworth filters can be applied using different filter orders. Higher-order filters will have a steeper frequency transition than lower order filters. However, higher order filters may cause ripples and 'ringing' in the output signal. Therefore it is wiser to stick to lower order filters if possible (Burgess, 2019; De Cheveigné & Nelken, 2019).

For the cutoff frequencies we have analysed the literature regarding microelectrode arrays. We have observed that there is no clear consensus about what these values should be, as almost every research uses different values, although somewhat in the same range. In table 1 we have cited several studies, reviews and analysis tools that have utilized a Butterworth filter for MEA analysis.

Study	Cutoff frequencies	Order
(Mossink et al., 2021)	100-3500 Hz	2, 4 respectively
(Wagenaar et al., 2005)	100-3000 Hz	-
(Negri et al., 2020)	200-2500 Hz	-
(Hu et al., 2022)	200 Hz (high-pass)	2
(Black et al., 2018)	200-3000 Hz	1
(McConnell et al., 2012)	300-5000 Hz	-
(Napoli & Obeid, 2016)	180-3000 Hz	2
(Bradley et al., 2018) – using Axion biosystems AxIS software	200-3000 Hz	-
(Passaro et al., 2021) – using Axion biosystems AxIS software	300-5000 Hz	-
(Trujillo et al., 2019)	300-3000 Hz	3
(Obien et al., 2015)	300-3000 Hz	-
(Haq et al., 2023)	200 Hz (high-pass)	2
(Bullmann et al., 2019)	100-3500 Hz	2
(Tsai et al., 2017)	100-3000	-

Table 1. Cutoff frequencies and Butterworth filter order used in literature.

To make sure that we do not alter the signal too significantly, causing incorrect interpretations (De Cheveigné & Nelken, 2019), we have decided to use a 2nd order Butterworth bandpass filter. Secondly, we have chosen cutoff frequencies distant from the frequency range of the activity of interest, as recommended by De Cheveigné & Nelken, 2019. As typical spikes usually occur in the frequency range of 300-3000 Hz (Belitski et al., 2008; Lieb et al., 2017; Quian Quiroga, 2009; Tsai et al., 2017), and with the literature in mind, we have chosen a low cutoff of 200 Hz, and a high cutoff of 3500 Hz as default values. [1,2,3]

Threshold

When information is transferred from one neuron to another, a small change in current can be measured by the tiny electrodes on the MEA (Obien et al., 2015). To detect these "spikes", certain requirements are determined that a signal must fulfil in order to be identified as a spike. Spikes can be detected based on amplitude (Hu et al., 2022), waveform shape (Lieb et al., 2017) or a combination (Kapucu et al., 2022).
We have chosen for a simple threshold detection method, as this is a fast, easy to implement, widely used, and reliable method to detect neuronal activity (Negri et al., 2020; Obien et al., 2015; Swindale & Spacek, 2015; Tsai et al., 2017). If a voltage value exceeds the set threshold, it will be registered as a spike.
Our threshold calculations are inspired by the methods of Hu et al., 2022 and Nick et al., 2013. Whereas some research might calculate this threshold based on the entire signal, in this method, the threshold will be calculated based on only the background noise of the input signal. This way frequent, high amplitude, neuronal activity will not influence the threshold value.
This threshold is calculated for each electrode separately. The need for this becomes clear when we plot 2 noise segments from 2 different electrodes from the same well above each other (fig. 2).

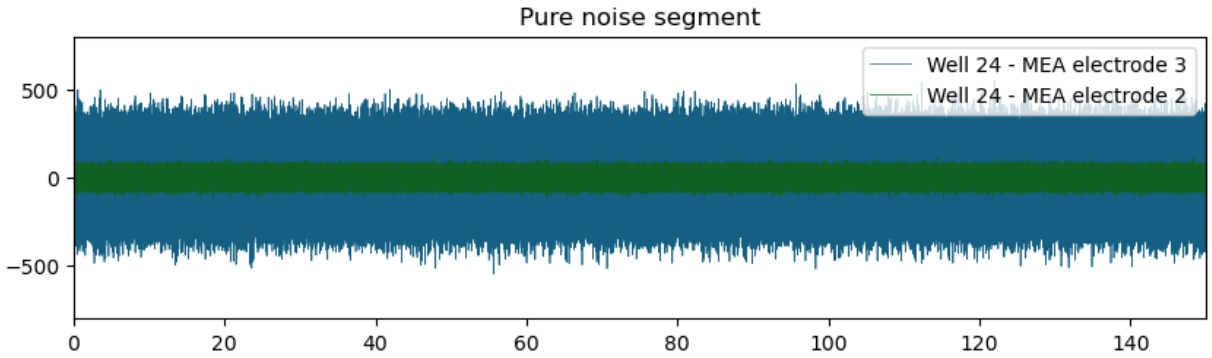


Figure 1. Comparison of pure noise segments of two different electrodes, located next to each other on the same well

In order to identify the background noise, we will be making use of the gaussian distribution of the background noise (Dolan et al., 2009; Müller, 2015; Tsai et al., 2017). Firstly, the data will first be split up in segments of 50ms. Next up, any segment that does not contain any voltage values outside a range of 0–5*SD (standard deviation) and 0+5*SD will be seen as "pure, spike-free noise". This means that these segments likely

do not contain any spikes. In other cases, there is data detected outside this range. Then it is likely that these “outliers” in the signal are caused by neuronal activity. These segments are not considered “pure noise”, as they most likely contain spikes. Example distributions of such segments can be found below (fig. 3,4), the red lines indicate the range of 0±5*SD.

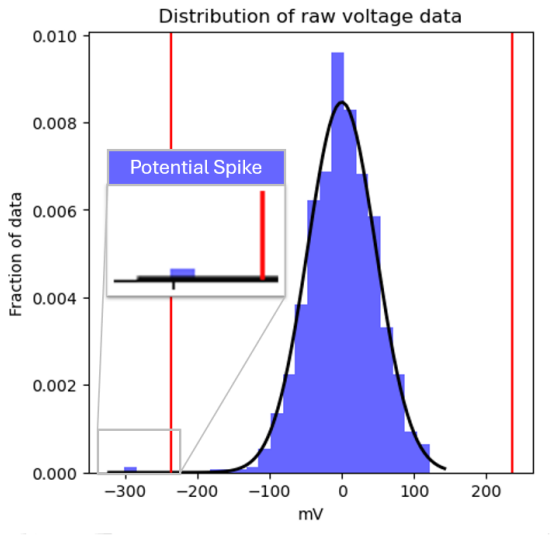


Figure 3. Probability distribution of data segment with potential spike event.

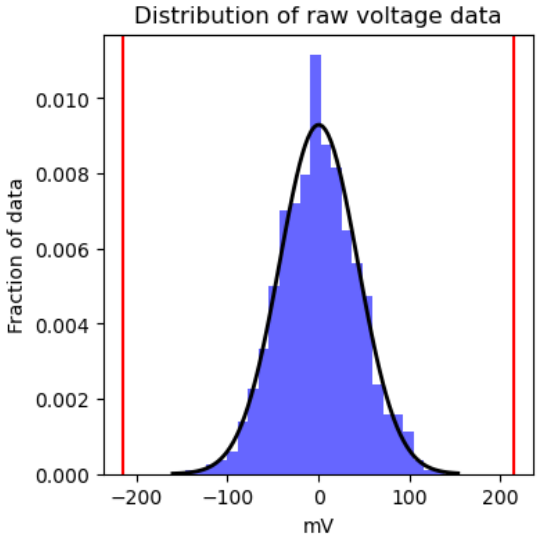


Figure 4. Probability distribution of data segment containing pure, spike-free noise.

The graph below shows how the algorithm marks the data. The red line indicates 50ms blocks of “pure noise”, while its absence indicates the presence of potential spikes (fig.5). Please note that this graph only shows a very small portion of the total data.

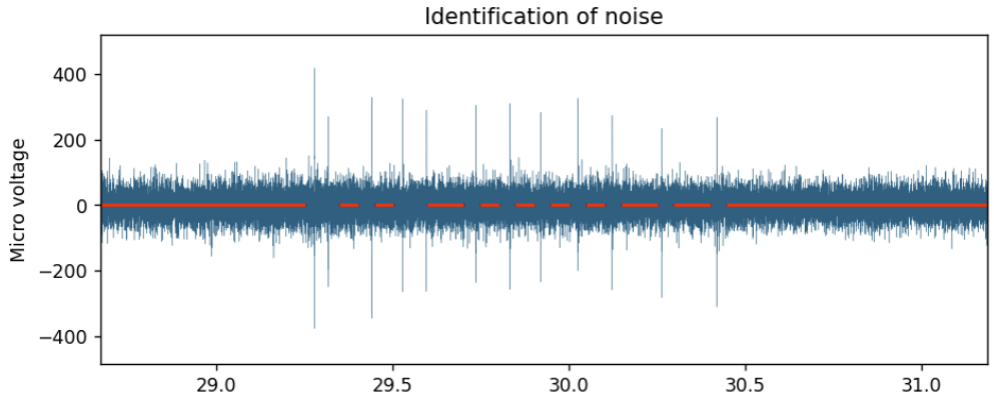


Figure 5. Identification of pure, spike-free noise in raw voltage data.

All these segments'' of “spike-free noise” will be collected and their values will be combined. From this the root mean square is calculated. The lower and upper threshold for spike detection will be set at –5*RMS and 5*RMS of the noise respectively (Hu et al., 2022; Negri et al., 2020; Nick et al., 2013; Obien et al., 2015). When we apply this threshold to a portion of the data that mainly consists of noise, we can see that the threshold is located just above the signal (fig.6).

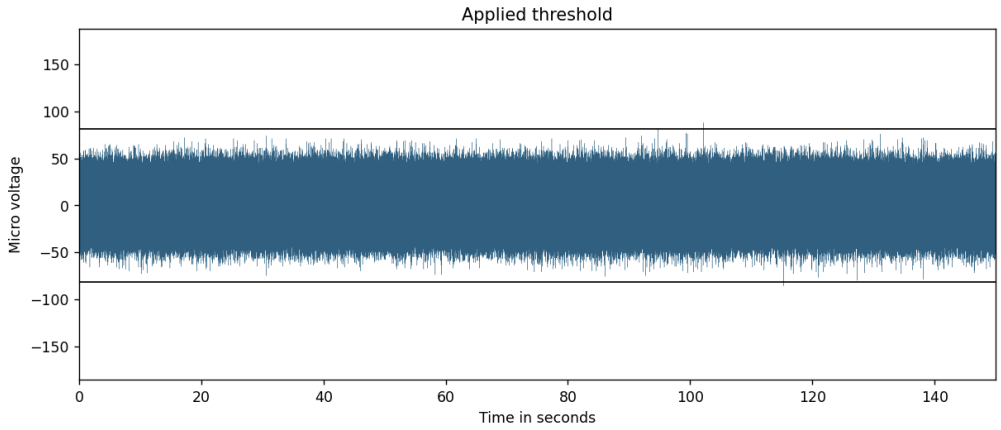


Figure 6. Threshold applied to a low-activity signal.

However, when we calculate the threshold on a very active measurement, we can see that the threshold stays just above the background noise level, meaning lower-amplitude spikes will also cross the threshold (fig. 7). This is because the threshold level is not affected by spike events.

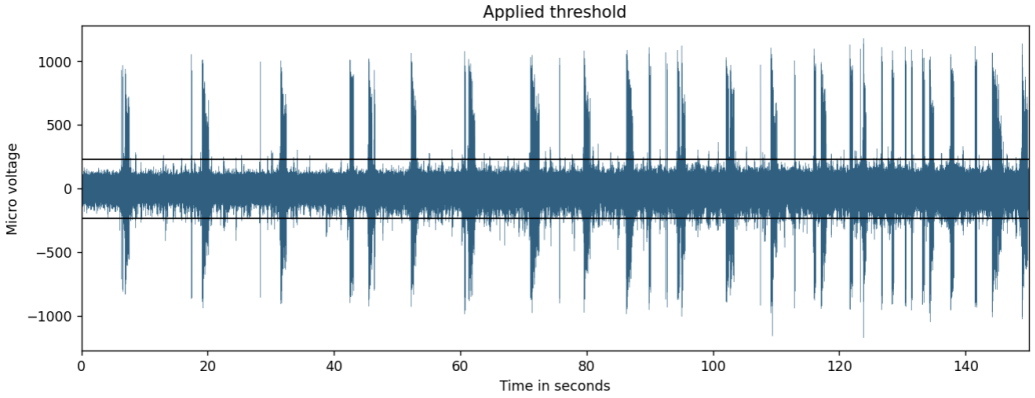


Figure 7. Threshold applied to a high-activity signal

Spike detection

For the first step of spike detection, the algorithm identifies every single voltage values that crosses the threshold. Next, it determines the peak of the spike, and make sure that only this peak is detected as a 'spike'. Without this step, multiple datapoints associated with an action potential will all be classified as a peak, resulting in an incorrect number of spikes measured. Below a typical spike from an example dataset is visualized, each red dot indicates a datapoint that has exceeded the calculated threshold for this electrode (fig. 8).

Figure 8. Multiple datapoints being associated with a single spiking event.

Many datapoints are currently being associated with just a single spike. In total, this single action potential will now register 3 negative and 5 positive spikes. The algorithm checks every datapoint that crosses the threshold, and inspects whether it has the highest absolute value within a ±1ms range, which is roughly the timeframe of an action potential (Hu et al., 2022; Muthmann et al., 2015; Swindale & Spacek, 2015; Tsai et al., 2017; Wagenaar et al., 2005). If it is not, this datapoint will no longer be registered as a spike (fig. 9).

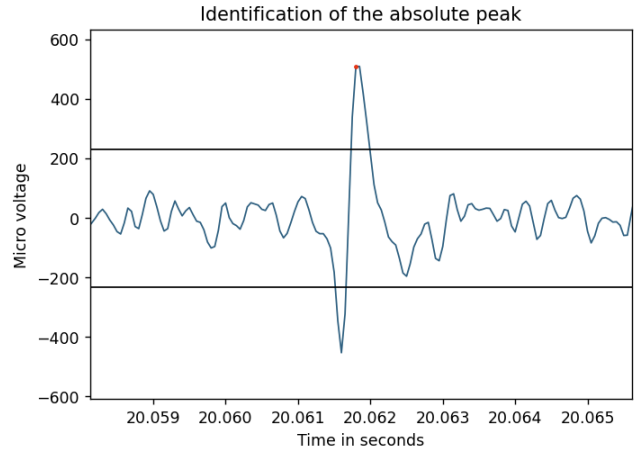


Figure 9. A single datapoint being associated with a single spiking event.

