NeuroExplorer

Release 5.441

Nex Technologies

Mar 21, 2025

CONTENTS

To download the latest version of NeuroExplorer, visit NeuroExplorer Web Site.

If you are new to NeuroExplorer, start with *Tutorials* and *How-to Guides*.

If you need to learn more about analyses, see *Analysis Reference*.

If you need help with scripting, see *Scripting Reference*.

CHAPTER

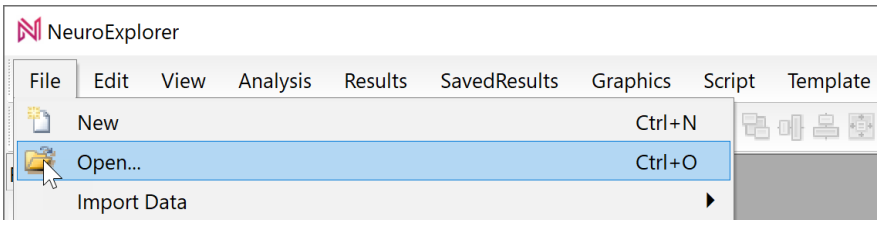
ONE

TUTORIALS

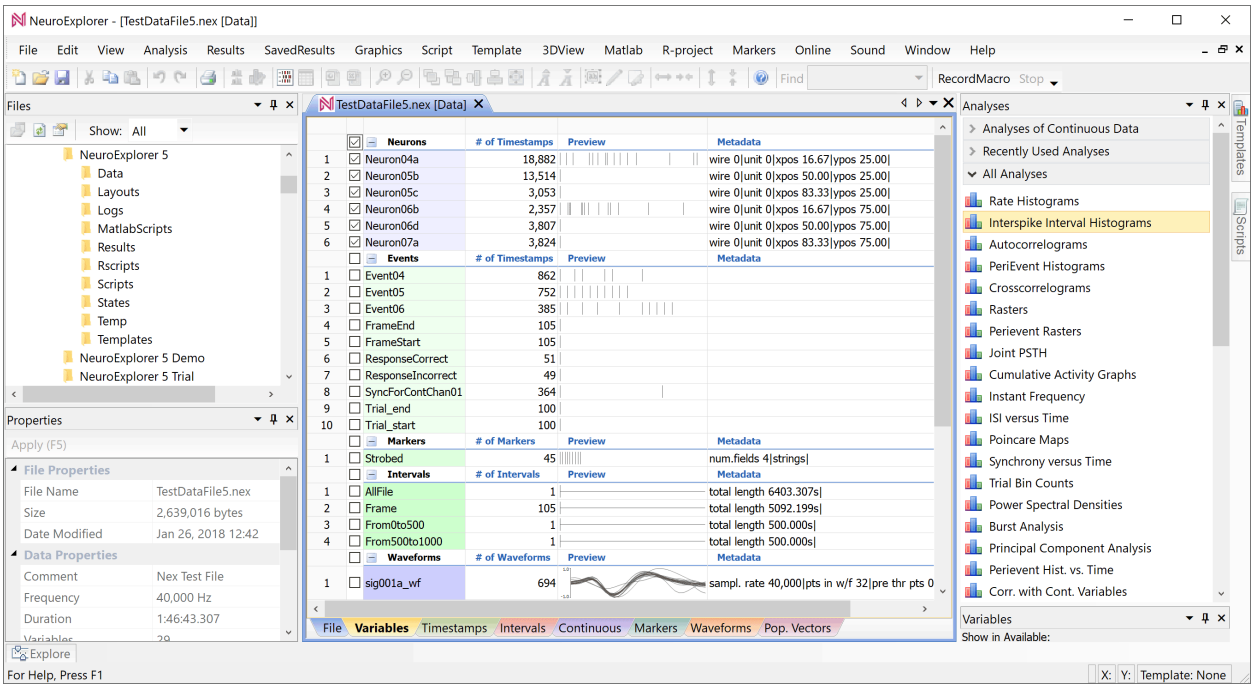
1.1 Look at Your Data in 1D Data Viewer

After you have loaded your data (see *How-to Guides*), you may want to look at the graphical representation of your data.

- Select **File | Open** menu command:



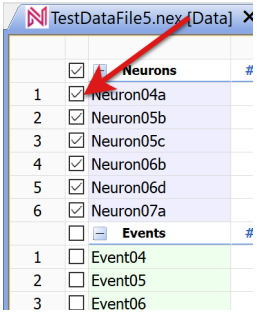
- Select **TestDataFile5.nex** file and click **OK** in the Open dialog. NeuroExplorer will load the test data file:



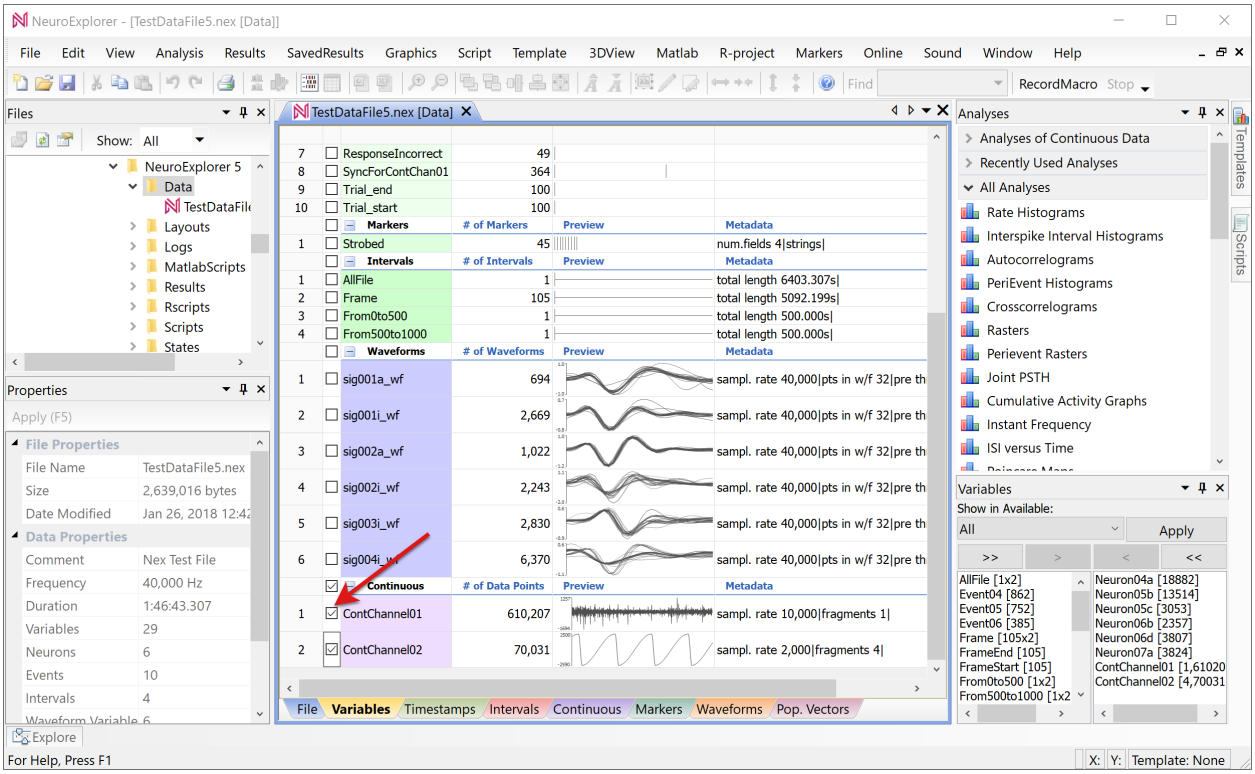
First, we need to select the variables (channels) that you want to view.

The checkbox to the left of a variable name indicates if the variable is selected to be viewed or to be analyzed.

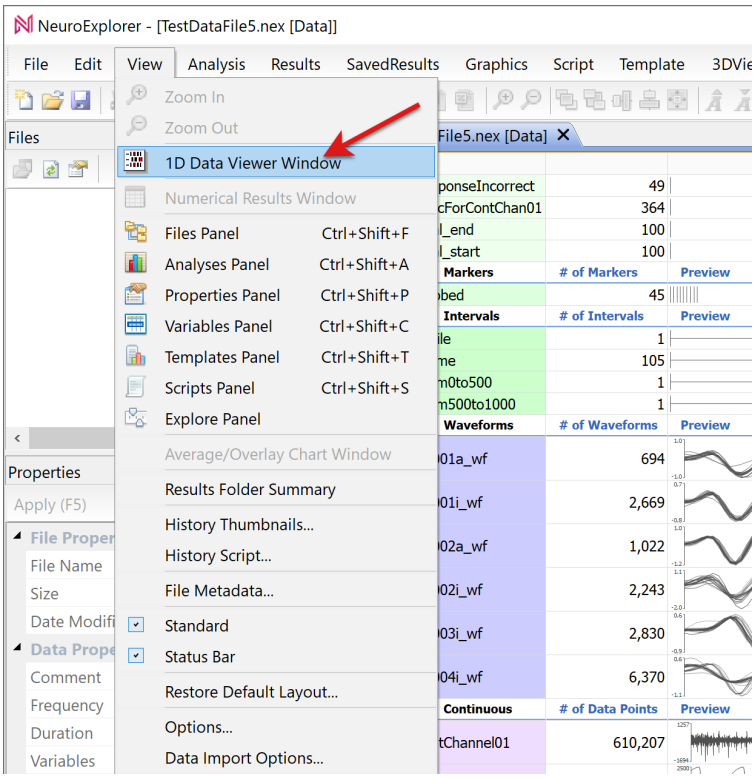
When NeuroExplorer loads a data file, if the Neuron variables are present, NeuroExplorer selects all the Neuron variables:



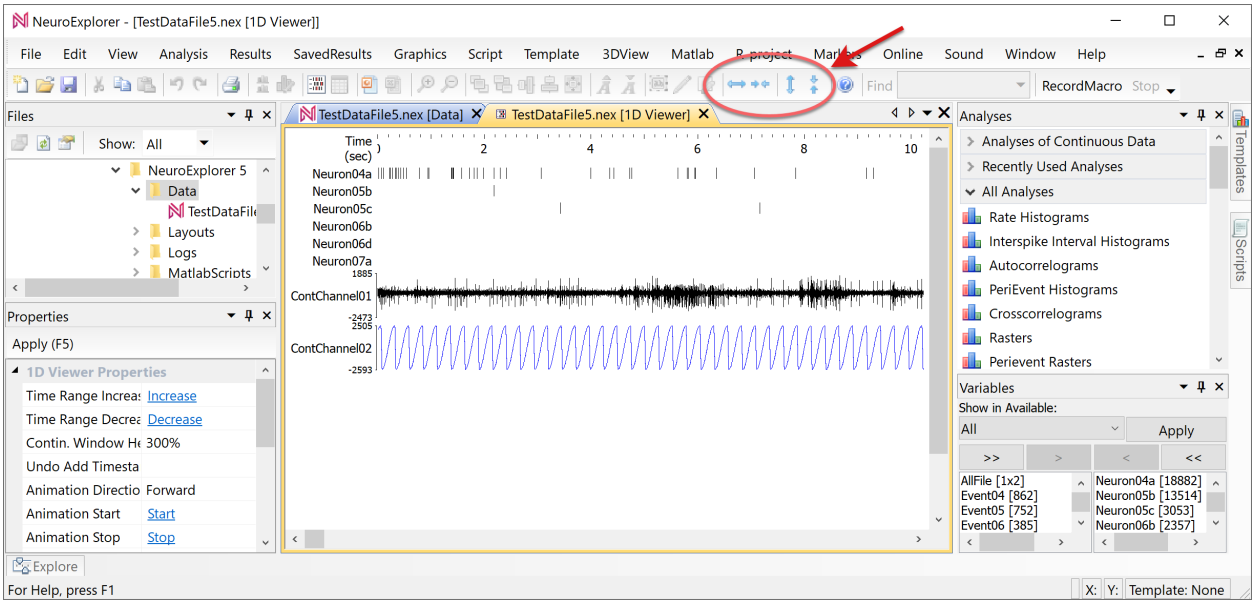
Let's also select continuous variables ContChannel01 and ContChannel02. To do this, click to the left of the channel names:



To view graphical representation of the selected channels, select **View | 1D Data Viewer Window** menu command:



NeuroExplorer will open 1D Viewer window:



Use toolbar buttons with horizontal and vertical arrows to zoom in or zoom out the 1D Viewer.