If we zoom out, this effect can be seen on a larger scale (fig. 10,11).

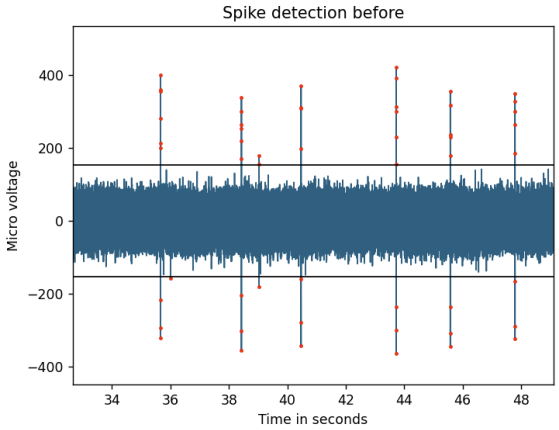


Figure 10. Spike detection before removing multiple registrations.

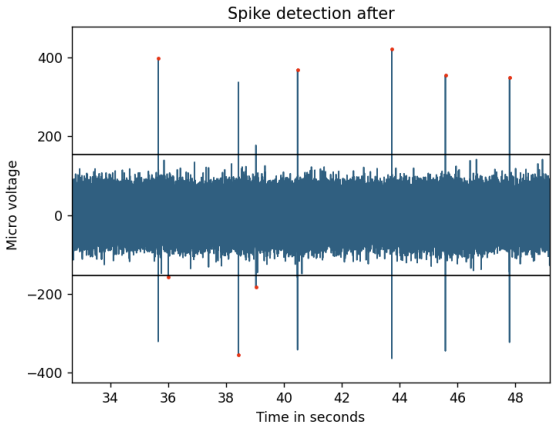


Figure 11. Spike detection after removing multiple registrations.

Spike validation

In order to reduce the number of false positives, we will apply certain requirements to a spike that it has to satisfy before being registered as one. The method we use for this is inspired by Hu et al., 2022; Swindale & Spacek, 2015 and Wagenaar et al., 2005, which require the spike to drop a certain amount of amplitude in a certain amount of time after it has reached its peak. Swindale & Spacek state that this drop must be twice the threshold value, in a timeframe of 0.24 milliseconds. The graph below visualizes how this method works. After the peak of spike has been detected, the signal must exit the rectangle at the horizontal edge (located at twice the threshold away from the peak) before it reaches the vertical edge (located 0.24ms away from the peak). A green rectangle indicates a spike has passed this requirement, while red indicates it did not (fig. 12,13). The 0.24ms timeframe using the parameter 'Exit time', see '**Spike detection**'.

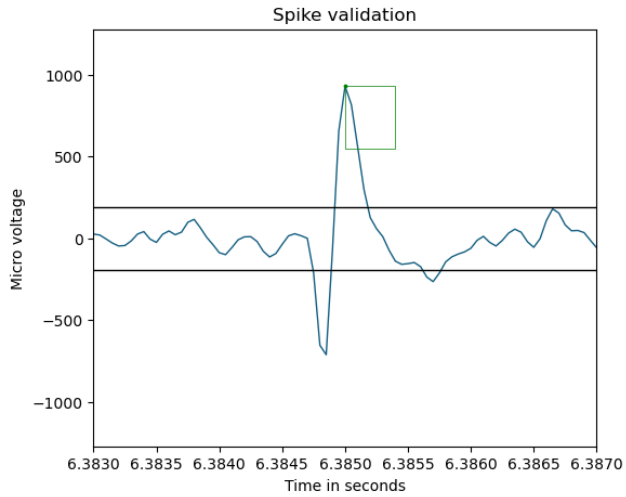


Figure 12. A spike successfully exits the bottom of the rectangle, thus being registered as a spike.

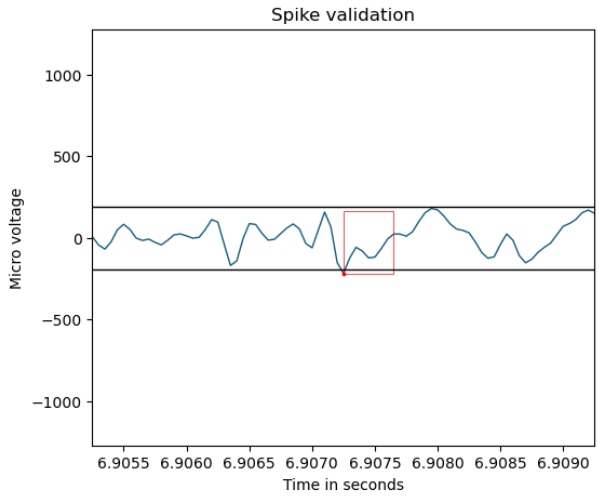


Figure 13. A threshold crossing fails to exit the bottom of the rectangle, thus being removed from further analysis.

After implementing this requirement into the algorithm, we noticed that this led to a steep decrease in detected spikes. Upon visual inspection of the data, these

requirements seemed quite harsh, only allowing for sharp high amplitude spikes to be detected, and easily discarding low amplitude signals. To retain as much information as possible, but also effectively filter out false positives, certain alternations were made to the original spike validation method.

Firstly, besides a spike only being accepted when it has a large increase/decrease in amplitude *following* its peak, a spike will also be accepted if it had a large increase/decrease *before* its highest peak. In other words, a spike will also be accepted when it enters the rectangle from the horizontal edge. This should facilitate the detection of spikes with a waveform similar to the one below, which would otherwise be discarded (fig. 14).

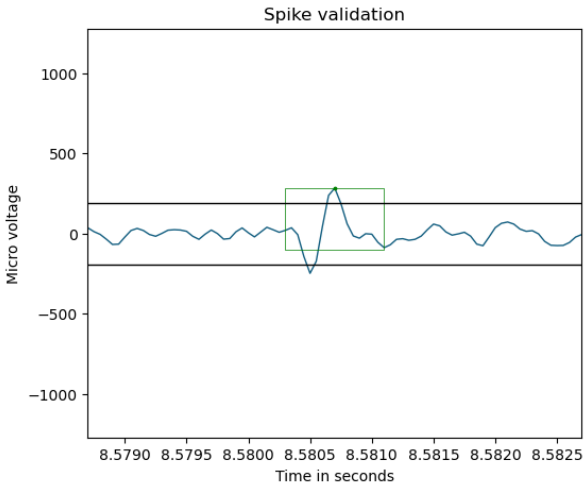


Figure 14. A spike successfully enters the horizontal edge of the rectangle, thus being registered as a spike.

Secondly, we noticed that some high amplitude spikes with very distinct waveforms were not getting registered, due to a slower descent of the amplitude. Following this we established another rule that if a peak reaches a certain amplitude, it will not have to pass the previously described test. The default value for this is 1.5 times the original electrode threshold value. This value can be changed using the parameter 'Height exception', see '**Spike detection**'.

While this method looks to achieve a very low false positive rate, it still seems to be too conservative when it comes to detecting spikes, potentially impacting the quality of burst detection and feature extraction later in the analysis. When looking at the data, we found that the noise level was not constant throughout the electrode. This means a harsher validation method might be necessary during high-noise periods, but might be eliminating valuable data during low-noise periods. To combat this, we altered the spike validation method by deriving the amplitude the spike must drop from the surrounding noise.

To calculate this new threshold, we fall back on the method described earlier, used for detecting spike-free noise. First, we identify a ±1 ms range around the detected spike, based on the aforementioned refractory period of the spike. This aims to prevent other spikes influencing the threshold. In this range, the standard deviation is calculated and

the amplitude a spike must drop will be set at RMS*5. This multiplier can be altered using the 'Drop amplitude' parameter, see '**Spike detection'**. To prevent this method from discarding more spikes than before, the required amplitude drop will have a default maximum value of 2 time the spike detection threshold of the electrode. This value can be altered using the 'Max drop amount' parameter, see '**Spike detection'**.

Below this method is visualized. The pink area represents the range around the spike where the standard deviation will be derived from. As a result from this algorithm, spikes in a low noise environment will be detected better (fig.15), while incorrectly identified spikes in a high noise environment will be discarded (fig. 16).

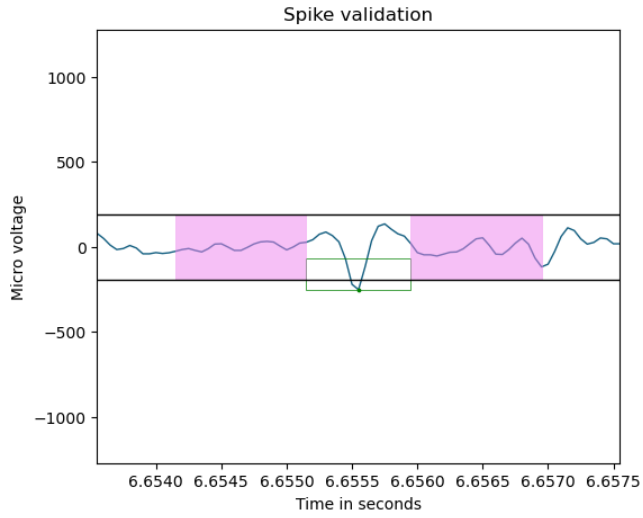


Figure 15. Due to the low amplitude of the surrounding noise, the height of the rectangle is lowered for a milder detection threshold.

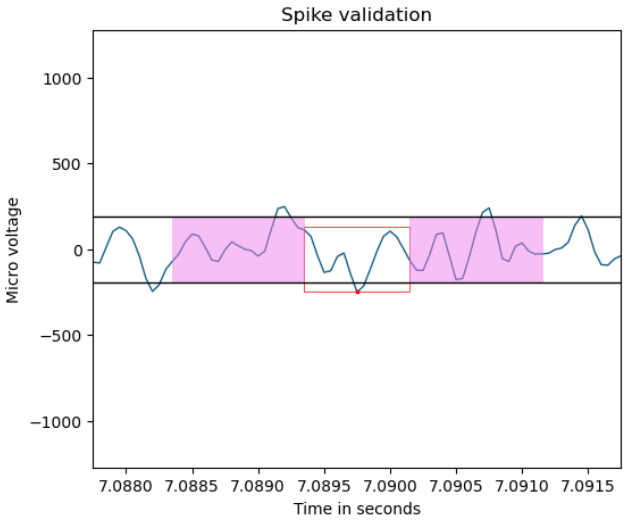


Figure 16. Due to the high amplitude of the surrounding noise, the height of the rectangle is increased for a harsher detection threshold.

Burst detection

A burst is defined as a series of action potentials generated within small intervals from each other. These bursts, and their different characteristics, are essential for communication between neurons (Zeldenrust et al., 2018). Therefore, correct identification of these burst is necessary to observe differences in behaviour between healthy and diseased cultures. Different methods of burst detection can be utilized depending on the situation and goals (Cotterill et al., 2016).

Over the years, many burst detection methods have been proposed, some more efficient than the other. Research suggests that methods based on the inter-spike interval (ISI) have the best overall performance (Bakkum et al., 2014; Cotterill et al., 2016; Mendis et al., 2016; Pasquale et al., 2010). Additionally, these methods are commonly used in other studies and MEA-analysis tools (Gelfman et al., 2018; Hu et al., 2022; Kapucu et al., 2022). Following this, we have decided to implement the a method based on the ISI in our pipeline, as described by Pasquale et al., 2010.

The LogISI method relies on the finding that, in some cases, the probability distribution of the ISIs follow a bimodal lognormal distribution. To calculate the distribution of the data, first the ISIs are calculated. Secondly, the distribution of the ISIs will be

established using the kernel density estimation (KDE) method. This is a well-established method for visualizing the distribution of continuous variables, eliminating certain drawbacks from histograms (Weglarczyk, 2018).

Once the kernel density estimate has been formed, all its peaks will be identified. Each value will be compared against a set amount of neighbouring values on both sides of the potential peak. If it is higher than all of the neighbours it was compared with, it is seen as a peak. Comparing a value with more neighbours will decrease the detected number of smaller peaks, and peaks closely located to each other. The amount of smaller neighbours will be set to 10 as default, meaning that the peak value must be higher than all of the 10 values surrounding it on both sides (marked in red).

If two peaks have been found, the minimum value (valley) between these peaks will be calculated. If the detected valley has a value less than 100 Ms, the valley value will be used as the maximal allowed interval between spikes in a burst (green). The default minimal number of spikes that a burst should consist of will be set as 5. In this example, the threshold was calculated at 26.59 Ms. By using this threshold, it is possible to discern the closely spaced spikes from actual bursts. If a standard threshold of 100 Ms would have been used, almost all spikes would have been identified as a burst.

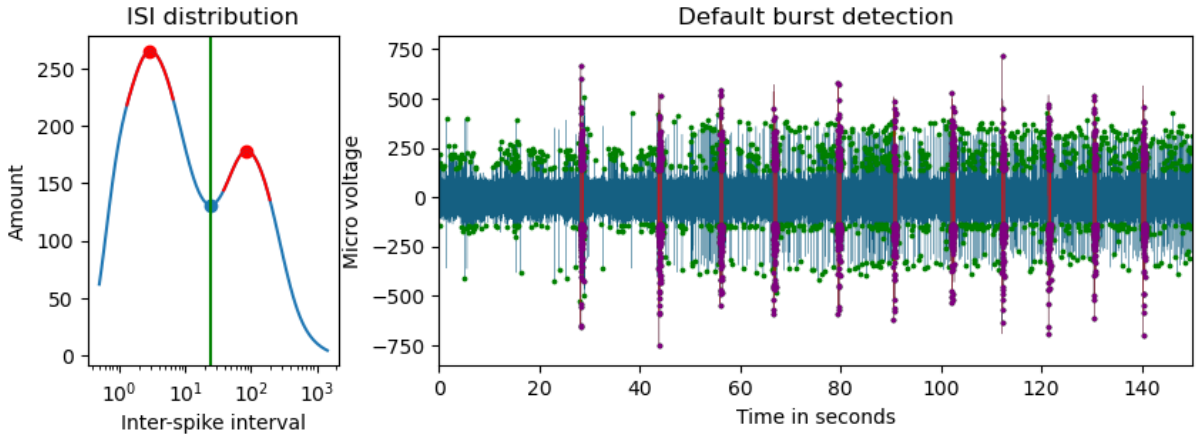


Figure 17. Burst detection using a lower threshold, derived from the distribution of the inter-spike intervals.

If the detected valley has a value of over 100 Ms, a different, advanced burst detection algorithm will be used. Two thresholds will be defined; threshold 1 will be set at 100 Ms and threshold 2 will be set at the value of the valley, with a maximum value of 1000 Ms. First, the algorithm will search the data for a “burst core”. A burst core consists of 5 spikes each separated by no less than the value defined as threshold 1 from each other. Secondly, all spikes outside of the burst core that have an ISI of less than threshold 2 will also be included in the burst. During this process, all bursts separated less than threshold 2 will be combined.

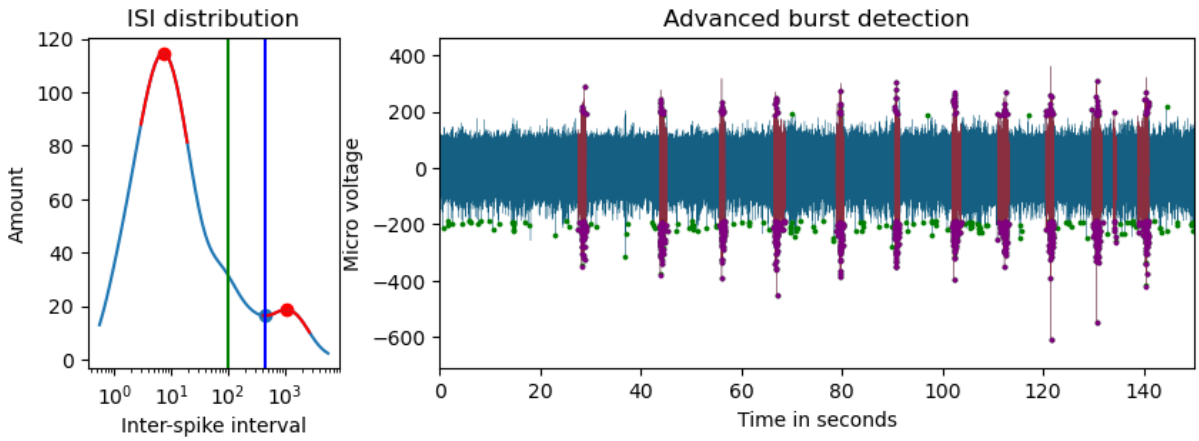


Figure 18. Burst detection using two thresholds, derived from the bimodal distribution of the inter-spike intervals.

If less than two peaks have been identified, the algorithm will fall back on a default algorithm with a max interval value of 100 Ms, and a minimal number of spikes of 5. Additionally, if the histogram does show a bimodal distribution, but does not detect a peak before 100 Ms, the burst detection will also be executed by the default algorithm.

If more than two peaks are identified, the algorithm will attempt to find two peaks closely located to the default threshold values (100 and 1000 Ms). If it is able to do so, the algorithm will apply the advanced burst detection algorithm. If it cannot find two suitable peaks, it will fall back on the default algorithm with a threshold value of 100 Ms.

Network activity

During maturation, neuronal networks show an increased amount of bursting, which eventually develops into synchronized bursting events that can be detected across different electrodes in the entire MEA (Mossink et al., 2021). Identification and analysis of these network burst can be used to measure the connectivity of a network, and to determine electrophysiological differences between healthy and diseased networks (Frega et al., 2019; Mossink et al., 2021). Therefore, it is essential to correctly detect these network events.

The method we have implemented is inspired by the methods of the MEA-toolbox and meaRtools (Gelfman et al., 2018; Hu et al., 2022). First, the algorithm calculates 4 different graphs. The upper graph is the raster plot, which shows all spikes (blue) and spikes related to single channel bursts (SCBs) (red). The second plot shows the total amount of burst that are currently taking place at that specific timepoint. The third graph shows a KDE of all the spikes that are registered in the well. The fourth and final graph shows a KDE of only spikes that are part of a burst. All KDE graphs are scaled from 0 to 1.

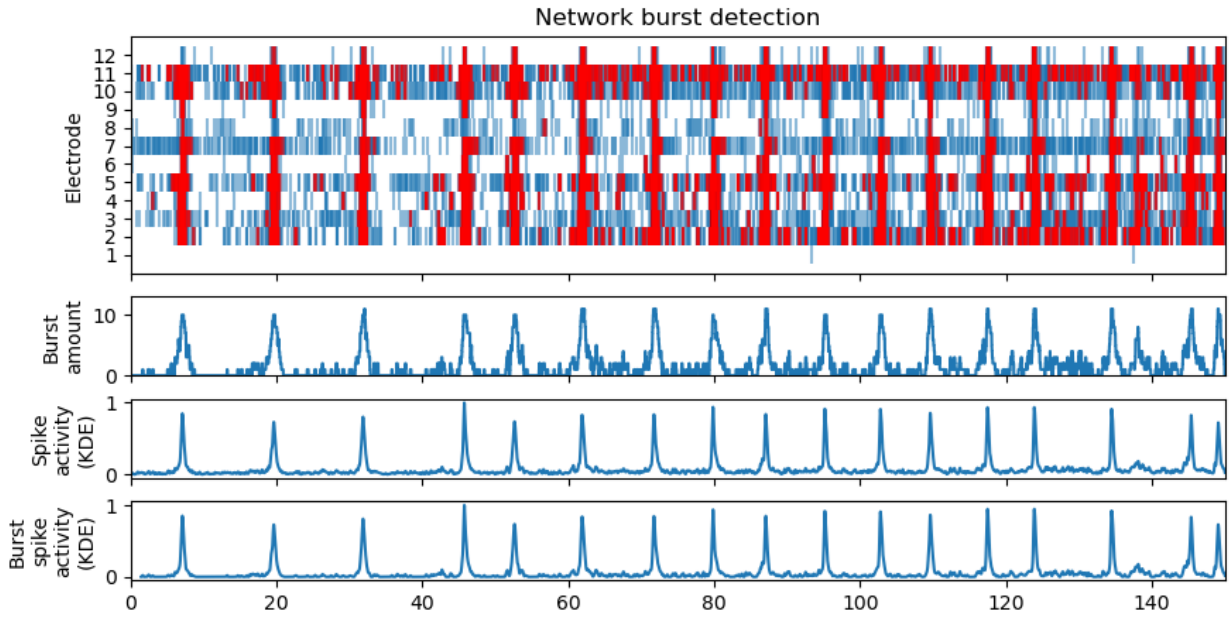


Figure 19. Graph showing the spike/burst activity per electrode, active bursts, spike activity and burst spike activity.

In this method, periods of high spike activity related to bursts across a significant portion of the well will be seen as a network burst. Using the bottom graph, a threshold will be calculated using either Otsu's or Yen's method (Otsu, Nobuyuki, 1979; Yen et al., 1995). In our experience, Otsu's method is more conservative in network burst detection, usually giving a higher threshold, while Yen's method usually calculates a lower threshold, identifying lower-intensity NBs. Any timeframe that crosses the threshold will be seen as a "network burst core" (NBC). With the initial crossing of the threshold set as the start of the NBC, and the subsequent dropping below the threshold as the end of the NBC (green).

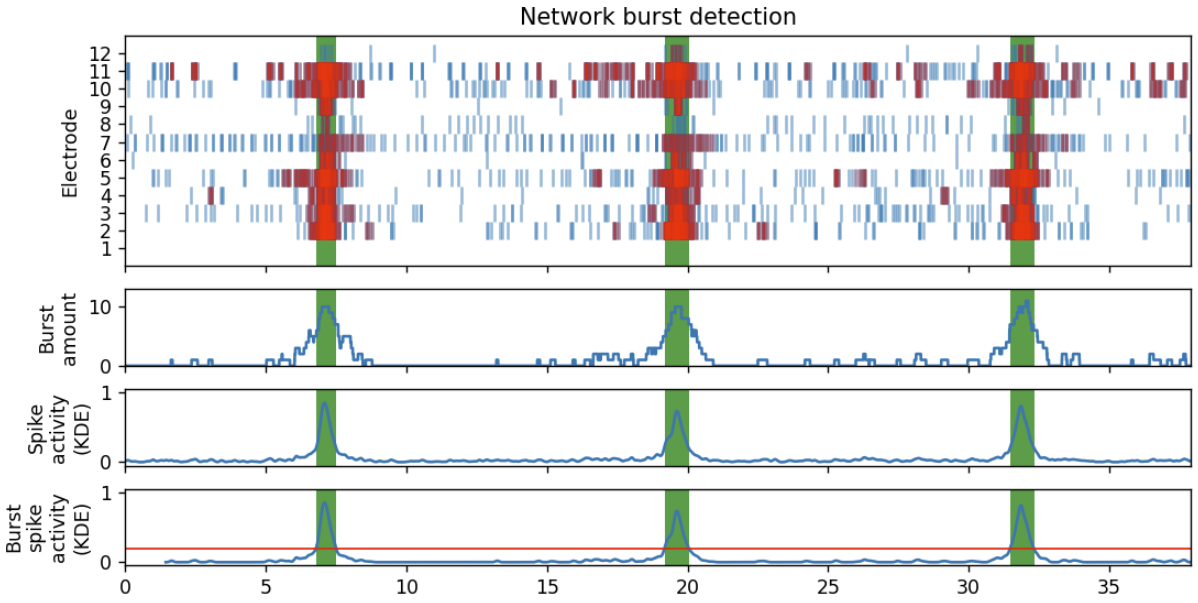


Figure 20. Identification of network burst cores using the burst spike activity, threshold and actively bursting channels.

To make sure a network burst is only detected when it is measured on multiple electrodes, the user can specify the percentage of electrodes that must participate in a NBC, for it to be identified as one. The standard value is set at 50% (0.5), meaning that at least half of the electrodes must register bursting activity for a network burst to be identified.

Lastly, all single channels bursts (SCBs) that overlap with the NBC will be identified. The lower and upper edges of the network burst will be set at the extremes of the SCBs (light green). In the case that this causes 2 network bursts to overlap, the shortest network burst will be removed.

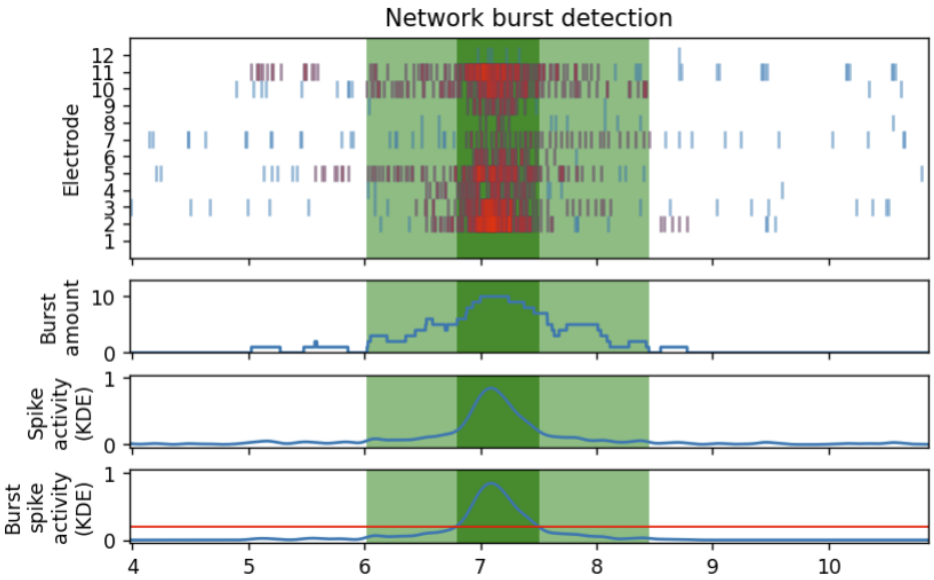


Figure 21. Identification of the outer edges of the network bursts, using the extremes of the single channel bursts
participating in the network burst.

Results

Feature extraction

The goal of this analysis pipeline is to extract meaningful features from the data, which can then be used for statistical analysis, or fed into a machine learning algorithm, such as a random forest model. In table 1, we describe which features the algorithm calculates. These features are inspired by various other studies (Gelfman et al., 2018; Hornauer et al., 2024; Hu et al., 2022). If certain features cannot be calculated, the value will be set as "NaN". At the end of the analysis, these features are exported to a .csv file, where the user can use it to perform statistical analysis. All single electrode features (spike and burst features) will be averaged over all electrodes in a well.

Feature	Description	Column name
Spike features		
Spikes	The total amount of spikes recorded in a single electrode (spikes)	`spikes`
Mean firing rate	The average activity of the electrode, calculated as the average amount of spikes per second. (spikes/s)	`mean_firingrate`
Mean ISI	Mean inter-spike interval. The average amount of time between two successive spikes. (s)	`mean_ISI`
Median ISI	The median of the inter-spike intervals. (s)	`median_ISI`
Ratio of median ISI over mean ISI	The ratio of the median ISI over mean ISI. Calculated by dividing the median by the mean.	`ratio_median_over_mean`
Inter-spike interval variance	The average of the square deviations from the mean from the inter-spike intervals. (s)	`IIV`
Coefficient of variation ISI	The coefficient of variation of the inter-spike interval. (s)	`CVI`
Partial autocorrelation function of the ISI	The partial autocorrelation of lag 1 of the ISIs. This value is calculated using the "statsmodels" python library (Seabold & Perktold, 2010)	`PACF`
Burst features		
Total number of bursts	The amount of burst that are detected in a single electrode. (bursts)	`burst_amnt`
Average burst length	The average length of a burst in a single electrode. (s)	`avg_burst_len`

Burst length variance	The variance in the length of the bursts. (s)	`burst_var`
Coefficient of variation burst length	The coefficient of variation of the burst length. (s)	`burst_CV`
Mean inter-burst interval	The average amount of time between two bursts. (s)	`mean_IBI`
Variance of inter-burst interval	The variance of the inter-burst interval. (s)	`IBI_var`
Coefficient of variation IBI	The coefficient of variation of the inter-burst interval. (s)	`IBI_CV`
Mean intra-burst firing rate	The average firing rate in a burst. (spikes/s)	`intraBFR`
Mean spikes per burst	The average amount of spikes per burst. (spikes)	`mean_spikes_per_burst`
Mean absolute deviation of spikes per burst	The mean absolute deviation of the amount of spikes per burst. (spikes)	`MAD_spikes_per_burst`
Isolated spikes	The portion of spikes that are isolated (not part of a burst). (value from 0 to 1)	`isolated_spikes`
Single channel burst rate	The average amount of bursts per second. (bursts/s)	`SCB_rate`
Network Features		
Network bursts	The total amount of network bursts in a well (network bursts)	`network_bursts`
Network burst duration	The average duration of a network bursts. Calculated as the distance between the outer edges of the network burst. (s)	`network_burst_duration`
Network burst core duration	The average duration of the network burst core. (s)	`network_burst_core_duration`
Network burst to network burst core ratio	The average ratio between the length of the network burst and network burst core. Calculated by dividing the	`NB_NBc_ratio`
Variation of network IBI	The variation between the network interburst intervals (s)	`nIBI_var`
Coefficient of variation network IBI	The coefficient of variation between the network interburst intervals (s)	`nIBI_CV`
Ratio of left outer burst	The average ratio of the left outer burst compared to the burst core	`NB_NBC_ratio_left`
Ratio of right outer burst	The average ratio of the left outer burst compared to the burst core	`NB_NBC_ratio_right`

Left right ratio	The ratio of the left outer burst compared to the right outer burst	`lr_NB_ratio`

Table 1. Features calculated by the MEA analysis algorithm.