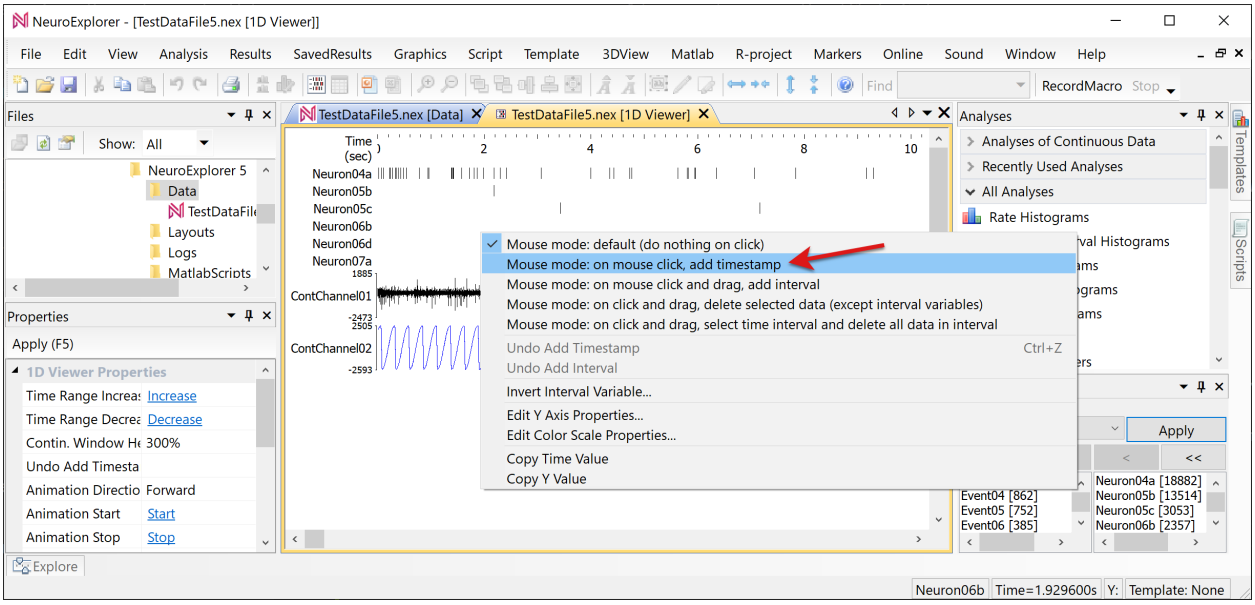
You can use scroll bars or keyboard left and right arrow buttons to scroll the 1D Viewer.

Mouse wheel can also be used to quickly navigate in 1D Viewer:

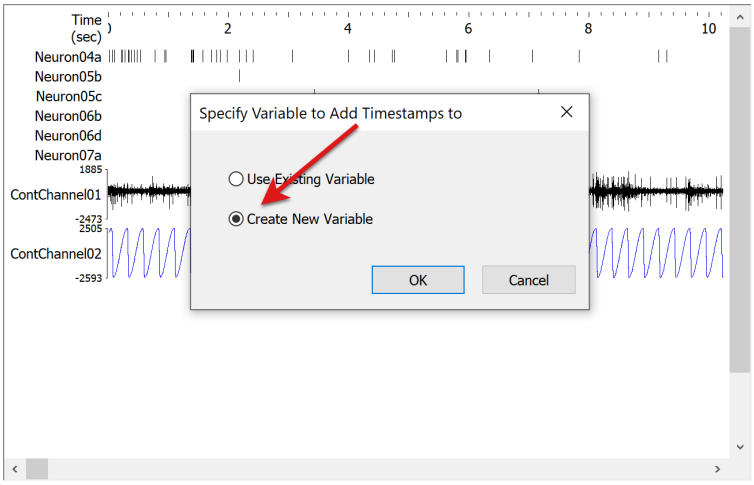
- Click in 1D View to set 1D View as the active window in NeuroExplorer
- Press Shift and rotate mouse wheel to shift 1D View horizontally
- Press Ctrl and rotate mouse wheel to increase the time range (zoom out) and decrease the time range (zoom in)

You can also use 1D Viewer to manually add events and intervals to the data file. For example, to add events:

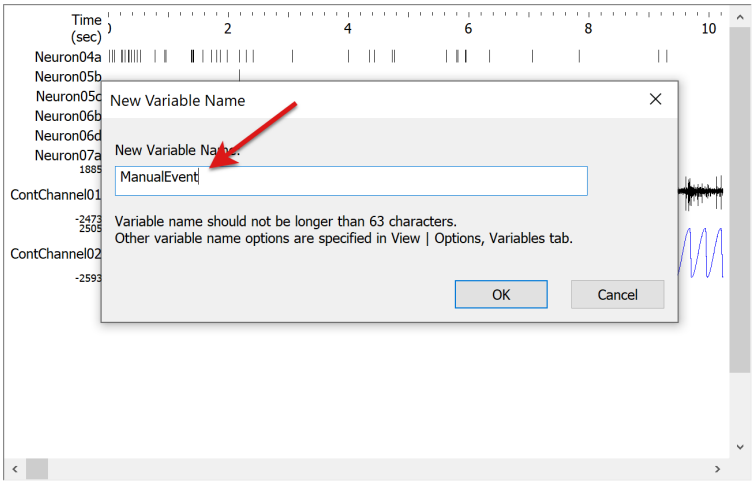
- Right-click in the 1D Viewer
- Select **Mouse mode: on mouse click, add timestamp** menu command



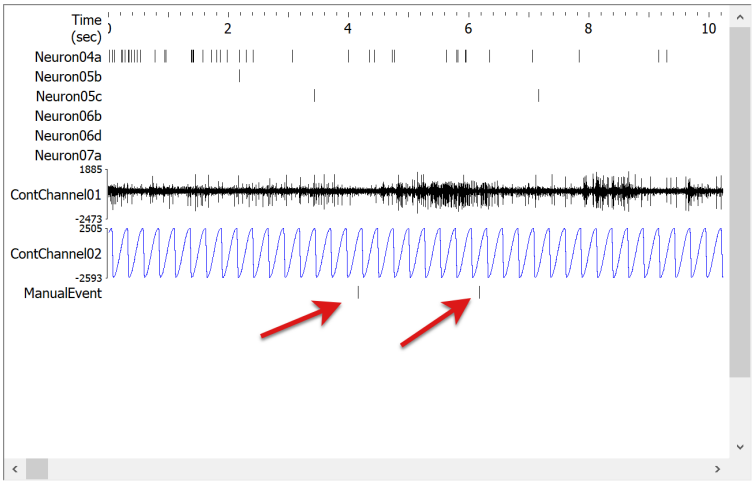
- Select *Create New Variable* option:



- Specify the new event name in the next dialog:



Now, when you press the left mouse button while the mouse pointer is in the 1D Viewer, NeuroExplorer will add a new timestamp to ManualEvent:



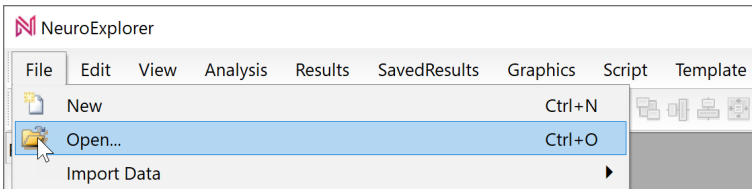
After you are done adding timestamps:

- Right-click in the 1D Viewer
- Select **Mouse mode: default (do nothing click)** menu command

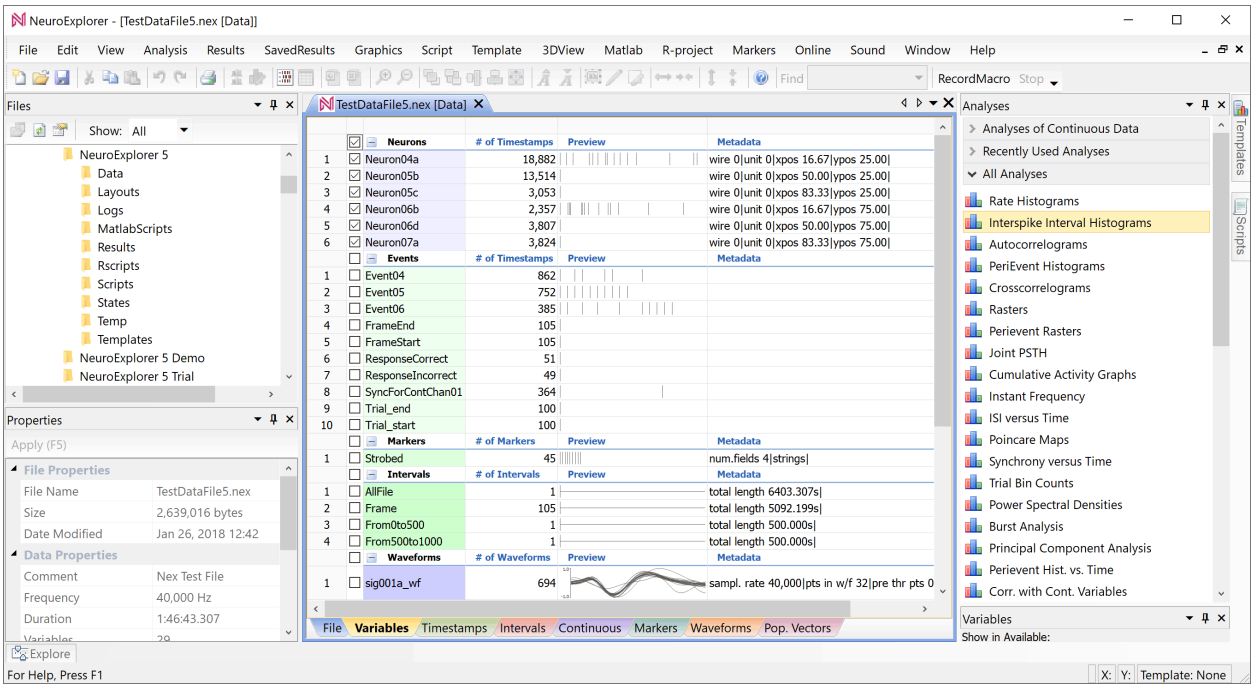
1.2 First Spike Train Analysis

By the end of this tutorial, you will learn the basic analysis workflow in NeuroExplorer. This includes opening a data file, selecting an analysis, running the analysis, viewing numerical results, and saving the results.