Validating the analysis pipeline using known existing phenotypes

To test the validity of the analysis pipeline, we have analysed data derived from neurons with known phenotypes, and checked whether the algorithm is able to replicate the same results as is found in the literature.

Kleefstra syndrome

We were able to receive a small MEA dataset, containing measurements of the neuronal activity of healthy and mutated neurons. The data originates from a study on Kleefstra syndrome (Frega et al., 2019), and was provided to us by researchers associated with the Leiden University Medical Center (LUMC).
The original kleefstra study reported significant differences on 4 individual features.

Network burst rate

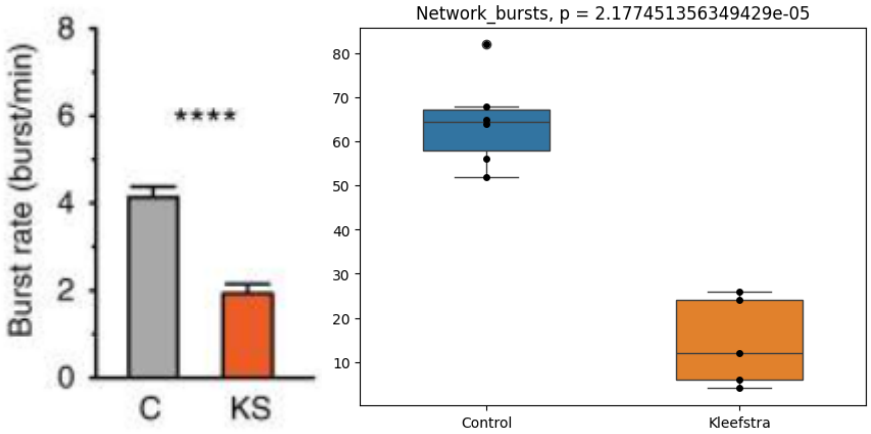


Figure 29. Comparison of average network burst rate.

Average network burst duration

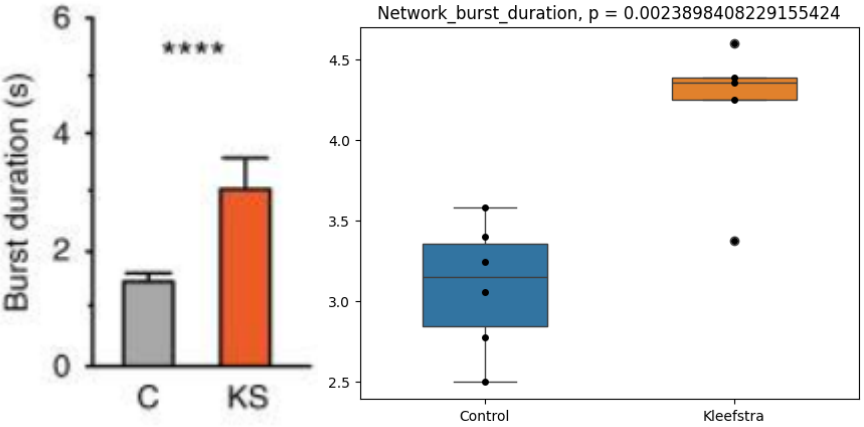


Figure 30. Comparison of average network burst duration

Network Inter-burst interval

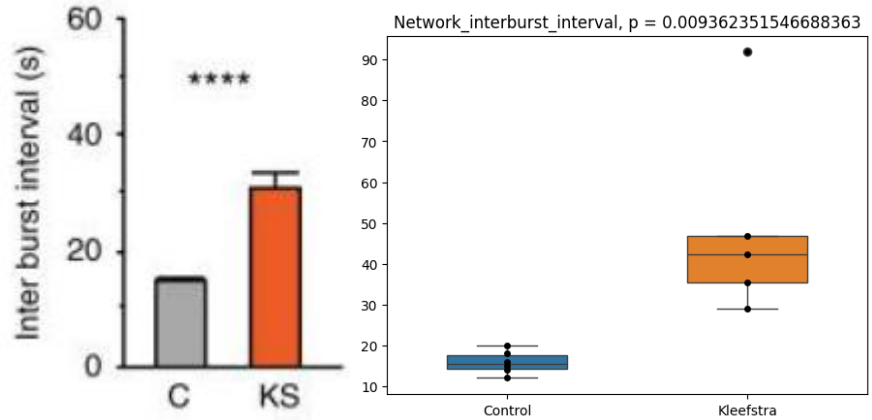


Figure 31. Comparison of network inter-burst interval.

Network Inter-burst interval coefficient of variation

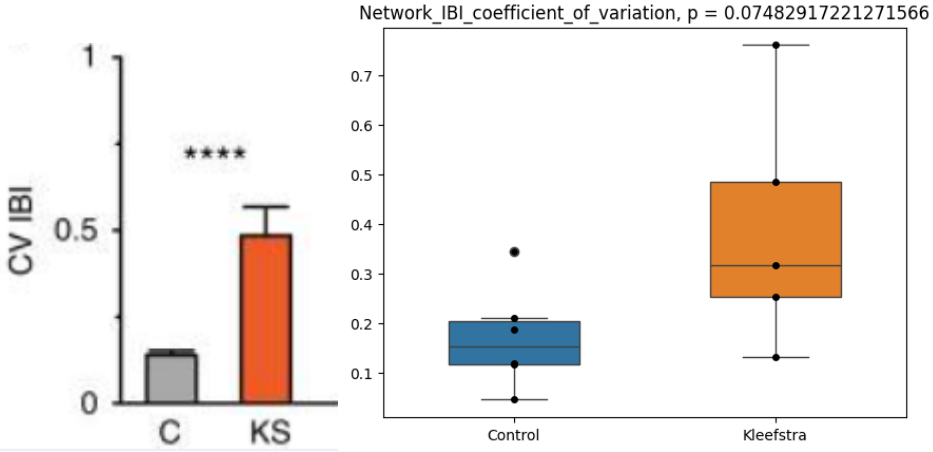


Figure 32. Comparison of network inter-burst interval coefficient of variation.

Please note that these results might differ slightly, as we do not have access to the full size dataset.

Glutaminase Deficiency

Additionally, we also received a dataset containing measurements originating from neurons containing a mutation in the glutaminase (GLS) gene. GLS facilitates the conversion of glutamine to glutamate. A decreased glutamate production should result in overall lower activity levels in the neuronal cultures.
Indeed, we do find in our analysis that GLS mutated neurons show significantly less spiking activity, which also leads to reduced bursting activity (fig. 33, 34).

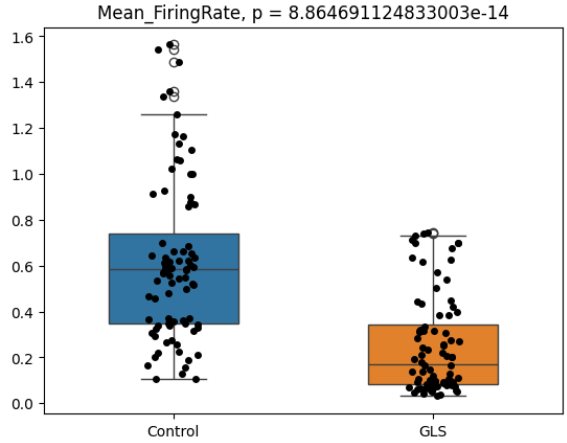


Figure 33. Comparison of average firing rate between healthy and mutated neuronal cultures.

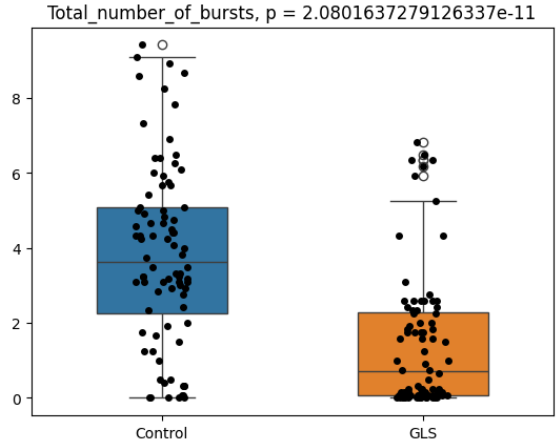


Figure 34. Comparison of average number of bursts between healthy and mutated neuronal cultures.

Speed

The algorithm has been benchmarked together with MEA-toolbox (Hu et al., 2022) on three different machines, using 2 different datasets. The multiprocessing setting in our own algorithm was only enabled on some runs, as it requires a large amount of RAM and processing power, which is not available on all machines. The analysis was run using default parameters.

	Type	CPU	CPU cores	CPU frequency	RAM	Storage
Machine 1	Laptop	AMD Ryzen 5 4500U	6	2.3GHz	8GB	SATA SSD (550 MB/s)
Machine 2	Personal computer	AMD Ryzen 7 2700x	8	3.7GHz	16GB	M.2 SSD (5150 MB/s)
Machine 3	Jupyter server	AMD EPYC 7313	16	3GHz	128GB	Unknown

Dataset	Size (hdf5 format)	Sampling frequency	Duration	Active Wells
Kleefstra	2.35 GB	10,000 Hz	1200s	12
GLS	880 MB	20,000 Hz	120s	24

Analysis details	MEA-Toolbox	Own algorithm
Machine 1 – GLS	38.12min	1.55min
Machine 1 – Kleefstra	7h24m**	9.21m
Machine 2 - GLS		1.27min*
Machine 2 – Kleefstra		6.41min*
Machine 3 – GLS	-	52.19s*
Machine 3 – Kleefstra	-	2.27min*

*Multiprocessing enabled

**Extrapolated from the time it took to process 1/12 wells

Library

The CureQ Microelectrode Array (MEA) analysis library currently contains one function, which contain the entire analysis pipeline, with many different parameters that can be set to alter the analysis.

Analyse_well()

The `analyse_well()` function contains the entire analysis pipeline; reading the raw data, filtering, spike detection, burst detection, network burst detection, and feature extraction.

The function has three required parameters that the user must set before the analysis:

- The path to the raw data file
- The sampling frequency of the MEA
- The amount of electrodes per MEA well

Besides these parameters, the function has many optional parameters, more information can be found in the **Parameter overview** section.

The library only supports hdf5 files with a certain internal file structure. Raw data from Multichannel systems' machines should work right away, Axion Biosystems' files should first be converted using a custom MATLAB script, which can be found further in this document.

Example usage

```
from CureQ.mea import analyse_well                # Library function for
analyzing wells

file_path = 'path/to/the/mea_file.h5'            # Path to the MEA file
hertz = 20000                                     # Sampling frequency of MEA
system
electrodes = 12                                   # Electrode amount per well

# Analyzes all wells in the MEA file
if __name__=='__main__':
   analyse_well(fileadress=file_path, hertz=hertz, electrode_amnt=electrodes)
```

Output

Every time this function gets called, it will create a folder containing all the results of the analysis. In this folder, the following items can be found:

Item	Contents
spike_values	Folder containing all spike timestamps for the spikes in all electrodes
burst_values	Folder containing information regarding bursts in all electrodes

network_data	Folder containing information regarding network bursts of all wells
figures	Folder containing figures for each well
features.csv	Csv file containing all features for each well
parameters.json	Json file containing all parameters used in this analysis

After the data has been analysed, it is possible view the raw data and inspect the results using the GUI. Go to **View results** and select the correct output folder and raw data file.

For experienced python programmers, it is possible to alter the library, and to calculate additional features. First, clone the GitHub repository and find the 'features.py' file. This file contains the code that is responsible for calculating all the features, as well as instructions on how to add extra features.

Dependencies

The library depends on several external libraries:

matplotlib
pandas
seaborn
scipy
numpy
statsmodels
KDEpy
scikit-image
h5py
plotly

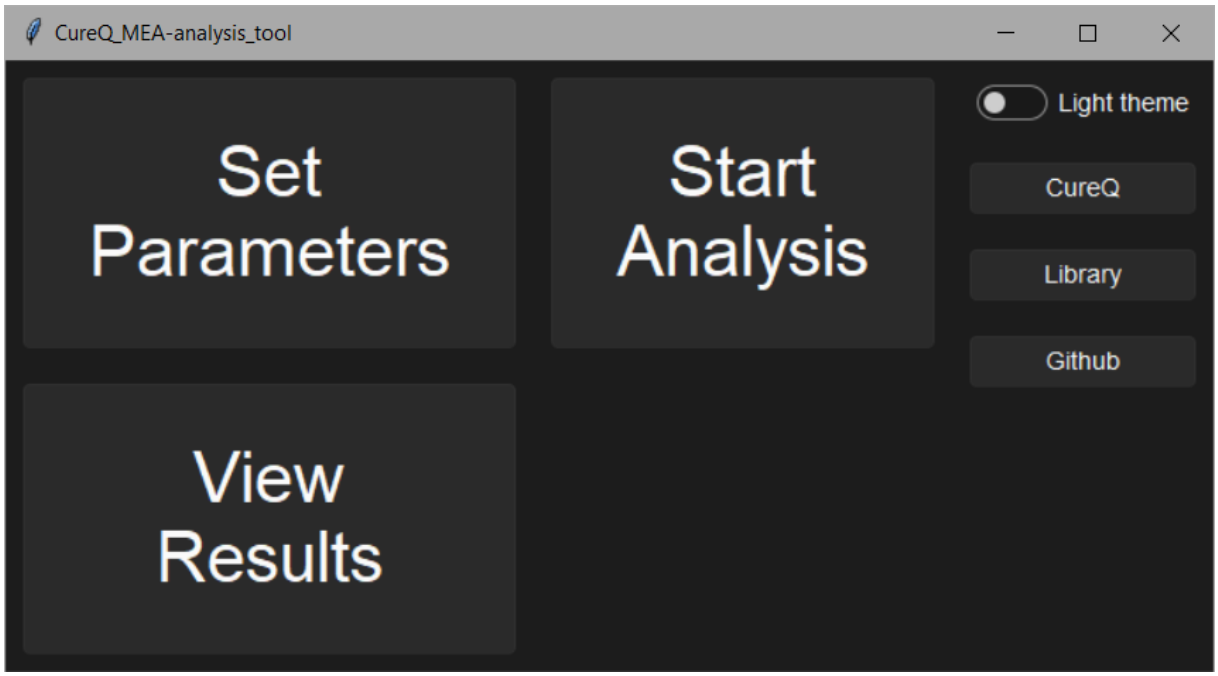
With pip, these libraries can be downloaded using the following command:

```
pip install matplotlib pandas seaborn scipy numpy statsmodels
KDEpy scikit-image h5py plotly
```

Graphical user interface

Start Menu

When the program is executed, the user will be presented with the following start menu. The side buttons will redirect to the CureQ website, pypi library and GitHub.



Set parameters

When selecting **Set Parameters**, the user can select the raw datafile, and alter different parameters for the analysis.

Start Analysis

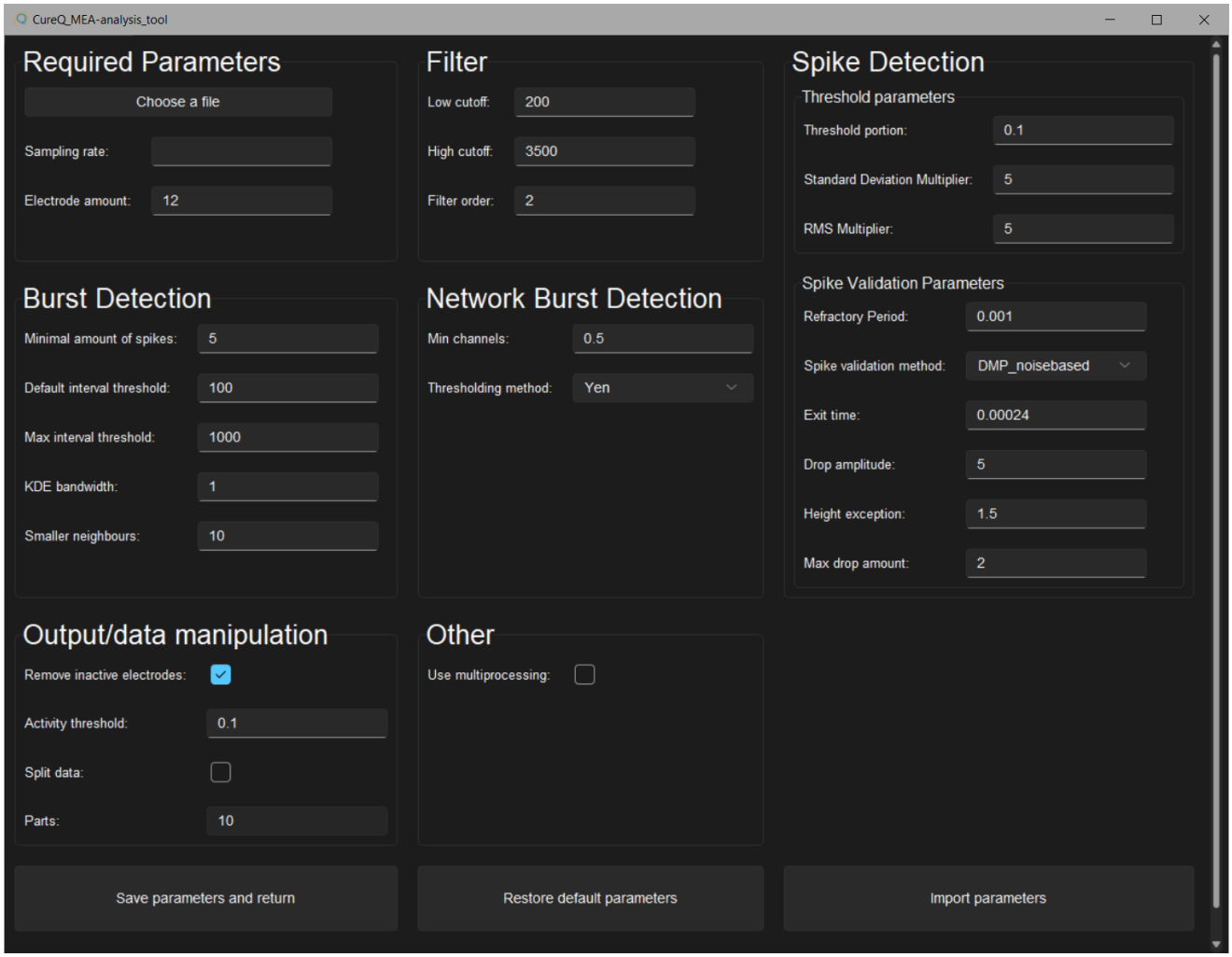
By clicking **Start Analysis**, the user confirms the parameters, and initiates the analysis. Once the analysis is complete, the results will be located in a folder with a name containing the file name, date and time of the analysis. This folder contains the results of the experiment in a file called 'Features.csv', as well as several graphs in the subfolder 'Figures'. The output folder is located in the same location as the original raw data.

View results

The **View results** button allows the user to view the results from the current, or previous experiments.

Parameters

The user will have the option to set a large variety of different parameters, which will later influence the analysis. A short description of each parameter can be found below. For more information about the parameters and the analysis pipeline, see the 'Methods' section.



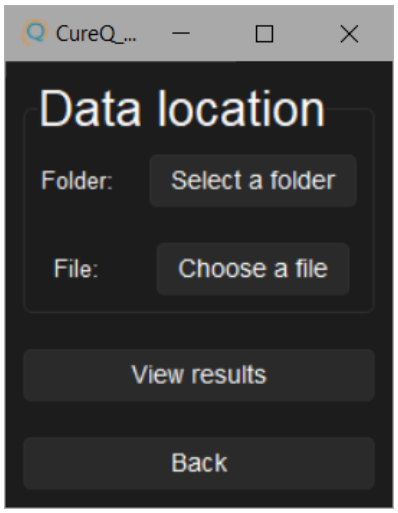
Every analysis will generate a file called 'parameters.json', which contain information about the parameters that were used for that particular analysis. Using the button **Import parameters**, the user can import parameters from a previous experiment.

When clicking **Restore default parameters**, all parameters will be reset to their default values.

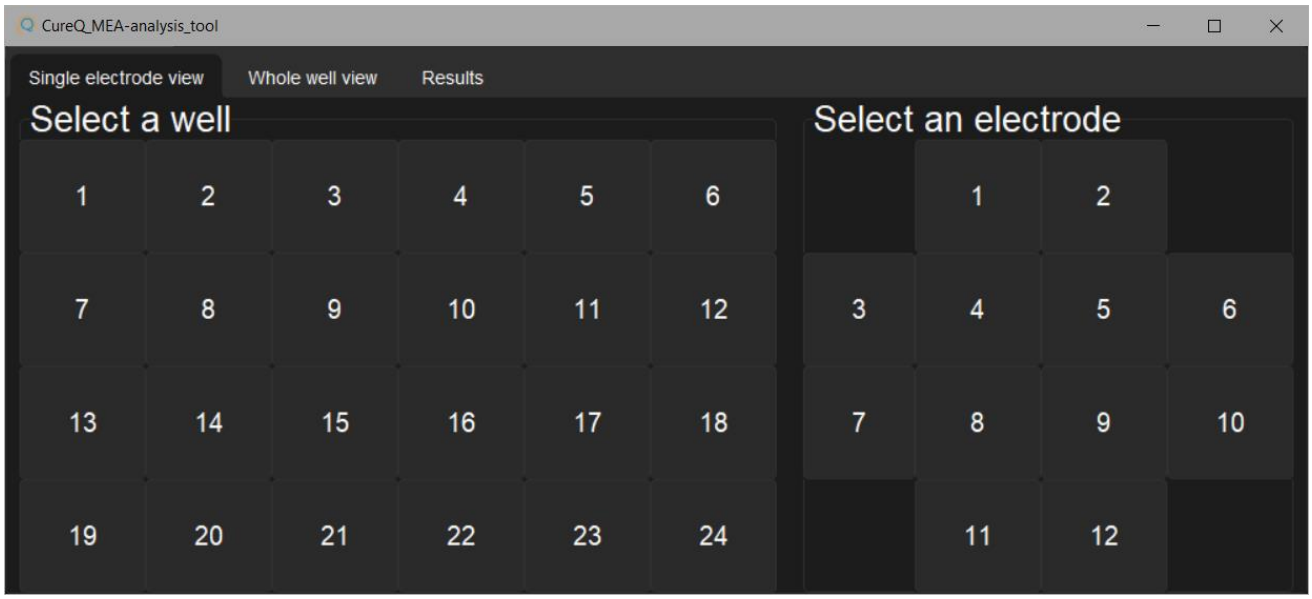
Finally, click **Save parameters and return** to save all parameter choices, and go back to the start menu.

View Results

The **View results** button allows the user to view the results from the current, or previous experiments.



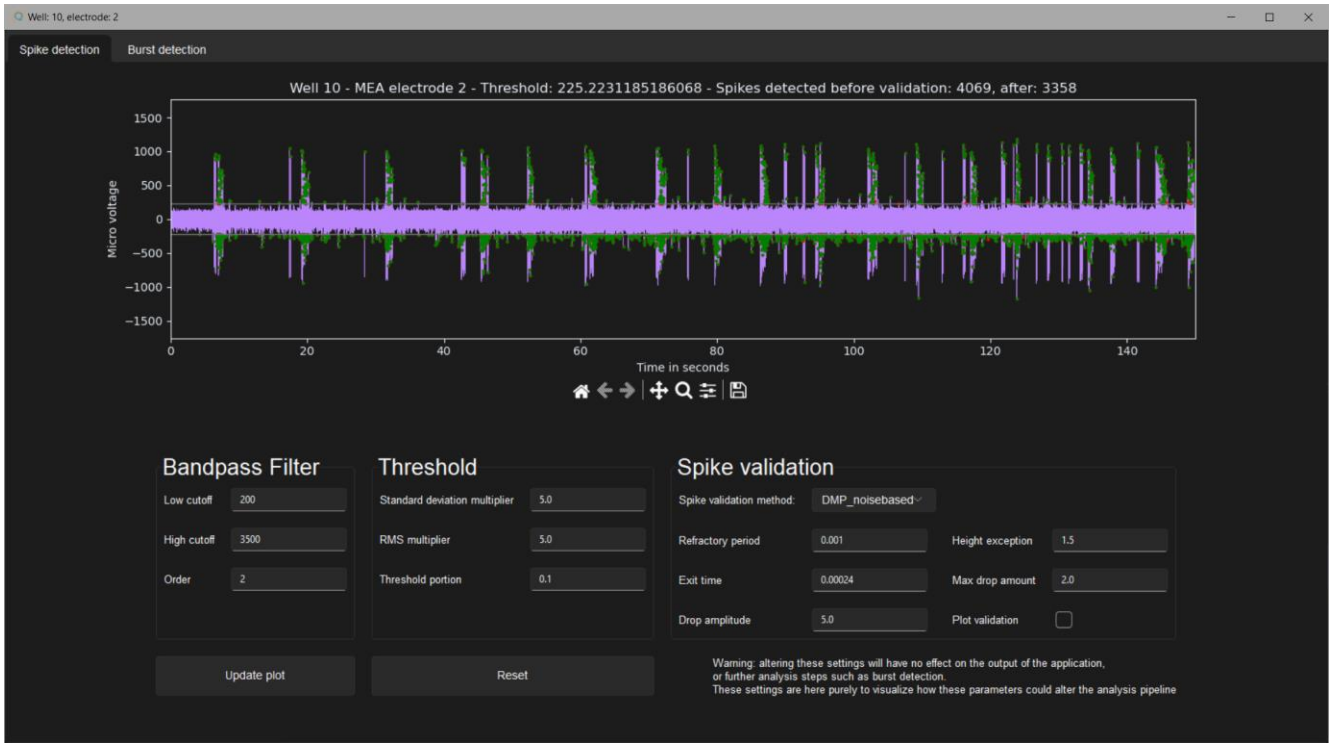
In the following window, the user can select the output folder associated with the desired experiment. Next, the user must select the raw data file that this experiment was done on. Lastly, press '**View results'**.



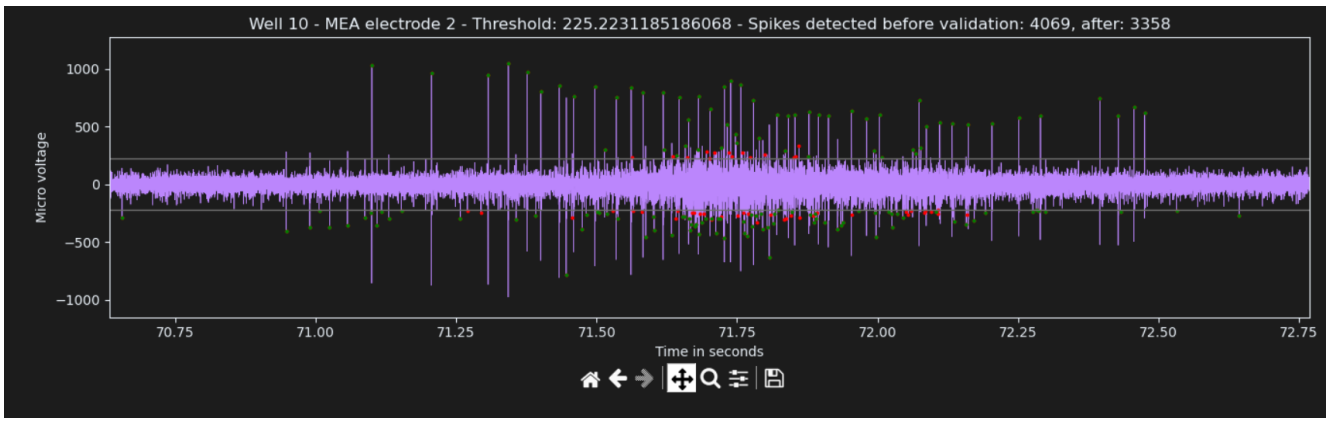
On the top of this window we can see 3 tabs, **Single electrode view**, **Whole well view** and **Results**.