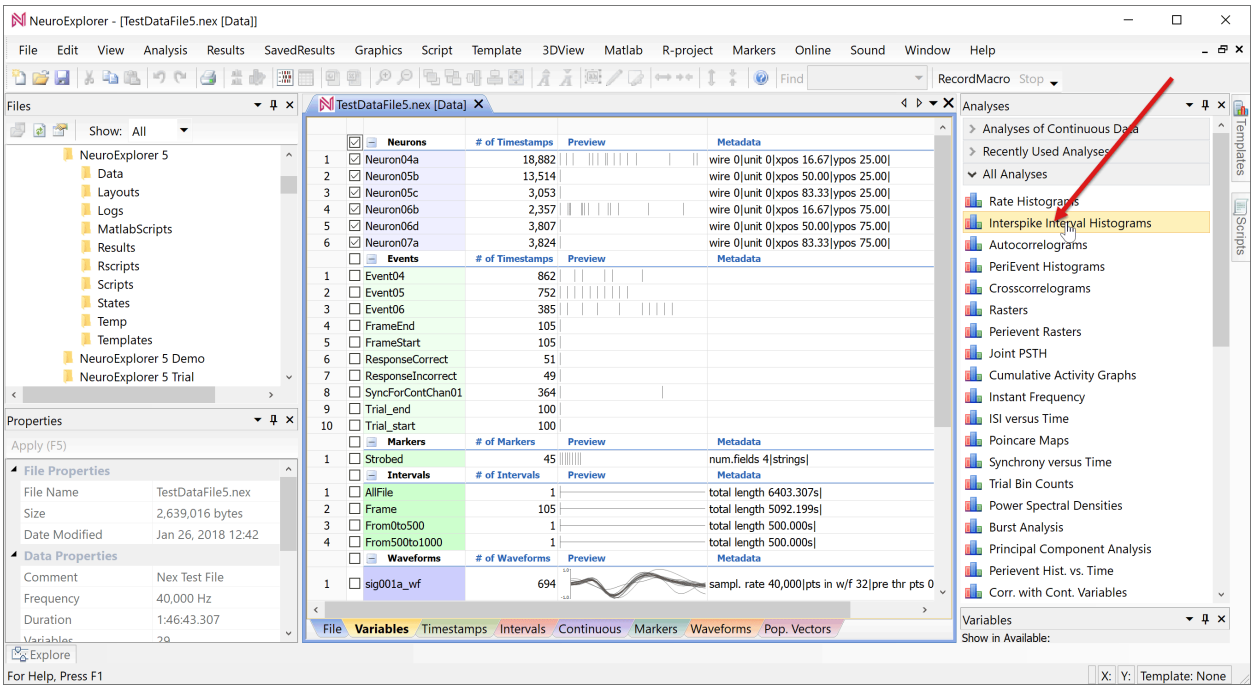
- Select **File | Open** menu command:



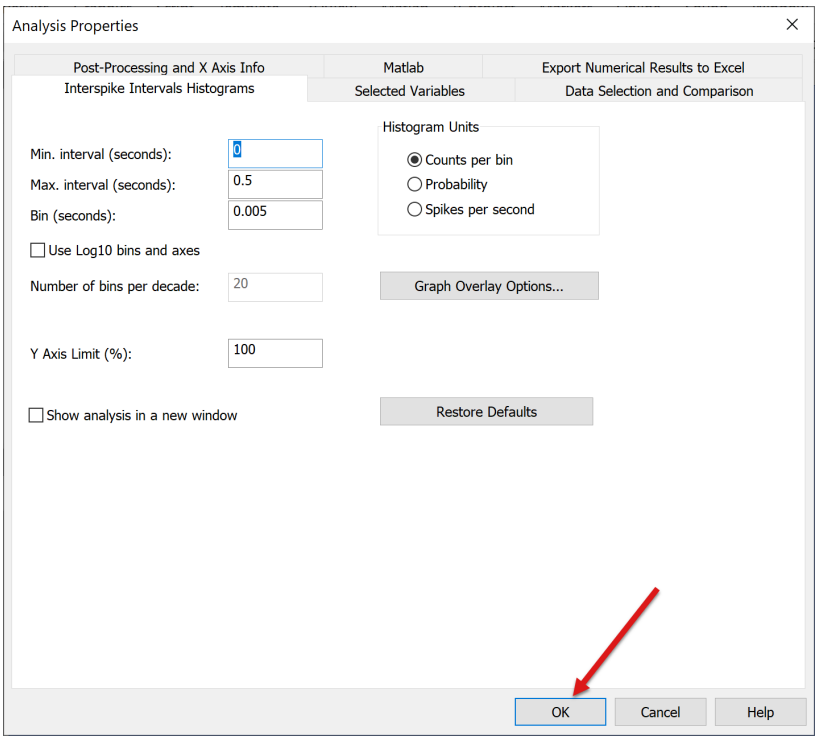
- Select **TestDataFile5.nex** file and click **OK** in the Open dialog. NeuroExplorer will load the test data file:



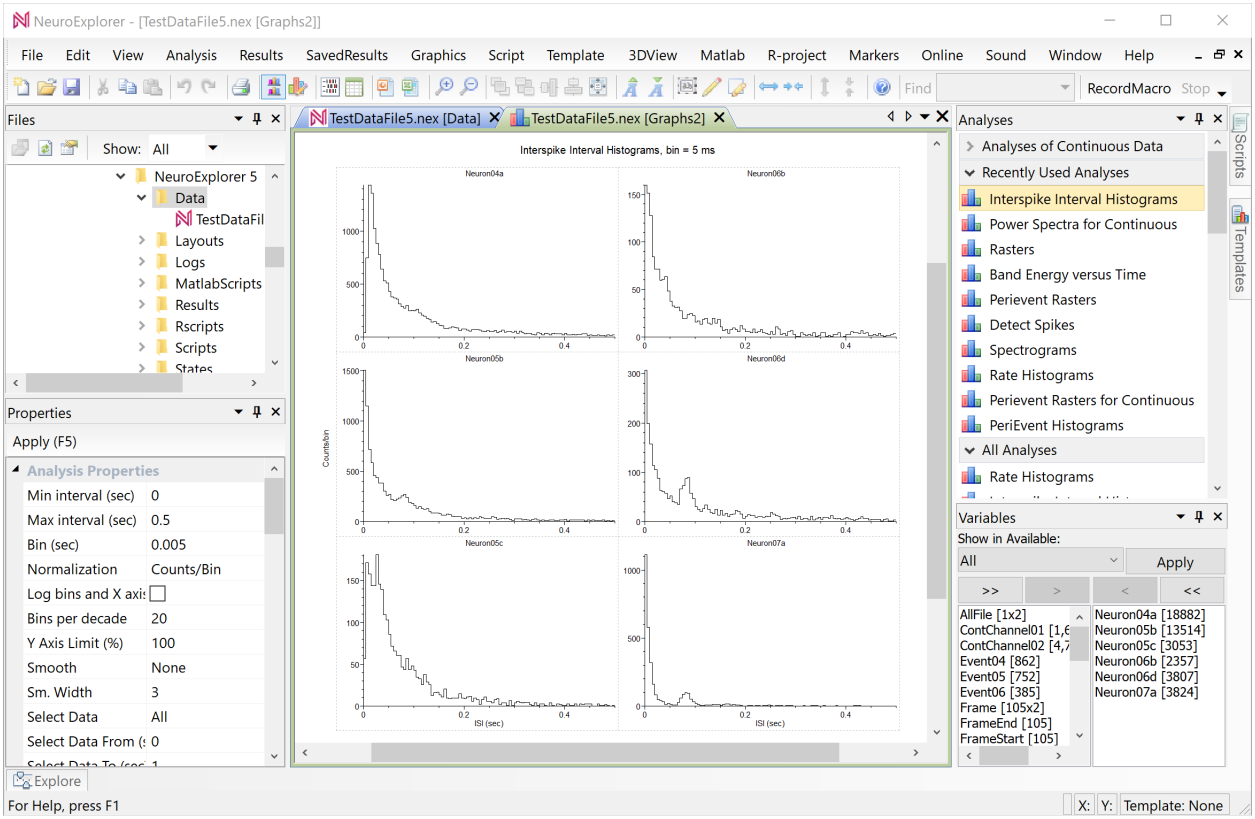
- Select analysis type by clicking the **Interspike Interval Histograms** line in the Analyses panel:



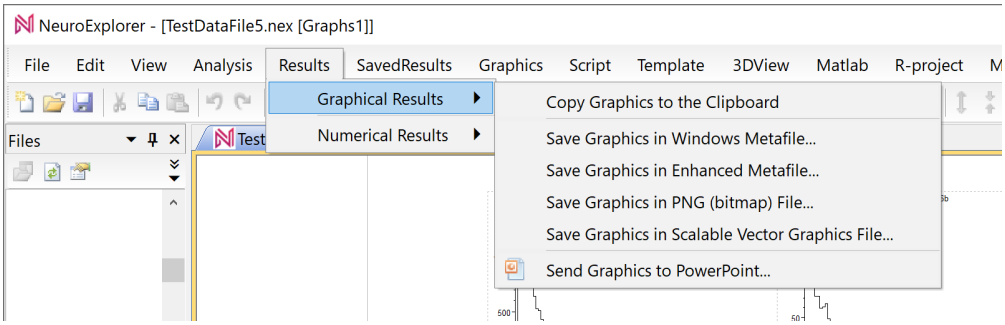
- NeuroExplorer will open **Analysis Properties** dialog. Click **OK** in the dialog:



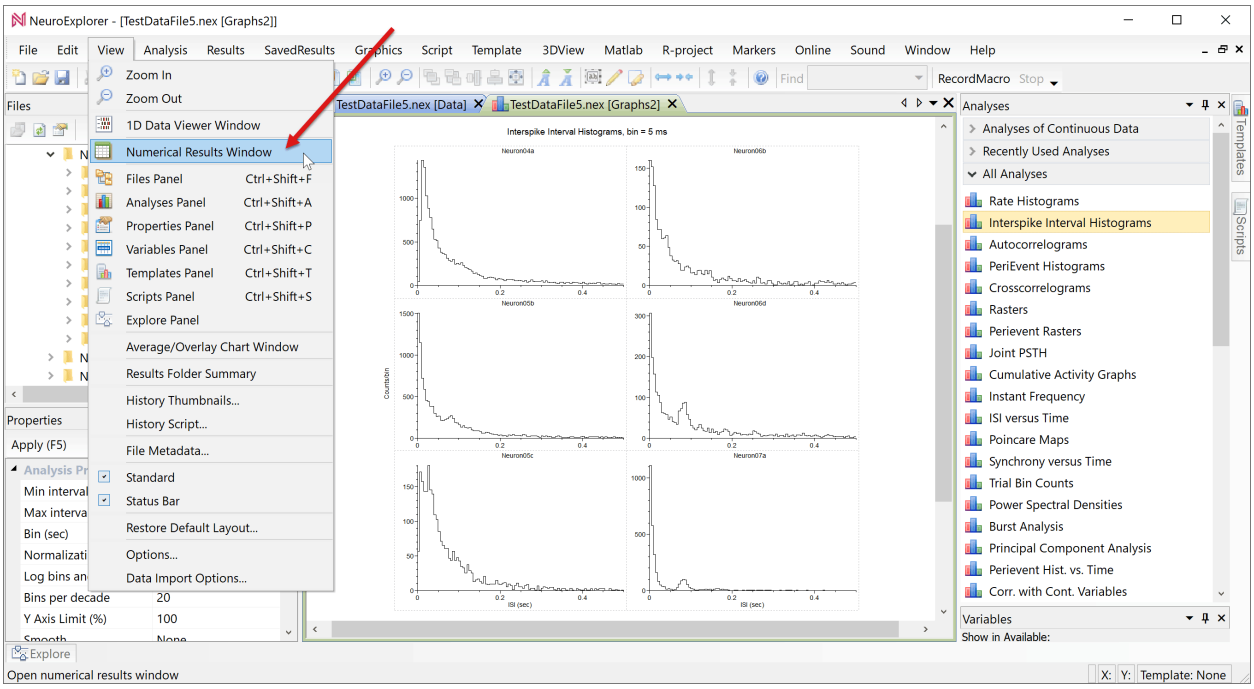
- NeuroExplorer will open a new window with graphical analysis results:



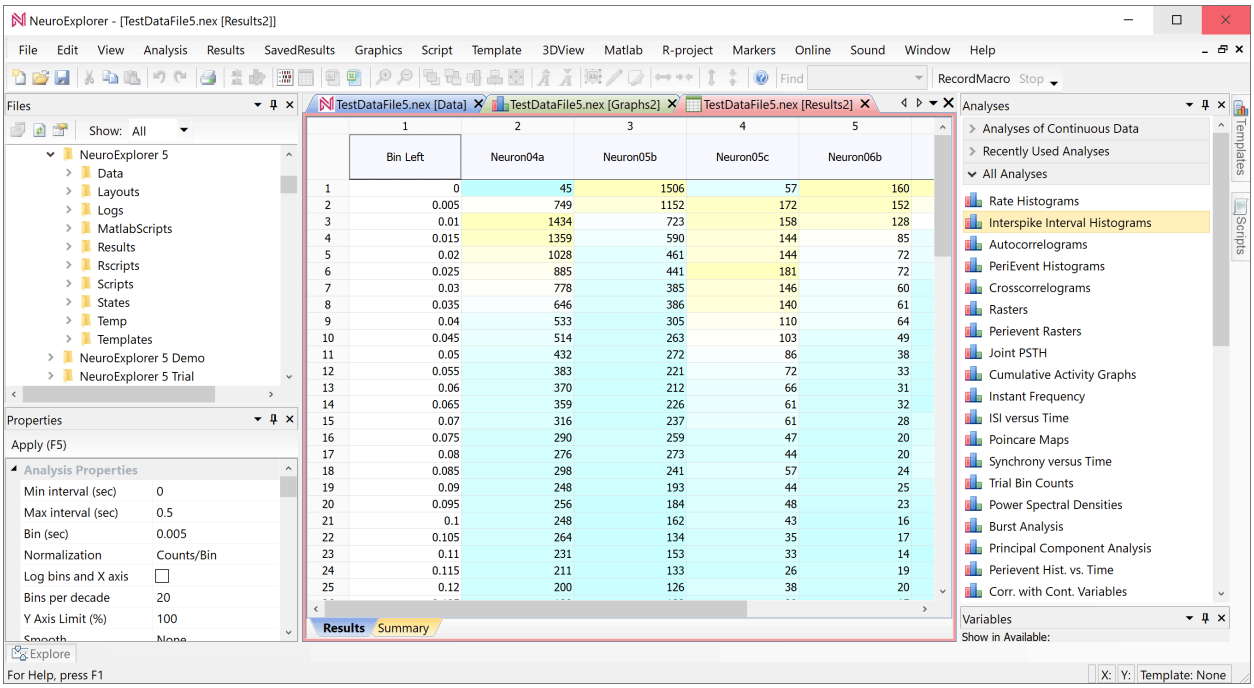
- You can save graphical analysis results using **Results | Graphical Results** menu commands:



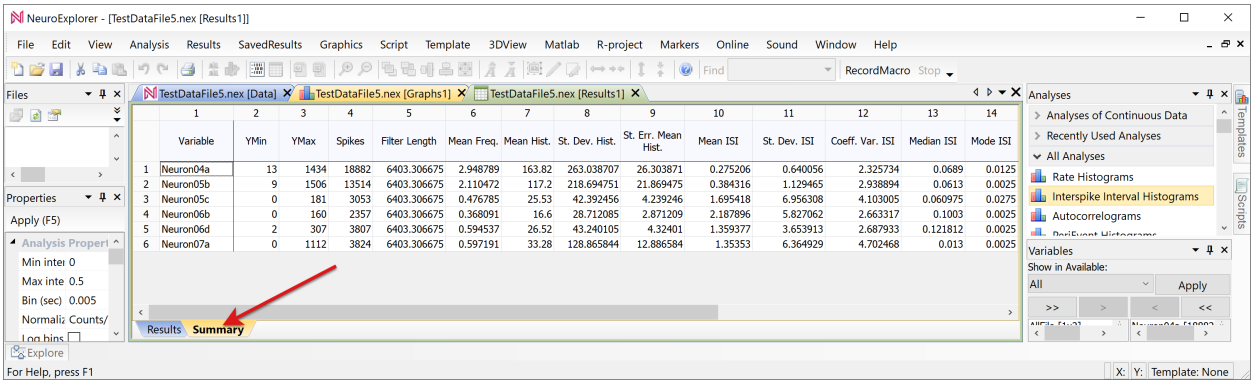
- Select **View | Numerical Results Window** menu command:



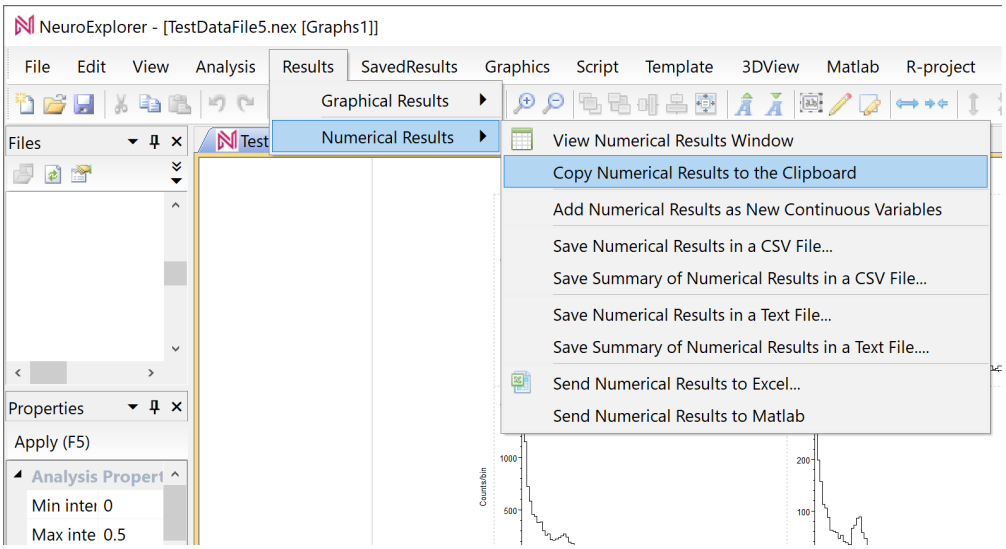
- NeuroExplorer will open a new window with numerical analysis results:



- Click the **Summary** tab at the bottom of the numerical results window. NeuroExplorer will open a new window with the summary statistics such as mean firing rate and median interspike interval:



- You can save numerical analysis results using **Results | Numerical Results** menu commands:



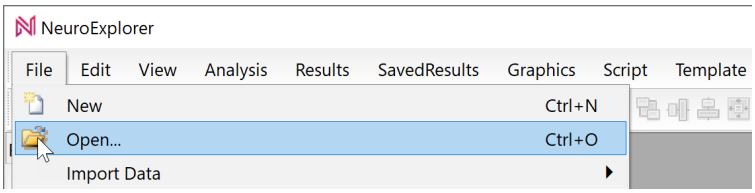
Congratulations! You performed your first data analysis in NeuroExplorer!!

1.3 All Pairwise Correlations for a Group of Neurons

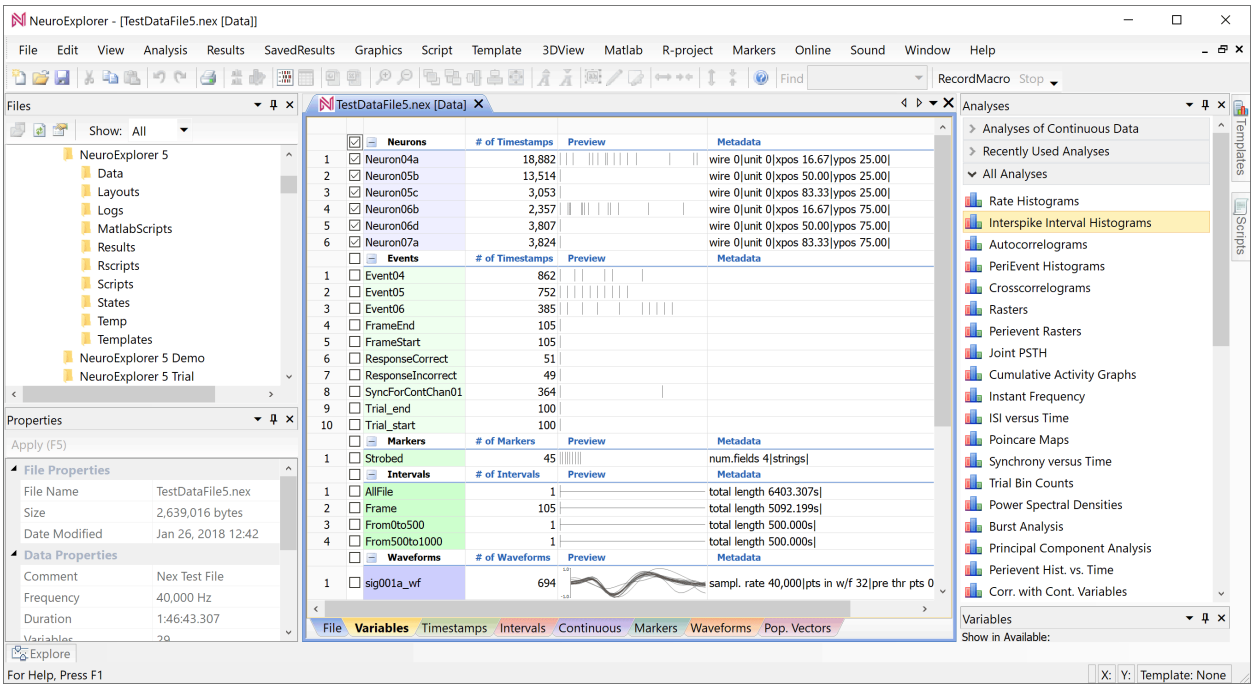
The main idea presented here is that in many analyses you can specify multiple reference events and NeuroExplorer will display the results as rows or columns of graphs.