Single electrode view

Using this page, the user can view the raw electrode data of a single electrode, as well as seeing the spike and burst detection process. First, click on the desired well, this choice will now be saved so there is no need to press it again later. Secondly, press the desired electrode.



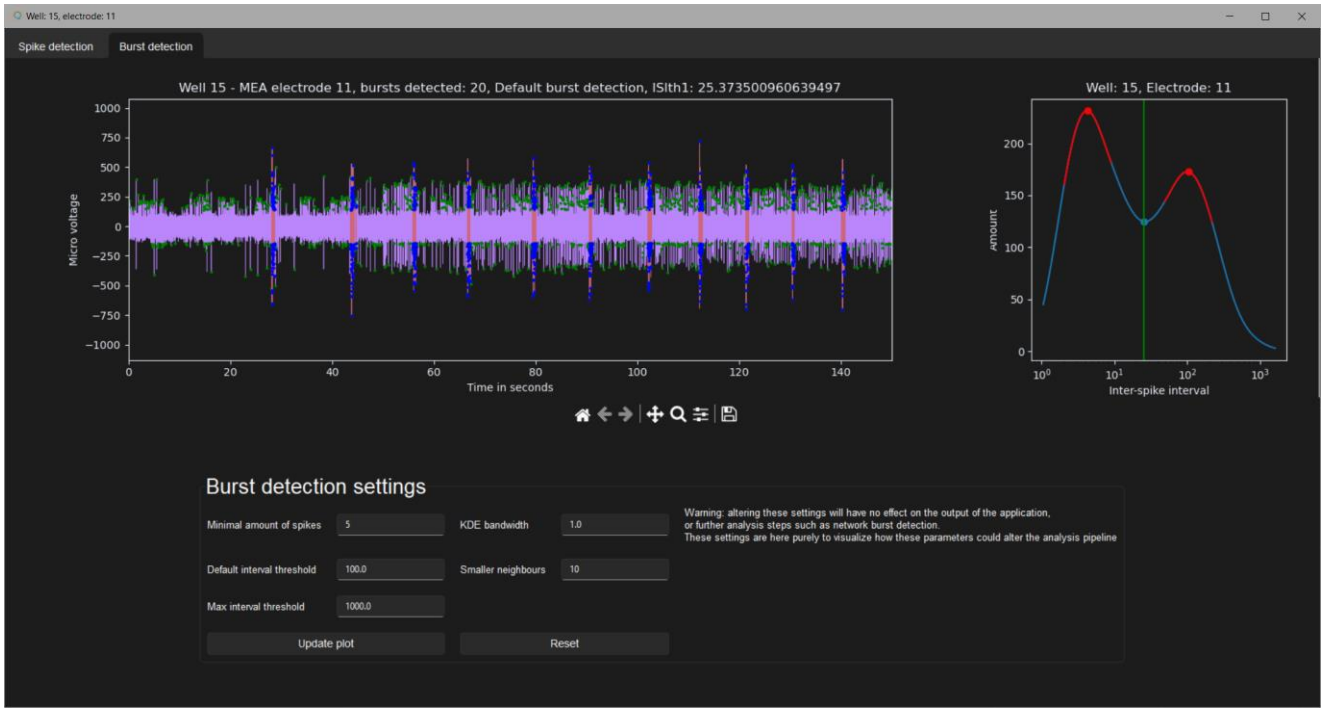
This will present the user with a window, consisting of two tabs; **Spike detection** and **Burst detection**.
On the **Spike detection** tab, the user can view the raw voltage data, zoom in using the magnifying glass, and pan using the arrows.



Additionally, the user can alter certain parameters of the spike detection process in the bottom half of the window. Change a parameter, then click **Update plot**. The spike detection will then be rerun with the new parameters. Please note that this functionality is only to visualize how these parameters could have potentially affected the outcome of the analysis. They do not alter any of the other steps such as burst detection, network burst detection and feature calculations. To use these parameters for the entire analysis, alter them at the **Set parameters** tab and rerun the entire analysis.

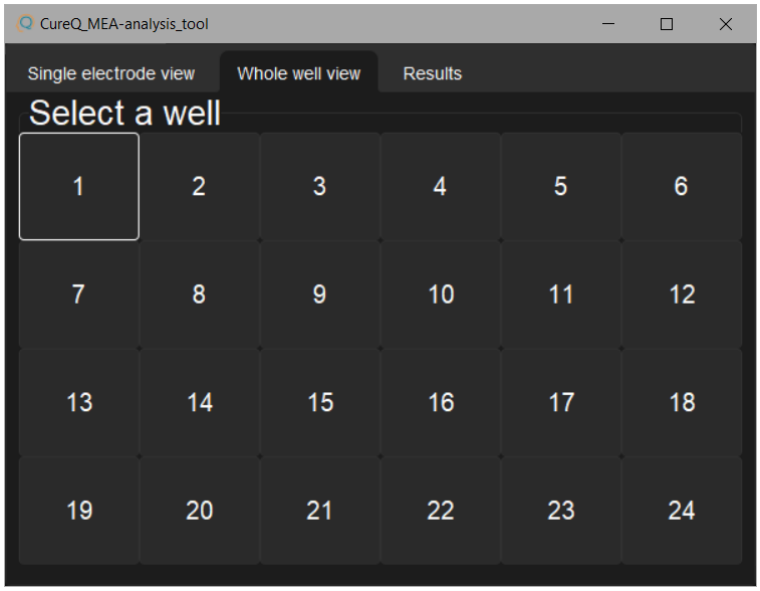
The **Burst detection** tab is similar to the spike detection tab. It is possible to interactively view the raw data, and inspect the burst detection process. On the top right

of the window, the kernel density estimate graph that is used for the burst detection is displayed. Once again it is possible to alter the burst detection parameters, but keep in mind these only serve as an example of how these parameters work.



Whole well view

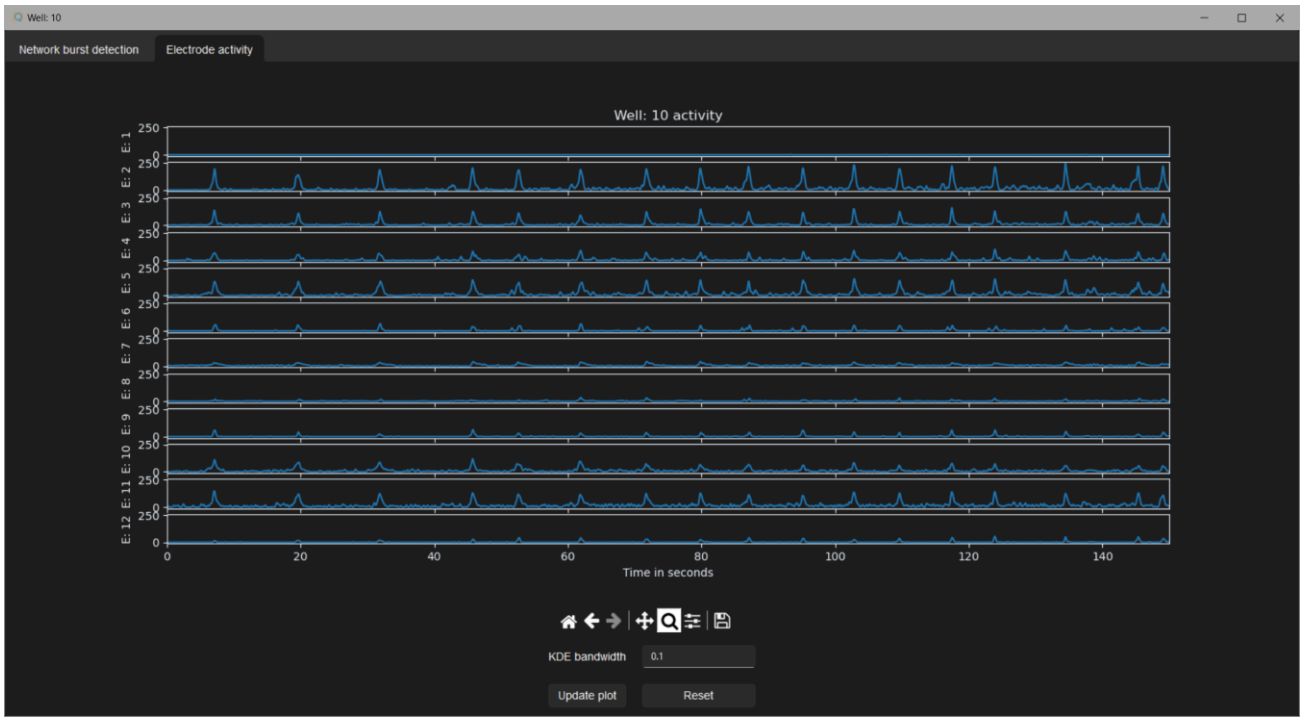
The ‘Whole well view’ tab will display a button for each of the wells in the measurement.



By selecting one of the wells, the user will be presented with the following window. This window shows the network burst detection process. The plot is interactive, and the parameters for the burst detection can be altered to discover what effect this has on the network burst detection.

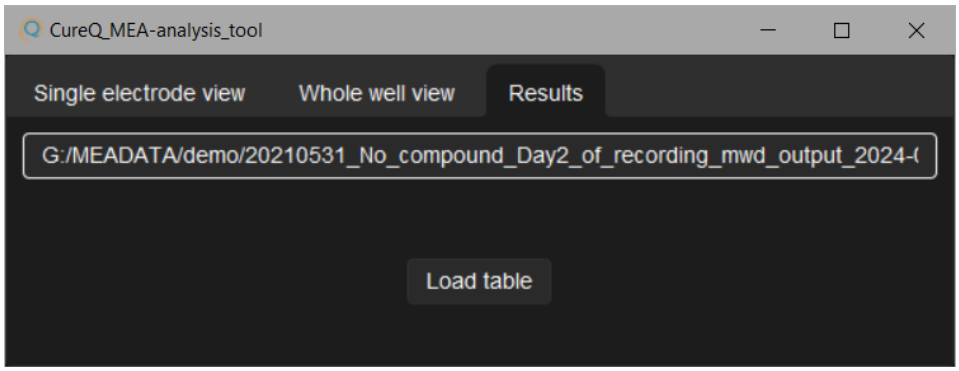
The top graph shows a raster plot of the entire well. The wells are on the y-axis, the x-axis is the time in seconds. Every vertical line means a spike, if this line is coloured red, it means that this spike is part of a single channel burst. The upper middle graph shows the number of channels (electrodes) that are currently bursting at a specific point in time. The lower middle graph displays the general activity of all spikes on all channels. The bottom graph shows the general activity of all spikes that are part of a single channel burst. This graph, in combination with all individually detected single channel bursts, will be used for the burst detection.
The light green vertical line shows the network burst core, while the darker green line shows the network burst edges.

Lastly, the **Electrode activity** tab shows the activity of every single electrode on this particular well, plotted above each other.

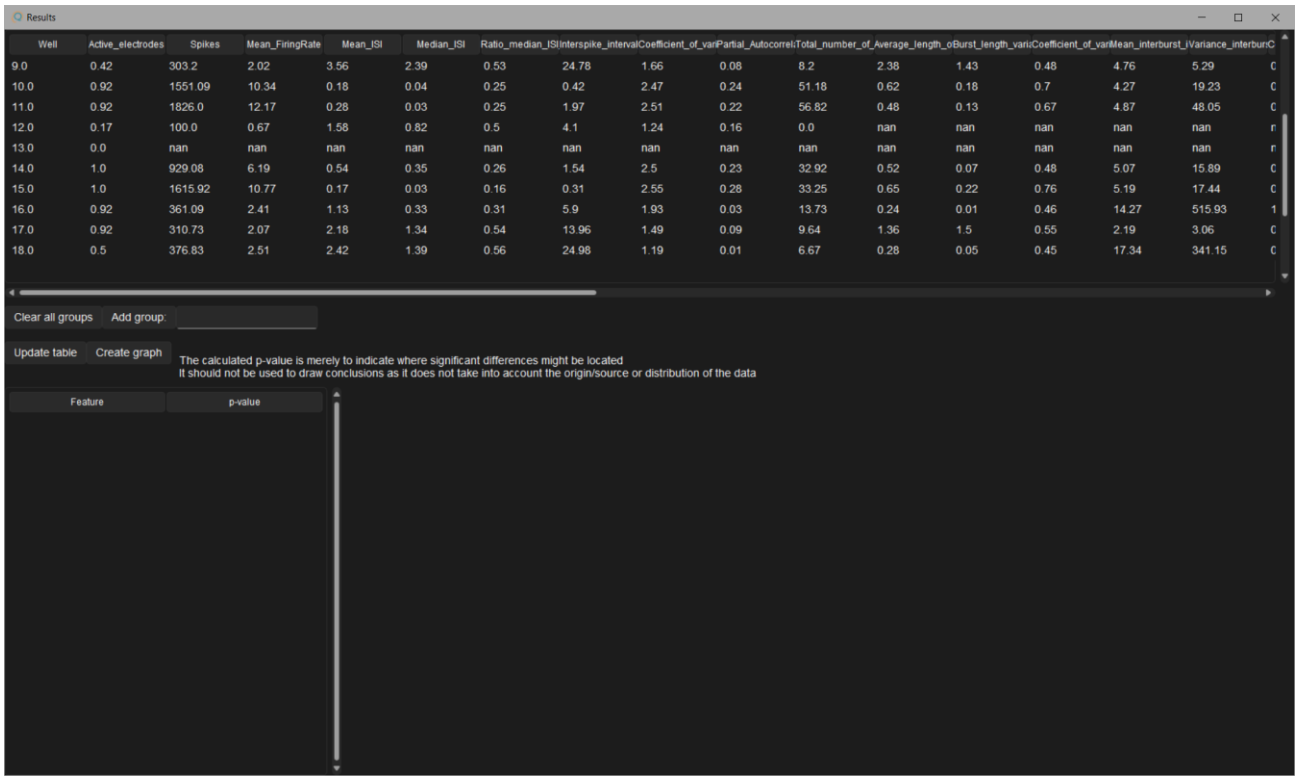


Results/table

In the **Results** tab, it is possible to load a new window, which contains a table with a short overview of the calculated features.



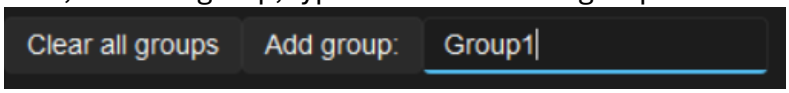
Make sure the right file is selected, then press **Load table**.