By the end of this tutorial, you will learn how to calculate all pairwise crosscorrelograms for a group of neurons.

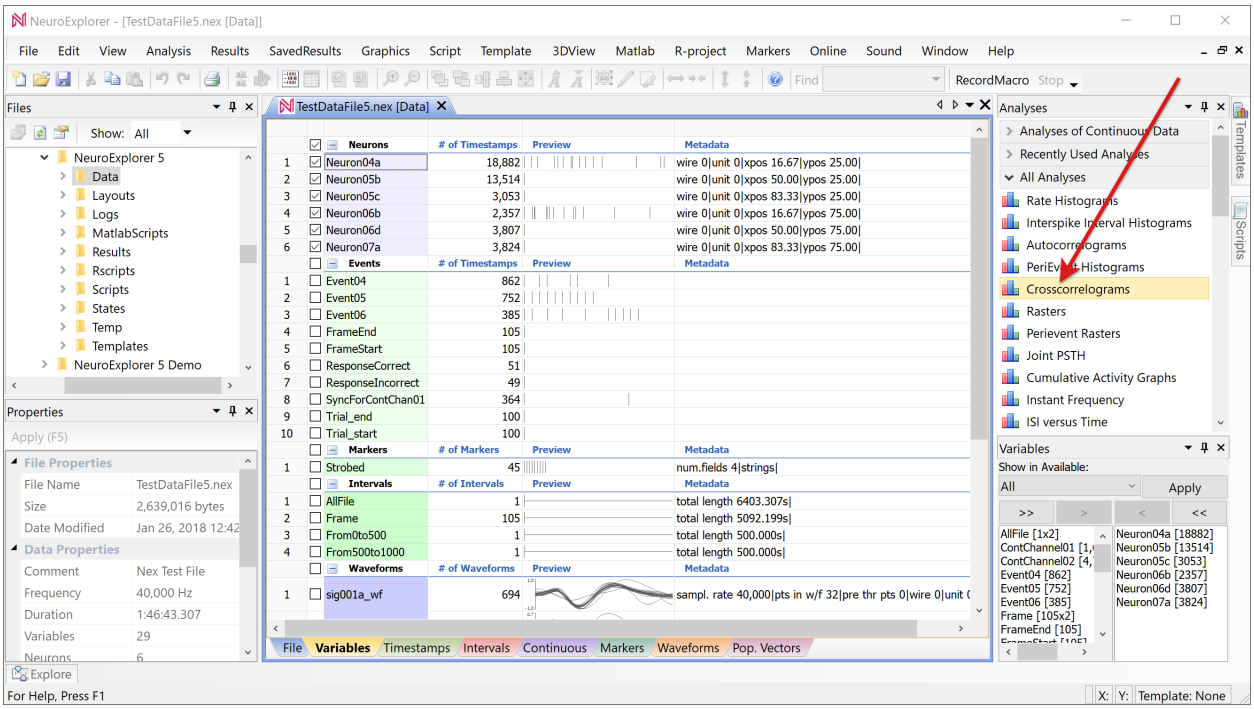
- Select **File | Open** menu command:



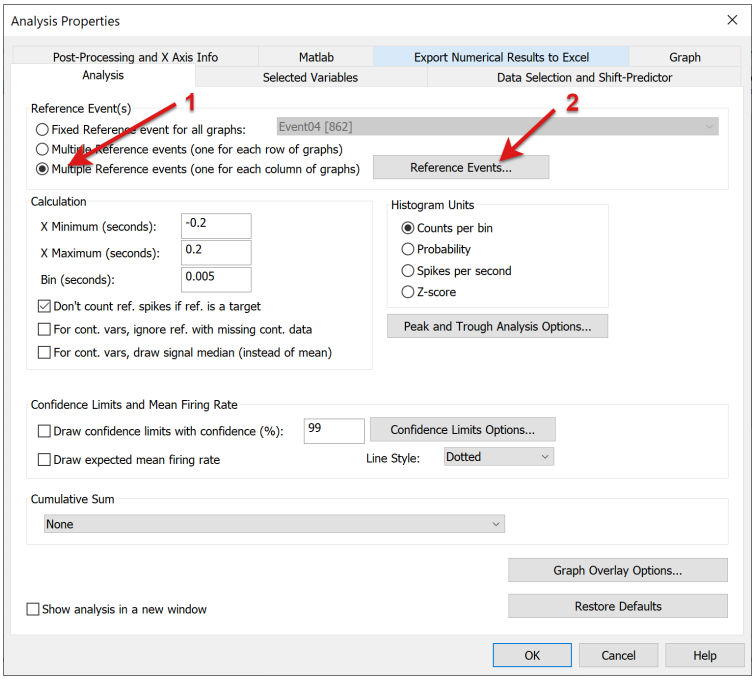
- Select **TestDataFile5.nex** file and click **OK** in the Open dialog. NeuroExplorer will load the test data file:



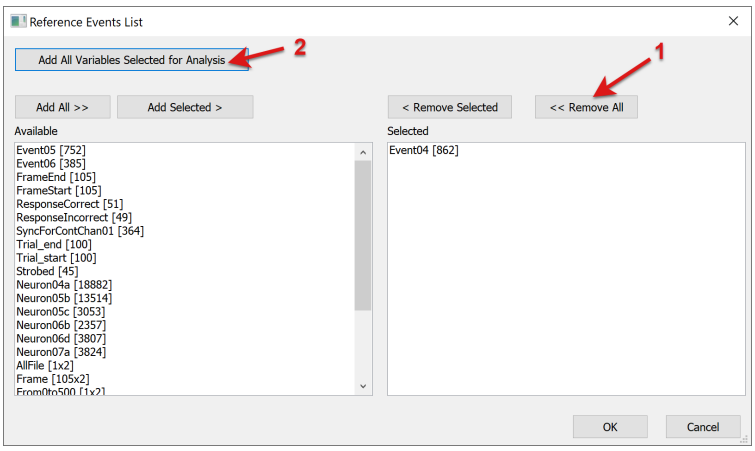
- Select analysis type by clicking the **Crosscorrelograms** line in the Analyses panel:



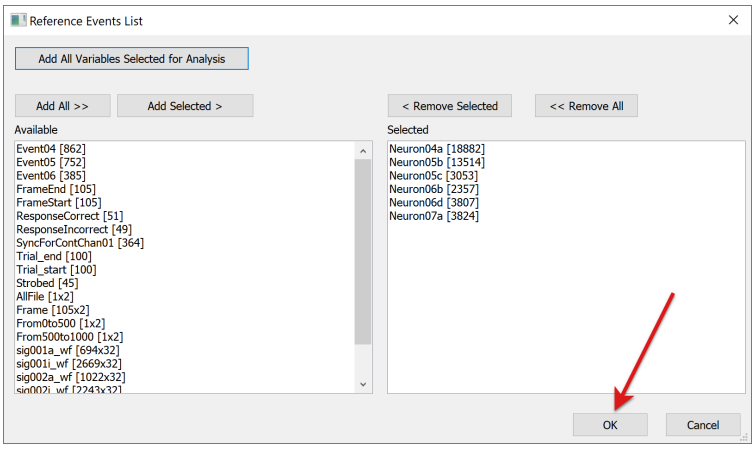
- Click **Multiple Reference Events (one for each column of graphs)** and then click the **Reference Events** button:



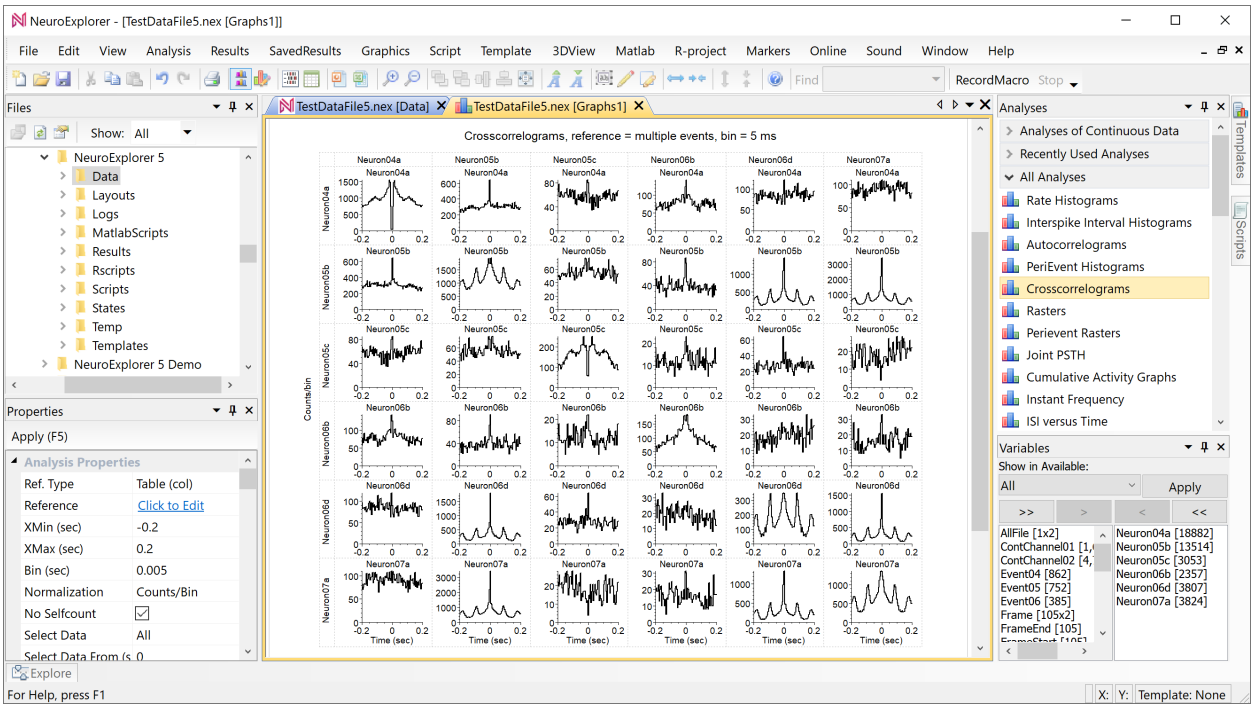
- In the Reference Events List dialog, click **Remove All** and then click **Add All Variables Selected For Analysis** button:



- Now we have all the neurons selected as reference events. Click **OK** button in the Reference Events List dialog:



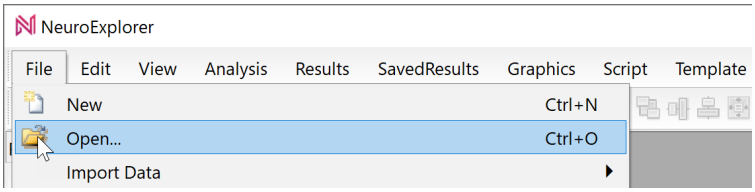
- Click **OK** in the Analysis Properties dialog. NeuroExplorer opens a new window with graphical analysis results.