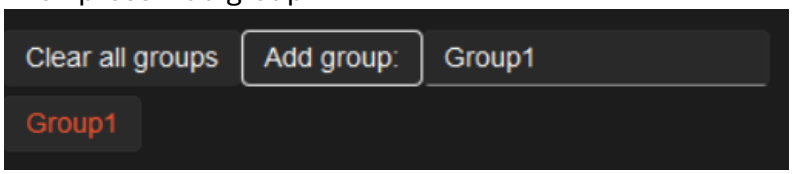
At the top of the window, there is a table that contains a row for each well, and a column for each feature. This data can also be found in the output folder of the algorithm and is called ‘Features.csv’.

Creating boxplots

The main functionality of this window is to get a short overview of the calculated features, and quickly compare several groups against each other.
First, to add a group, type the name of the group in the empty text field:



Then press ‘Add group’:



Repeat this process for any other groups, for a maximum of 10 groups.

Next, the user can select the desired group, and then click on the rows to label them.

Well	Active_electrodes	Spikes	Mean_FiringRate	Mean_ISI
8.0	0.25	187.0	1.25	3.46
9.0	0.42	303.2	2.02	3.56
10.0	0.92	1551.09	10.34	0.18
11.0	0.92	1826.0	12.17	0.28
12.0	0.17	100.0	0.67	1.58
13.0	0.0	nan	nan	nan
14.0	1.0	929.08	6.19	0.54
15.0	1.0	1615.92	10.77	0.17
16.0	0.92	361.09	2.41	1.13
17.0	0.92	310.73	2.07	2.18

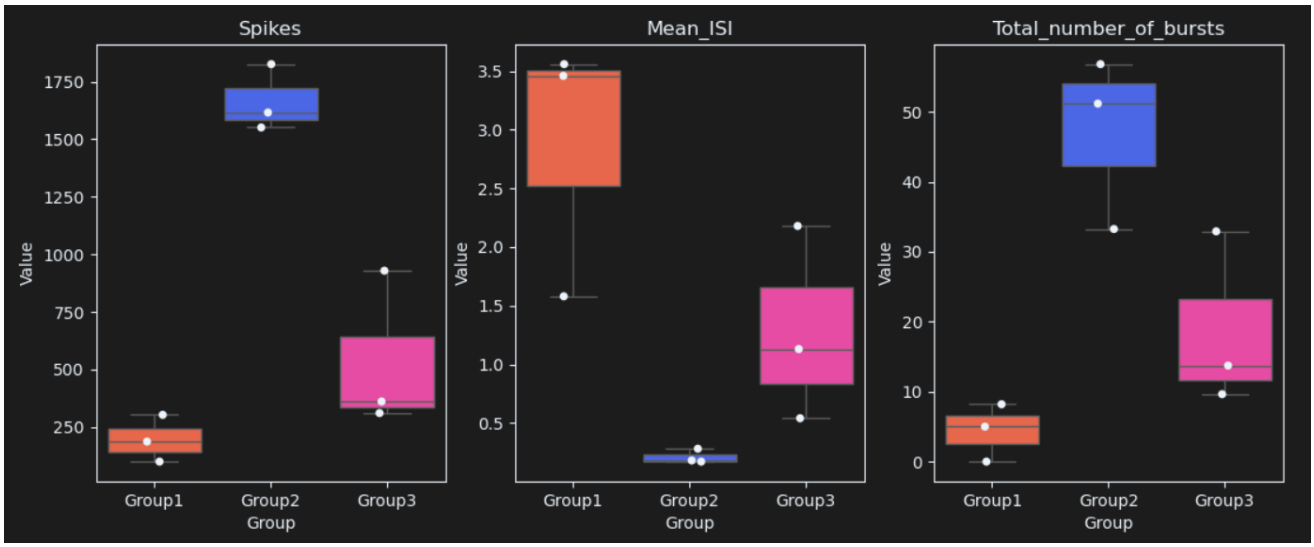
Clear all groups | Add group: Group3

Group1 Group2 Group3

Click ‘Update table’ to compare these groups against each other and calculate a p-value. Please note that this p-value is solely to indicate where significant differences might lie. These p-value calculations do not consider the origin and distribution of the data. Always perform a separate thorough statistical analysis.

Feature	P_value (ANOVA)
Active_electrodes	8.651705690922505e-05
Spikes	0.00046564209970292815
Mean_FiringRate	0.0004633239396259531
Mean_ISI	0.018735428486523726
Median_ISI	0.029167846413188425
Ratio_median_ISI_over_mean_	0.017627833205984208
Interspike_interval_variance	0.12471228240762877
Coefficient_of_variation_ISI	0.015140032447317408
Partial_Autocorrelaction_Func	0.08326262962962971
Total_number_of_bursts	0.0064199060524015176

Finally, select the features to be compared against each other, and press ‘Create graph’.



The application will generate a boxplot for each of the features, and compare all groups to each other.

Parameter overview

Required Parameters

File selection

This tool only supports MEA files in the hdf5 (.h5) format that have been generated by the Multi Channel DataManager, or Raw Axion Biosystems files that have been converted to hdf5 using our custom MATLAB script, which can be found further in this document.
GUI: Using the Choose a file button, the user can browse their personal machine, and select the file the want to analyse.
Library: Give the path of the file that should be analysed to the 'analyse_well' function using the parameter 'fileadress':
`fileadress="C:/Users/User/Documents/MEA_data/example_recording.h5"`

Sampling rate

The sampling rate is the frequency with which the MEA takes measurements. Common values for this are 10000, 12500 and 20000 Hz.
GUI: Sampling rate field
Library: Using the 'hertz' parameter: `hertz=20000`

Electrode amount

The Electrode amount is the number of electrodes that each well contains. Common values for this are 12 or 16.
GUI: Electrode amount field.
Library: Using the 'electrode_amnt' parameter: `electrode_amnt=12`

Filter

Low cutoff

The Low cutoff is the high-pass value that will be used for the bandpass filter.
GUI: Low cutoff field.
Library: Using the 'low_cutoff' parameter: `low_cutoff=200`
Default value: 200 Hz

High cutoff

The High cutoff is the low-pass value that will be used for the bandpass filter.
GUI: High cutoff field.
Library: Using the 'high_cutoff' parameter: `high_cutoff=3500`
Default value: 3500 Hz

Filter order

The Filter order is the order that will be used for the Butterworth bandpass filter.
GUI: Filter order field
Library: Using the 'order' parameter: `order=2`
Default value: 2

Spike detection

For more information on spike detection, see ‘Methods’ section.

Threshold portion

The portion of the raw data that will be used to calculate the threshold. A higher value will lead to a slightly more accurate threshold, but a slower processing time.
GUI: Threshold portion field.
Library: Using the ‘threshold_portion’ parameter: `threshold_portion=0.1`
Default value: 0.1 (10%)

Standard deviation multiplier

The value that will be multiplied with the standard deviation of the raw data segment. Used to determine whether a segment contains spike activity or pure noise.
GUI: Standard deviation multiplier field.
Library: Using the ‘stdevmultiplier’ parameter: `stdevmultiplier=5`
Default value: 5

RMS multiplier

Root Mean Square (RMS) multiplier. Multiplied with the identified noise to calculate the threshold.
GUI: RMS multiplier field.
Library: Using the ‘rmsmultiplier’ parameter: `rmsmultiplier=5`
Default value: 5

Refractory period

The refractory period of the spike. The time in which only one spike can be detected. Value should be given in seconds
GUI: Refractory period field.
Library: Using the ‘spikeduration’ parameter: `spikeduration=0.001`
Default value: 0.001 s

Spike validation method

The method that should be used to validate the detected spikes. Options: ‘DMP_noisebased’ or ‘none’. The first option is the method as described earlier in this document, the second option will not perform any validation.
GUI: Spike validation method dropdown menu.
Library: Using the ‘validation_method’ parameter: `validation_method="DMP_noisebased"`
Default value: DMP_noisebased

Exit time

See “Methods - Spike validation” section. Value should be given in seconds.
GUI: Exit time field.
Library: Using the ‘exit_time_s’ parameter: `exit_time_s=0.00024`
Default value: 0.00024 s

Drop amplitude
See “Methods - Spike validation” section.
GUI: Drop amplitude field.
Library: Using the ‘drop_amplitude_sd’ parameter: `drop_amplitude_sd=5`
Default value: 5

Height exception
See “Methods - Spike validation” section.
GUI: Height exception field.
Library: Using the ‘heightexception’ parameter: `heightexception=1.5`
Default value: 1.5

Max drop amount
See “Methods - Spike validation” section.
GUI: Max drop amount field.
Library: Using the ‘max_drop_amount’ parameter: `max_drop_amount=2`
Default value: 2

Burst detection

For more information on burst detection, see ‘Methods’ section.

Minimal amount of spikes
The minimal amount of spikes that a burst can consist of.
GUI: Minimal amount of spikes field.
Library: Using the ‘minspikes_burst’ parameter: `minspikes_burst=5`
Default value: 5

Default interval threshold
The default inter-spike-interval (ISI) threshold used for burst detection. Value should be given in milliseconds.
GUI: Default interval threshold field.
Library: Using the ‘default_threshold’ parameter: `default_threshold=100`
Default value: 100 ms

Max interval threshold
The maximum value the second ISI threshold of the burst detection method can be. Value should be given in milliseconds.
GUI: Max interval threshold field.
Library: Using the ‘max_threshold’ parameter: `max_threshold=1000`
Default value: 1000 ms

KDE bandwidth
The bandwidth of the Kernel Density Estimate (KDE) used for determining the burst detection parameters.
GUI: KDE bandwidth field
Library: Using the ‘kde_bandwidth’ parameter: `kde_bandwidth=1`

Default value: 1

Smaller neighbours
Number of lower-value neighbouring values a peak in de KDE should have, before being considered as a peak.
GUI: Smaller neighbours field
Library: Using the 'smallerneighbours' parameter: `smallerneighbours=10`
Default value: 10

Network burst detection

For more information on network burst detection, see 'Methods' section.

Min channels
The minimal portion of channels a network burst should be active on.
GUI: Min channels field.
Library: Using the 'min_channels' parameter: `min_channels=0.5`
Default value: 0.5 (50%)

Thresholding method
The method that is used to calculate the threshold for network burst detection. Options: Yen or Otsu.
GUI: Thresholding method dropdown menu.
Library: Using the 'threshold_method' parameter: `threshold_method="Yen"`
Default value: Yen

Other

Remove inactive electrodes
Remove inactive electrodes from the measurements, affects feature calculation.
GUI: Remove inactive electrodes checkbox.
Library: Using the 'remove_inactive_electrodes' parameter:
`remove_inactive_electrodes=True`
Default value: True

Activity threshold
The activity threshold, electrodes with a lower activity than this will be removed. Value should be given in spikes/s. E.g. 0.1 means that electrodes with less than one spike every 10 seconds will be removed.
GUI: Activity threshold field.
Library: Using the 'activity_threshold' parameter: `activity_threshold=0.1`
Default value: 0.1

Split data
Possibility to split the data into smaller parts. **Warning:** if this option is selected, it will not be possible to view the (interactive) spike, burst and network burst detection process afterwards.

GUI: Split data checkbox.
Library: Using the 'cut_data_bool' parameter: `cut_data_bool=False`
Default value: False

Parts

The number of parts the original data should be cut into. For example, if a measurement of 20 minutes is cut up into 10 parts, it will create 10 measurements of 2 minutes. In the output file (.csv file with features) of the algorithm well 1 will be well 1-10, well 2 will be well 11-20 and so on.
GUI: Parts field.
Library: Using the 'parts' parameter: `parts=10`
Default value: 10

Use multiprocessing

Whether the algorithm should utilize multiprocessing to potentially speed up the analysis or not. The algorithm will create a subprocess for every electrode (so 12 electrodes per well = 12 processes) allowing the raw data to be analysed in paralel. Using this option is only recommended if there is enough RAM available. In general, 8 times the (compressed) raw file size should be enough RAM to support multiprocessing.

While turning this option on when using the python library, always make sure the function call is in an 'if __name__ == "__main__":' statement, as follows:

```
if __name__=='__main__':
    analyse_well(fileadress="test.h5", hertz=20000, electrode_amnt=12)
```

Without this statement, the algorithm will slowly create an infinite number of processes, eventually crashing the algorithm/computer.

GUI: Use multiprocessing checkbox.
Library: Using the 'use_multiprocessing' parameter: `use_multiprocessing=False`
Default value: False

Axion file conversion using MATLAB

Data generated from an Axion Biosystems MEA will be saved in the .raw format. Unfortunately, it is not possible to open this format in python. For the MEA-library to analyse these files, they must first be converted to the hdf5 format. The only way to do so, is to read the .raw files in MATLAB using the AxionFileLoader.
The script below will read the raw voltage data from the file, and save them in a new hdf5 file. After this, the hdf5 file can be analysed by the MEA-library. For this script to work, the AxionFileLoader must be installed.

Script:

```
% Create a function that takes the filepath of the original file, and converts
it into the hdf5 format
% Installation of the AxionFileLoader is required for this to work:
https://github.com/axionbio/AxionFileLoader
function raw_to_hdf(filePath)
    % Replace the file extension with .h5
    newExtension = '.h5';
    [filePathNoExt, fileName, ~] = fileparts(filePath);
    hdf5_path = fullfile(filePathNoExt, [fileName newExtension]);
    disp(hdf5_path)

    % Load in the MEA data using the Axion file loader
    mea_data = AxisFile(filePath).RawVoltageData.LoadData;
    mea_size=size(mea_data);

    % Retrieve dimensions of the data to create an h5 dataset
    allelectrodes=mea_size(1)*mea_size(2)*mea_size(3)*mea_size(4);
    measurements=size(mea_data{1,1,1,1}.GetVoltageVector);

    % Create a dataset with the same naming conventions as MCS machines
    % This should allow other tools that read MCS data to read this file
    % aswell, given that they don't have other requirements that the file
    % must adhere to
    dataset_name='/Data/Recording_0/AnalogStream/Stream_0/ChannelData';
    % Alternatively, one can change the interal file structure by
    % uncommenting the next piece of code, and altering the folder
    % structure

    %dataset_name='/Data/Rawdata'

    h5create(hdf5_path,dataset_name,[measurements(1), allelectrodes]);
    h5disp(hdf5_path);

    % Loop through all the wells from top left to bottom right (for both wells
and electrodes, similar to
    % how it works with MCS hdf5 files
```

```
    counter=0;
    for verticalwells = 1:mea_size(1)
        for horizontalwells = 1:mea_size(2)
            % For some reason, the rows are counted from bottom to top in
            % Axis software, so i do the same here
            for verticalelectrodes = mea_size(3):-1:1
                for horizontalelectrodes = 1:mea_size(4)
                    % Axion indexing: Vertical wells, Horizontal wells,
                    % Horizontal electrodes, Vertical electrodes
                    raw_electrode_data = mea_data{verticalwells,
horizontalwells, horizontalelectrodes, verticalelectrodes}.GetVoltageVector;
                    counter=counter+1;
                    h5write(hdf5_path, dataset_name, raw_electrode_data, [1,
counter], [measurements(1), 1]);
                    disp(['Converted electrode ', num2str(counter), ' out of
', num2str(allelectrodes)]);
                end
            end
        end
    end
    disp('done')
end

% Specify the file path here
path='G:/MEADATA/tsc/TSC_20DIV.raw';
raw_to_hdf(path);

% This script loops through the axion data from the top left well to the
% bottom right, and from the top left electrode to the bottom right. This
% means it creates an array of wells*electrodes rows and samples amount of
% columns.
% In the case of a 4*6 (24) well with a 4*4 (16) electrode (so 24*16=384)
% configuration  and 3000000 samples, the array size would be 384*3000000
%
% This script has been tested and verified using a 24 well 16 electrode axion
plate
```

After the conversion has been completed, these files are not compressed yet, and might take up a lot of storage space. Hdf5 files can be compresses using the following tool: https://support.hdfgroup.org/HDF5/doc/RM/Tools.html#Tools-Repack. Depending on the GZIP level, files might shrink up to 6 times in size.