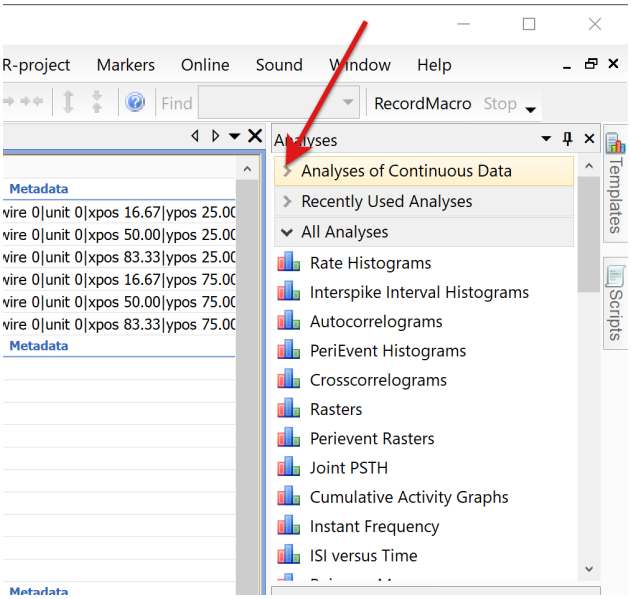
We calculated all 36 pairwise crosscorrelograms for 6 neurons.

1.4 First Continuous Variables Analysis

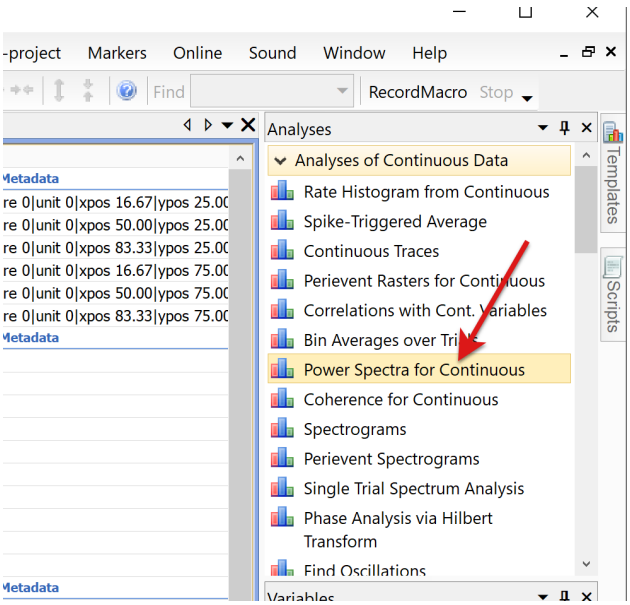
- Select **File | Open** menu command:



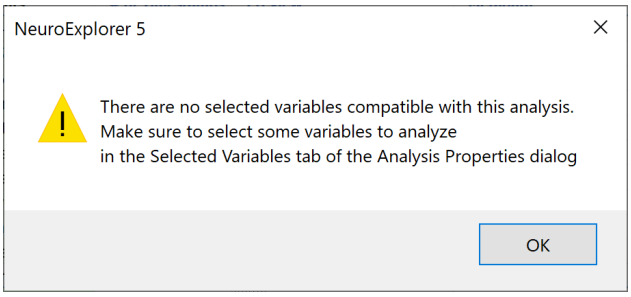
- Select **TestDataFile5.nex** file and click **OK** in the Open dialog. NeuroExplorer will load the test data file.
- Open **Analyses of Continuous Data** folder by clicking the **>** icon at the top of the Analyses panel:



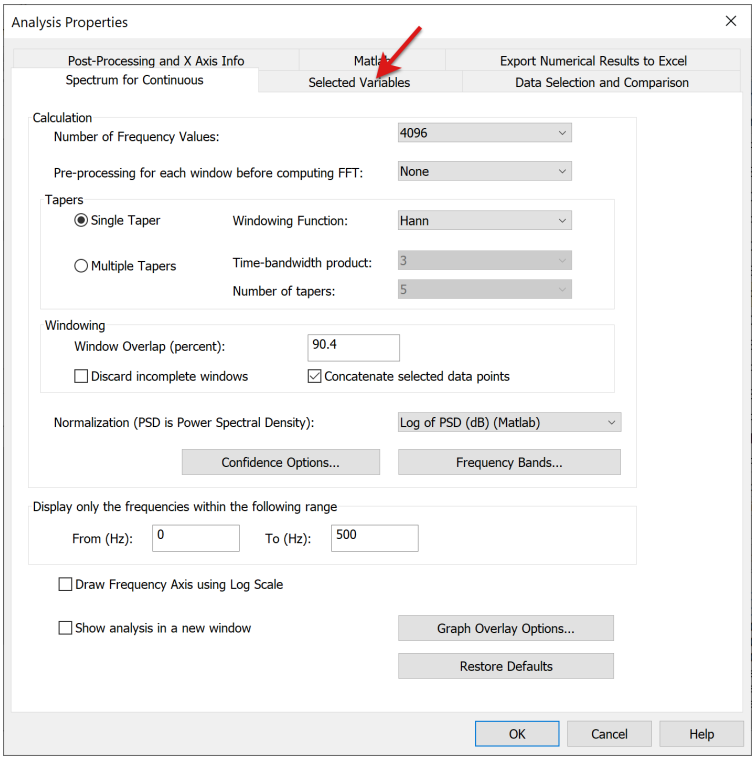
- Select analysis type by clicking the **Power Spectra for Continuous** line in the Analyses panel:



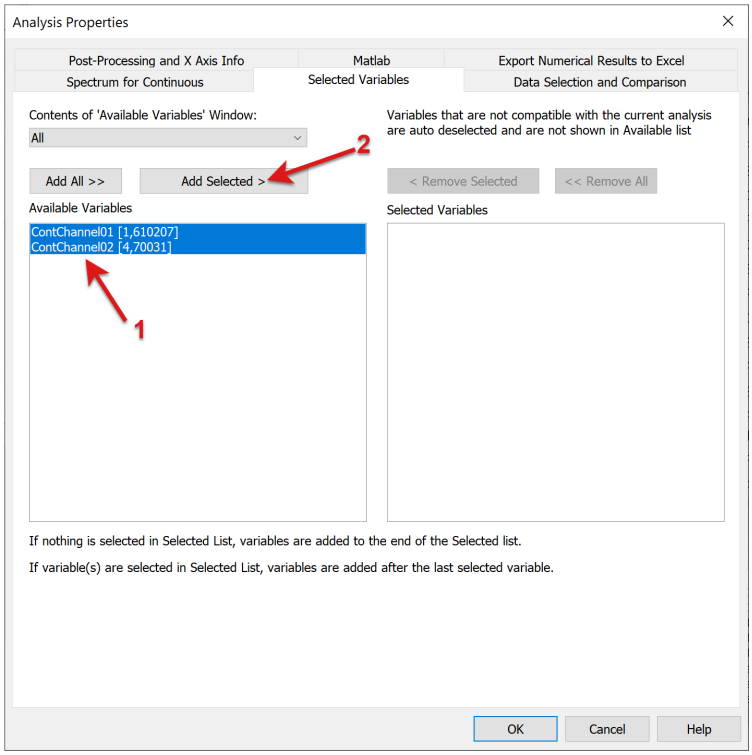
- NeuroExplorer will show the following message box:



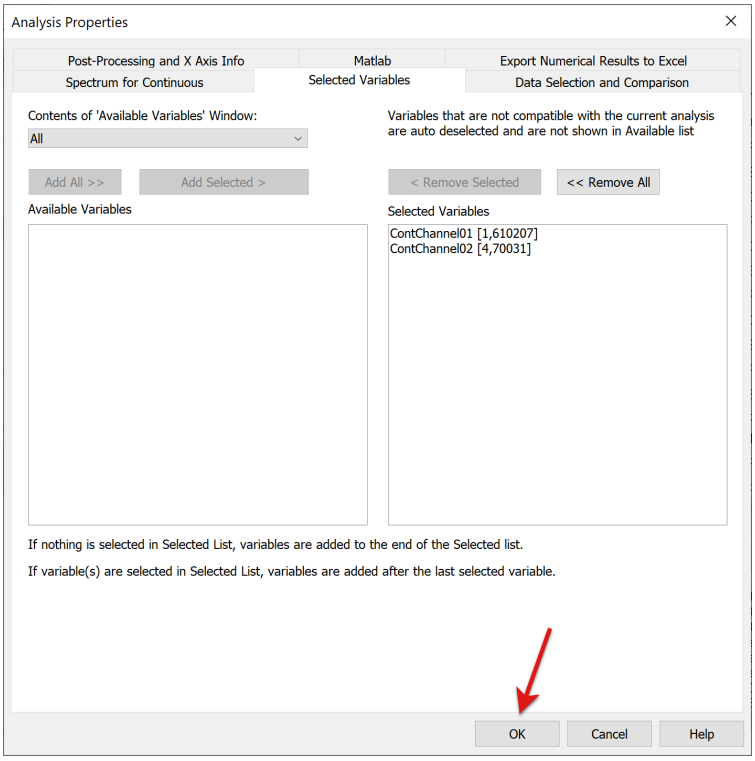
- Close the message box. NeuroExplorer will open Analysis Properties dialog. Click on **Selected Variables** tab:



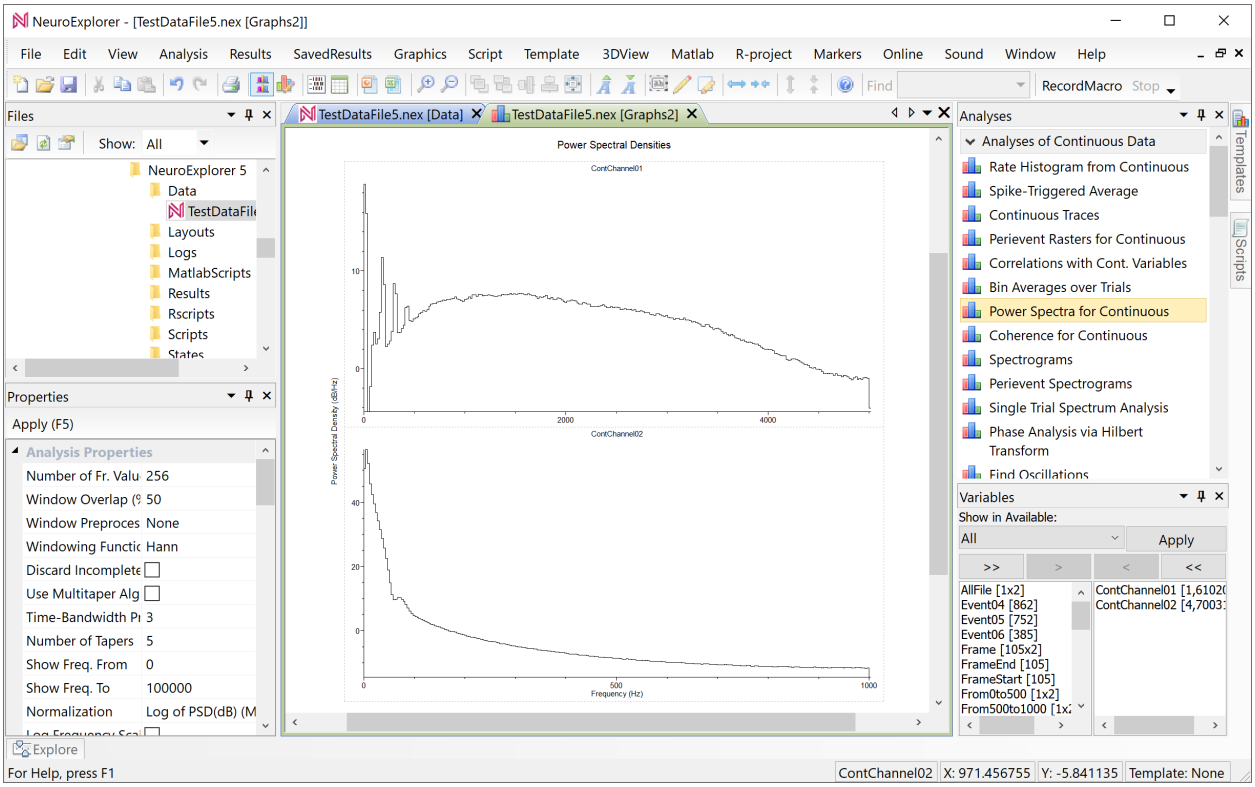
- Select the continuous variables that you want to be analyzed in the left variables list (use Ctrl+Click to select multiple variables) and press **Add Selected >** button:



- Press OK in the Analysis Properties dialog



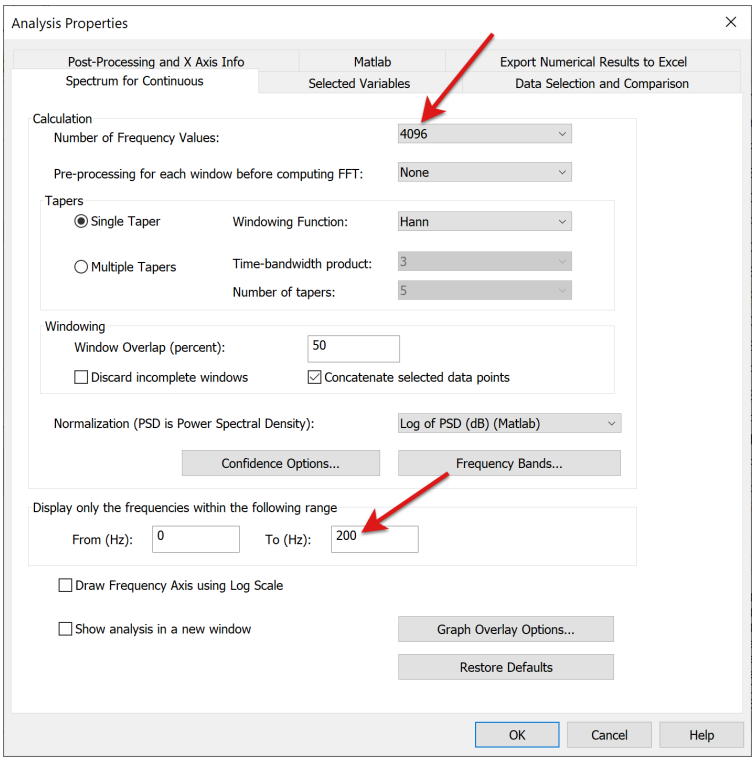
- NeuroExplorer will calculate the spectra of selected continuous variables and will open a new window with graphical analysis results:



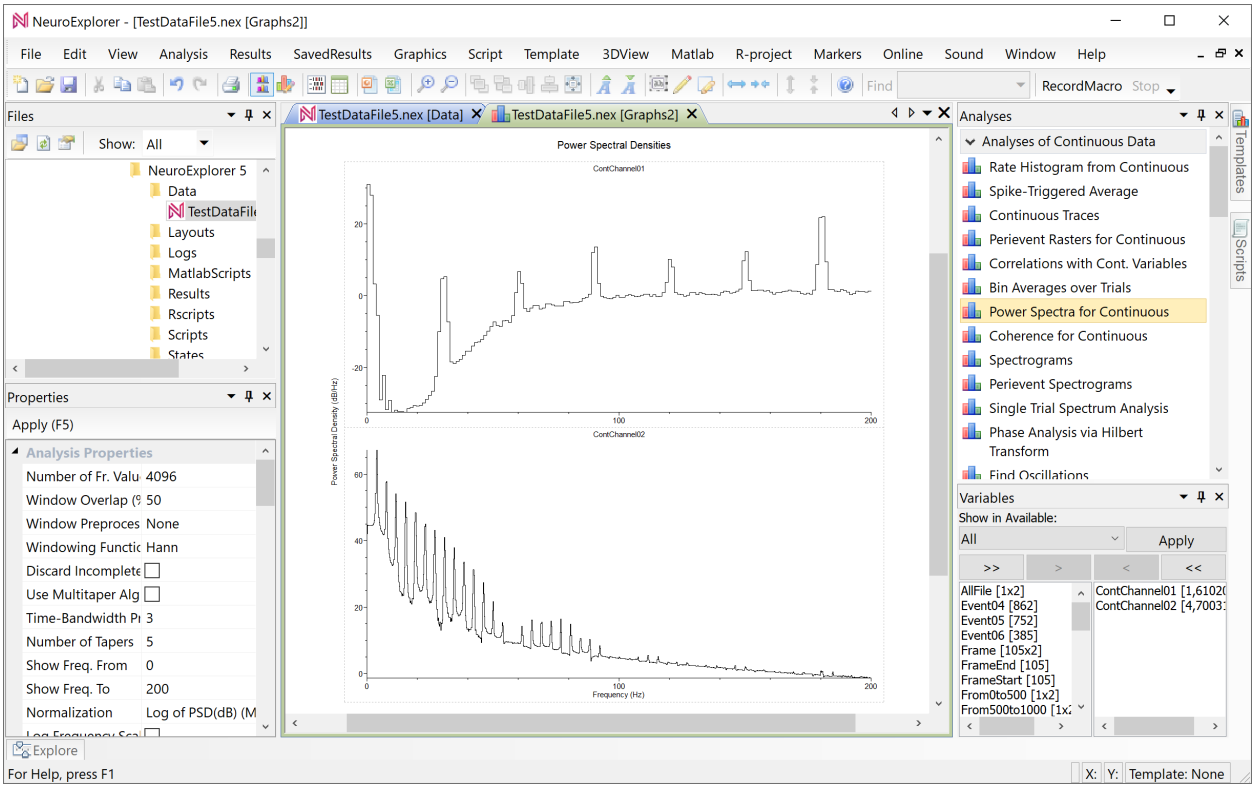
- To zoom in the low frequency values of the spectra, double-click inside any of the

graphs in the new window. NeuroExplorer will open Analysis Properties dialog.

- Change two analysis parameters in the dialog: set **Number of Frequency Values** to 4096, change **Display Frequency To** to 200 and press OK:



- NeuroExplorer will show results of the analysis with the new parameters:



1.5 Save and Restore Analysis Parameters Using Templates

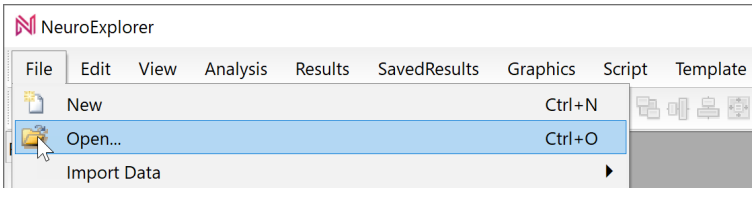
In any analysis in NeuroExplorer, you can adjust a large number of parameters:

- Analysis type (Rate Histograms, IIH, etc)
- Analysis parameters (Bin, XMin, XMax, etc.)
- Graphics parameters (graph type, graph color, X and Y axes, etc)

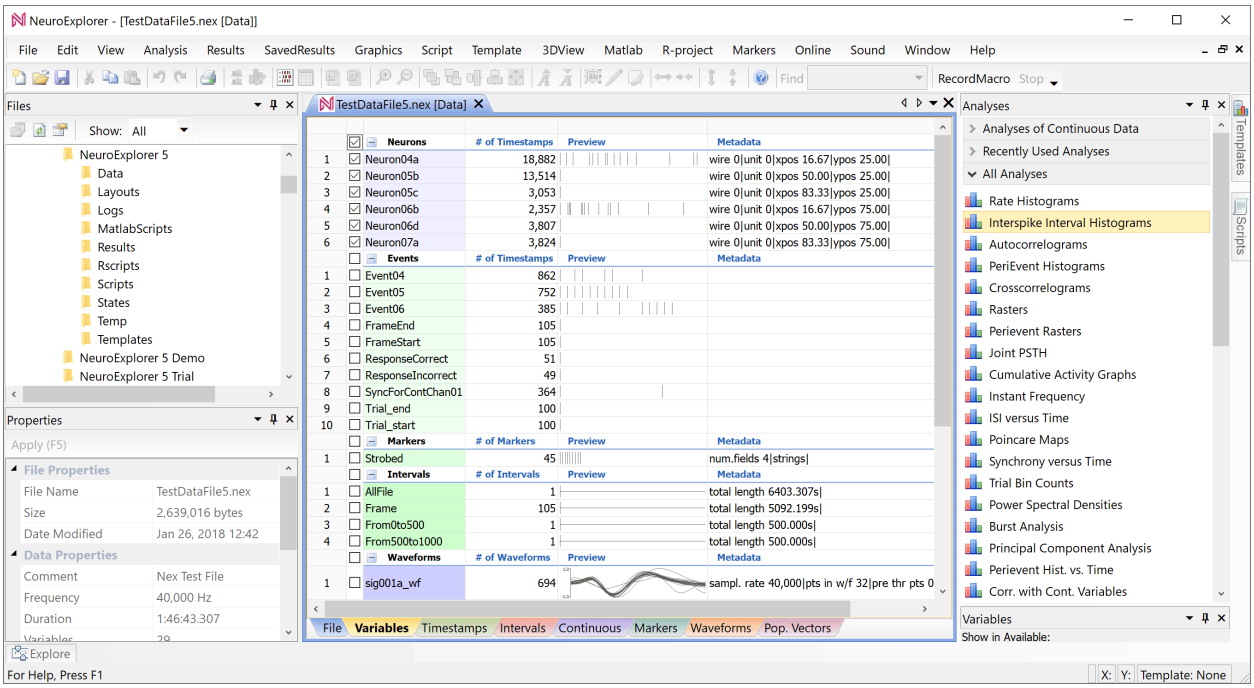
NeuroExplorer allows you to save all these parameters so that when you open another data file, you can easily reproduce exactly the same analysis with the same axes, labels and so on.

The set of all the analysis parameters is called **the Analysis Template**.