Literature

Bakkum, D., Radivojevic, M., Frey, U., Franke, F., Hierlemann, A., & Takahashi, H. (2014). Parameters for burst detection. *Frontiers in Computational Neuroscience*, *7*. https://www.frontiersin.org/articles/10.3389/fncom.2013.00193

Belitski, A., Gretton, A., Magri, C., Murayama, Y., Montemurro, M. A., Logothetis, N. K., & Panzeri, S. (2008). Low-Frequency Local Field Potentials and Spikes in Primary Visual Cortex Convey Independent Visual Information. *The Journal of Neuroscience*, *28*(22), 5696–5709. https://doi.org/10.1523/JNEUROSCI.0009-08.2008

Black, B. J., Atmaramani, R., Kumaraju, R., Plagens, S., Romero-Ortega, M., Dussor, G., Price, T. J., Campbell, Z. T., & Pancrazio, J. J. (2018). Adult mouse sensory neurons on microelectrode arrays exhibit increased spontaneous and stimulus-evoked activity in the presence of interleukin-6. *Journal of Neurophysiology*, *120*(3), 1374–1385. https://doi.org/10.1152/jn.00158.2018

Bradley, J. A., Luithardt, H. H., Metea, M. R., & Strock, C. J. (2018). In Vitro Screening for Seizure Liability Using Microelectrode Array Technology. *Toxicological Sciences*, *163*(1), 240–253. https://doi.org/10.1093/toxsci/kfy029

Bullmann, T., Radivojevic, M., Huber, S. T., Deligkaris, K., Hierlemann, A., & Frey, U. (2019). Large-Scale Mapping of Axonal Arbors Using High-Density Microelectrode Arrays. *Frontiers in Cellular Neuroscience*, *13*. https://www.frontiersin.org/articles/10.3389/fncel.2019.00404

Burgess, R. C. (2019). Chapter 4—Filtering of neurophysiologic signals. In K. H. Levin & P. Chauvel (Eds.), *Handbook of Clinical Neurology* (Vol. 160, pp. 51–65). Elsevier. https://doi.org/10.1016/B978-0-444-64032-1.00004-7

Cotterill, E., Charlesworth, P., Thomas, C. W., Paulsen, O., & Eglen, S. J. (2016). A comparison of computational methods for detecting bursts in neuronal spike trains and their application to human stem cell-derived neuronal networks. *Journal of Neurophysiology*, *116*(2), 306–321. https://doi.org/10.1152/jn.00093.2016

Crosse, M. J., Zuk, N. J., Di Liberto, G. M., Nidiffer, A. R., Molholm, S., & Lalor, E. C. (2021). Linear Modeling of Neurophysiological Responses to Speech and Other Continuous Stimuli: Methodological Considerations for Applied Research. *Frontiers in Neuroscience*, *15*. https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2021.705621

De Cheveigné, A., & Nelken, I. (2019). Filters: When, Why, and How (Not) to Use Them. *Neuron*, *102*(2), 280–293. https://doi.org/10.1016/j.neuron.2019.02.039

Dolan, K., Martens, H. C. F., Schuurman, P. R., & Bour, L. J. (2009). Automatic noise-level detection for extra-cellular micro-electrode recordings. *Medical & Biological Engineering & Computing*, *47*(7), 791–800. https://doi.org/10.1007/s11517-009-0494-4

Frega, M., Linda, K., Keller, J. M., Gümüş-Akay, G., Mossink, B., van Rhijn, J.-R., Negwer, M., Klein Gunnewiek, T., Foreman, K., Kompier, N., Schoenmaker, C., van den Akker, W., van der Werf, I., Oudakker, A., Zhou, H., Kleefstra, T., Schubert, D., van Bokhoven, H., & Nadif Kasri, N. (2019). Neuronal network dysfunction in a model for Kleefstra syndrome mediated by enhanced NMDAR signaling. *Nature Communications*, *10*(1), 4928. https://doi.org/10.1038/s41467-019-12947-3

Gelfman, S., Wang, Q., Lu, Y.-F., Hall, D., Bostick, C. D., Dhindsa, R., Halvorsen, M., McSweeney, K. M., Cotterill, E., Edinburgh, T., Beaumont, M. A., Frankel, W. N., Petrovski, S., Allen, A. S., Boland, M. J., Goldstein, D. B., & Eglen, S. J. (2018). meaRtools: An R package for the analysis of neuronal networks recorded on microelectrode arrays. *PLOS Computational Biology*, *14*(10), e1006506. https://doi.org/10.1371/journal.pcbi.1006506

Haq, W., Zrenner, E., Ueffing, M., & Paquet-Durand, F. (2023). Using Micro-Electrode-Array Recordings and Retinal Disease Models to Elucidate Visual Functions: Simultaneous Recording of Local Electroretinograms and Ganglion Cell Action Potentials Reveals the Origin of Retinal Oscillatory Potentials. *Bioengineering*, *10*(6), 725. https://doi.org/10.3390/bioengineering10060725

Home-CureQ – CureQ: Voorspellen, vertragen en genezen van erfelijke hersenziektes. (n.d.). Retrieved 6 March 2024, from https://cureq.nl/

Hornauer, P., Prack, G., Anastasi, N., Ronchi, S., Kim, T., Donner, C., Fiscella, M., Borgwardt, K., Taylor, V., Jagasia, R., Roqueiro, D., Hierlemann, A., & Schröter, M. (2024). DeePhys: A machine learning-assisted platform for electrophysiological phenotyping of human neuronal networks. *Stem Cell Reports*, S2213-6711(23)00501-5. https://doi.org/10.1016/j.stemcr.2023.12.008

Hu, M., Frega, M., Tolner, E. A., van den Maagdenberg, A. M. J. M., Frimat, J. P., & le Feber, J. (2022). MEA-ToolBox: An Open Source Toolbox for Standardized Analysis of Multi-Electrode Array Data. *Neuroinformatics*, *20*(4), 1077–1092. https://doi.org/10.1007/s12021-022-09591-6

Kapucu, F. E., Vinogradov, A., Hyvärinen, T., Ylä-Outinen, L., & Narkilahti, S. (2022). Comparative microelectrode array data of the functional development of hPSC-

derived and rat neuronal networks. *Scientific Data*, *9*(1), Article 1. https://doi.org/10.1038/s41597-022-01242-4

Lieb, F., Stark, H.-G., & Thielemann, C. (2017). A stationary wavelet transform and a time-frequency based spike detection algorithm for extracellular recorded data. *Journal of Neural Engineering*, *14*(3), 036013. https://doi.org/10.1088/1741-2552/aa654b

Lv, S., He, E., Luo, J., Liu, Y., Liang, W., Xu, S., Zhang, K., Yang, Y., Wang, M., Song, Y., Wu, Y., & Cai, X. (2023). Using Human-Induced Pluripotent Stem Cell Derived Neurons on Microelectrode Arrays to Model Neurological Disease: A Review. *Advanced Science*, *10*(33), 2301828. https://doi.org/10.1002/advs.202301828

McConnell, E. R., McClain, M. A., Ross, J., LeFew, W. R., & Shafer, T. J. (2012). Evaluation of multi-well microelectrode arrays for neurotoxicity screening using a chemical training set. *NeuroToxicology*, *33*(5), 1048–1057. https://doi.org/10.1016/j.neuro.2012.05.001

Mendis, G. D. C., Morrisroe, E., Petrou, S., & Halgamuge, S. K. (2016). Use of adaptive network burst detection methods for multielectrode array data and the generation of artificial spike patterns for method evaluation. *Journal of Neural Engineering*, *13*(2), 026009. https://doi.org/10.1088/1741-2560/13/2/026009

Mossink, B., Verboven, A. H. A., van Hugte, E. J. H., Klein Gunnewiek, T. M., Parodi, G., Linda, K., Schoenmaker, C., Kleefstra, T., Kozicz, T., van Bokhoven, H., Schubert, D., Nadif Kasri, N., & Frega, M. (2021). Human neuronal networks on micro-electrode arrays are a highly robust tool to study disease-specific genotype-phenotype correlations in vitro. *Stem Cell Reports*, *16*(9), 2182–2196. https://doi.org/10.1016/j.stemcr.2021.07.001

Müller, J. (2015). *High-Density Microelectrode Array Platform in CMOS Technology* [Doctoral Thesis, ETH Zurich]. https://doi.org/10.3929/ethz-a-010554762

Muthmann, J.-O., Amin, H., Sernagor, E., Maccione, A., Panas, D., Berdondini, L., Bhalla, U. S., & Hennig, M. H. (2015). Spike Detection for Large Neural Populations Using High Density Multielectrode Arrays. *Frontiers in Neuroinformatics*, *9*, 28. https://doi.org/10.3389/fninf.2015.00028

Napoli, A., & Obeid, I. (2016). Comparative Analysis of Human and Rodent Brain Primary Neuronal Culture Spontaneous Activity Using Micro-Electrode Array Technology. *Journal of Cellular Biochemistry*, *117*(3), 559–565. https://doi.org/10.1002/jcb.25312

Negri, J., Menon, V., & Young-Pearse, T. L. (2020). Assessment of Spontaneous Neuronal Activity In Vitro Using Multi-Well Multi-Electrode Arrays: Implications for Assay Development. *eNeuro*, *7*(1), ENEURO.0080-19.2019. https://doi.org/10.1523/ENEURO.0080-19.2019

Nick, C., Goldhammer, M., Bestel, R., Steger, F., Daus, A. W., & Thielemann, C. (2013). DrCell – A Software Tool for the Analysis of Cell Signals Recorded with Extracellular Microelectrodes. *Signal Processing*.

Obien, M. E. J., Deligkaris, K., Bullmann, T., Bakkum, D. J., & Frey, U. (2015). Revealing neuronal function through microelectrode array recordings. *Frontiers in Neuroscience*, *8*. https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2014.00423

Otsu, Nobuyuki. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(1). https://doi.org/10.1109/TSMC.1979.4310076

Pasquale, V., Martinoia, S., & Chiappalone, M. (2010). A self-adapting approach for the detection of bursts and network bursts in neuronal cultures. *Journal of Computational Neuroscience*, *29*(1), 213–229. https://doi.org/10.1007/s10827-009-0175-1

Passaro, A. P., Aydin, O., Saif, M. T. A., & Stice, S. L. (2021). Development of an objective index, neural activity score (NAS), reveals neural network ontogeny and treatment effects on microelectrode arrays. *Scientific Reports*, *11*, 9110. https://doi.org/10.1038/s41598-021-88675-w

Quian Quiroga, R. (2009). What is the real shape of extracellular spikes? *Journal of Neuroscience Methods*, *177*(1), 194–198. https://doi.org/10.1016/j.jneumeth.2008.09.033

Ravalia, A. S., Lau, J., Barron, J. C., Purchase, S. L. M., Southwell, A. L., Hayden, M. R., Nafar, F., & Parsons, M. P. (2021). Super-resolution imaging reveals extrastriatal synaptic dysfunction in presymptomatic Huntington disease mice. *Neurobiology of Disease*, *152*, 105293. https://doi.org/10.1016/j.nbd.2021.105293

Seabold, S., & Perktold, J. (2010). *statsmodels: Econometric and statistical modeling with python*. [Computer software]. Proceedings of the 9th Python in Science Conference.

Swindale, N. V., & Spacek, M. A. (2015). Spike detection methods for polytrodes and high density microelectrode arrays. *Journal of Computational Neuroscience*, *38*(2), 249–261. https://doi.org/10.1007/s10827-014-0539-z

Trujillo, C. A., Gao, R., Negraes, P. D., Gu, J., Buchanan, J., Preissl, S., Wang, A., Wu, W., Haddad, G. G., Chaim, I. A., Domissy, A., Vandenberghe, M., Devor, A., Yeo, G. W., Voytek, B., & Muotri, A. R. (2019). Complex Oscillatory Waves Emerging from Cortical Organoids Model Early Human Brain Network Development. *Cell Stem Cell*, *25*(4), 558-569.e7. https://doi.org/10.1016/j.stem.2019.08.002

Tsai, D., Sawyer, D., Bradd, A., Yuste, R., & Shepard, K. L. (2017). A very large-scale microelectrode array for cellular-resolution electrophysiology. *Nature Communications*, *8*(1), Article 1. https://doi.org/10.1038/s41467-017-02009-x

Wagenaar, D., DeMarse, T. B., & Potter, S. M. (2005). MeaBench: A toolset for multi-electrode data acquisition and on-line analysis. *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.*, 518–521. https://doi.org/10.1109/CNE.2005.1419673

Weglarczyk, S. (2018). Kernel density estimation and its application. *ITM Web of Conferences*, *23*, 00037. https://doi.org/10.1051/itmconf/20182300037

Yen, J. C., Chang, F. J., & Chang, S. (1995). A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, *4*(3), 370–378. https://doi.org/10.1109/83.366472

Zeldenrust, F., Wadman, W. J., & Englitz, B. (2018). Neural Coding With Bursts—Current State and Future Perspectives. *Frontiers in Computational Neuroscience*, *12*. https://doi.org/10.3389/fncom.2018.00048