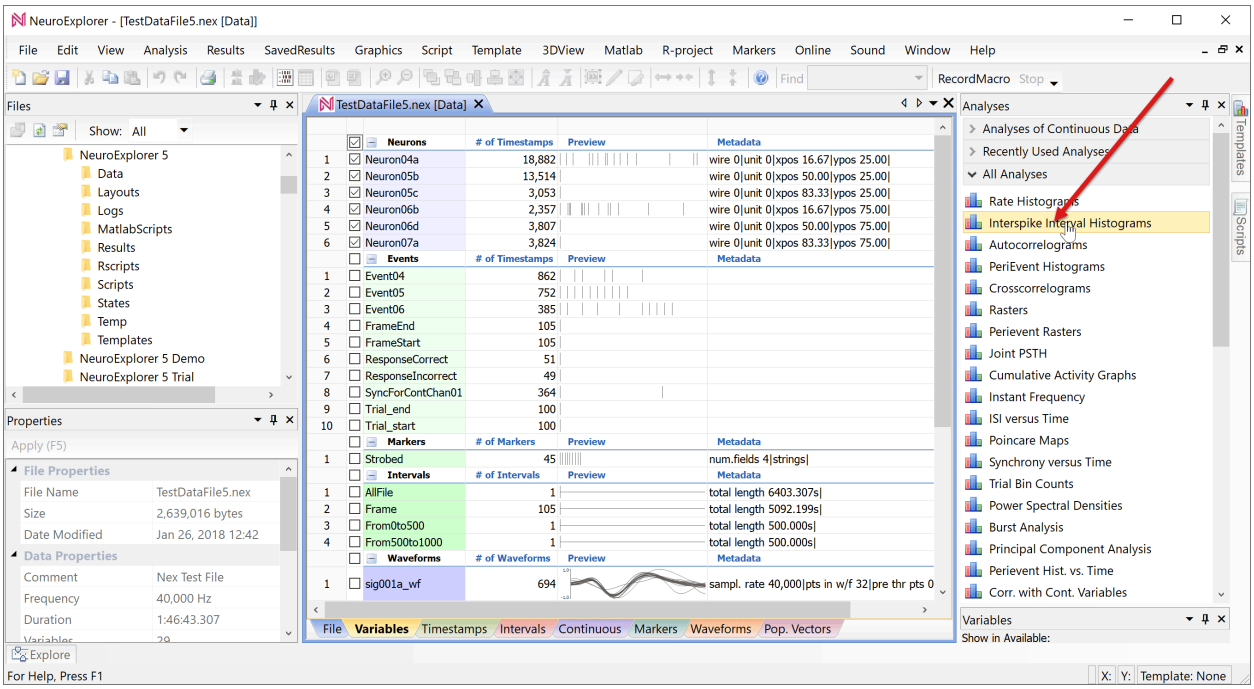
- Select **File | Open** menu command:



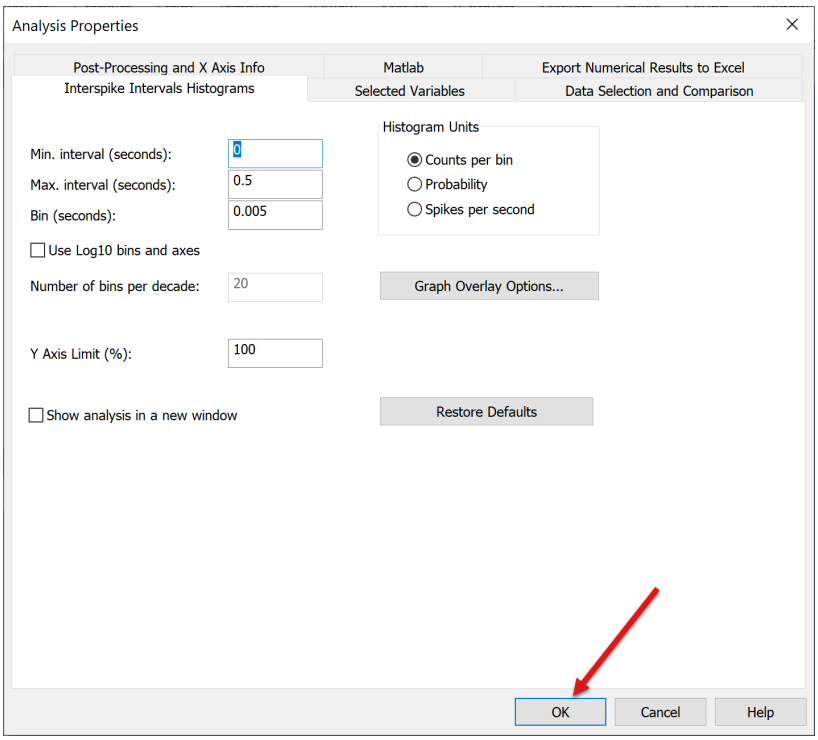
- Select **TestDataFile5.nex** file and click **OK** in the Open dialog. NeuroExplorer will load the test data file:



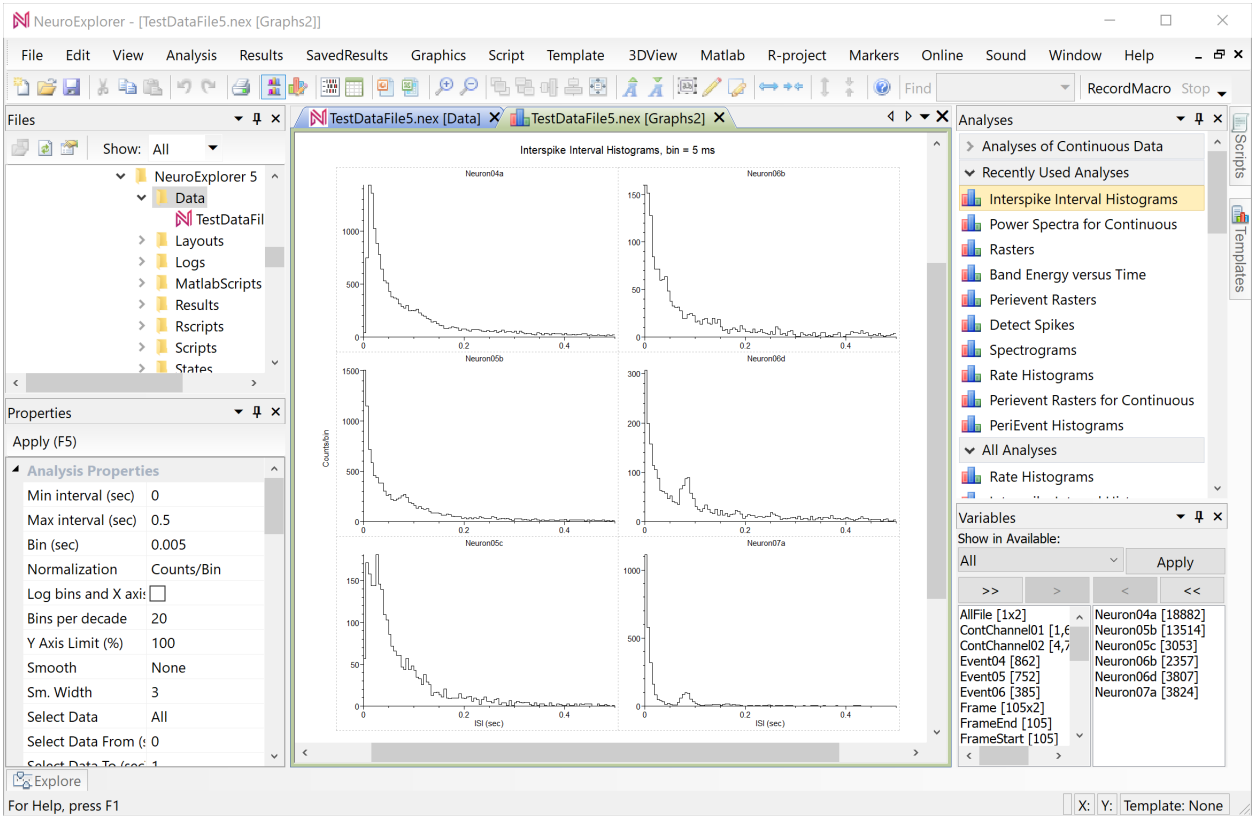
- Select analysis type by clicking the **Interspike Interval Histograms** line in the Analyses panel:



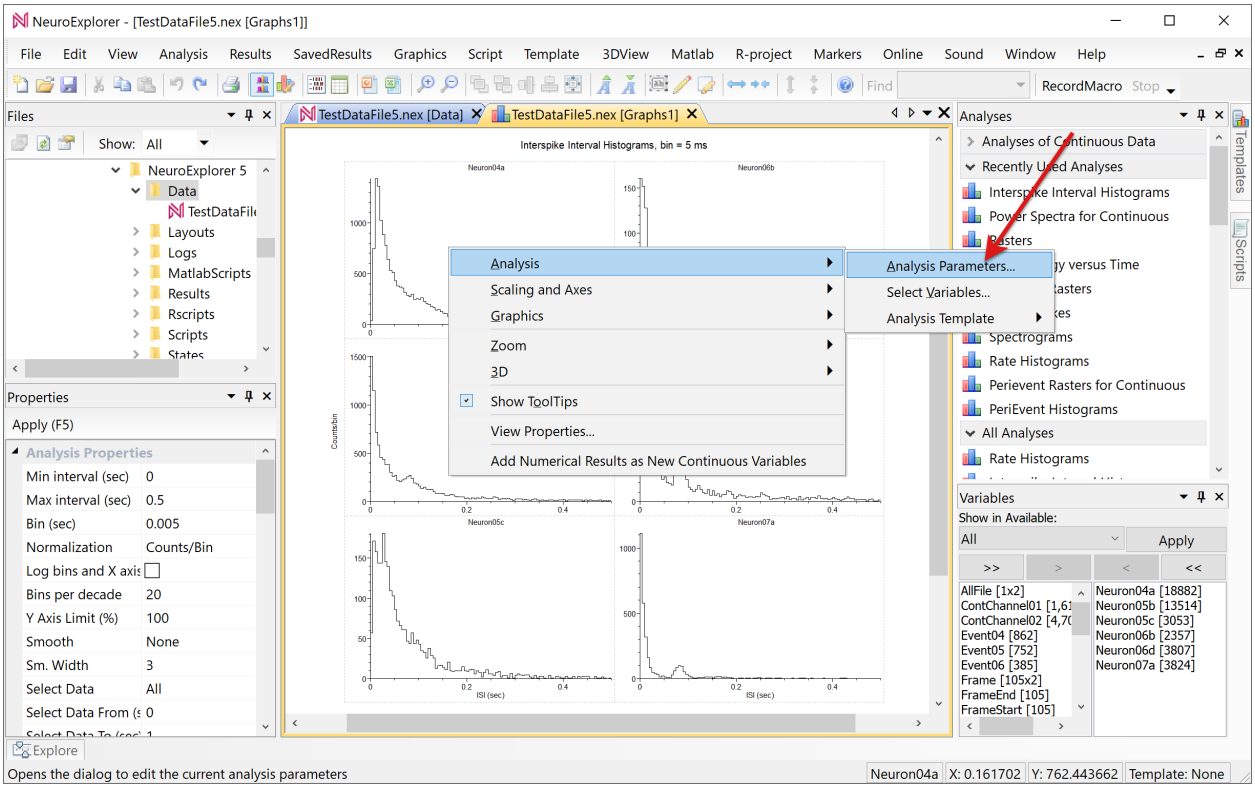
- NeuroExplorer will open **Analysis Properties** dialog. Click **OK** in the dialog:



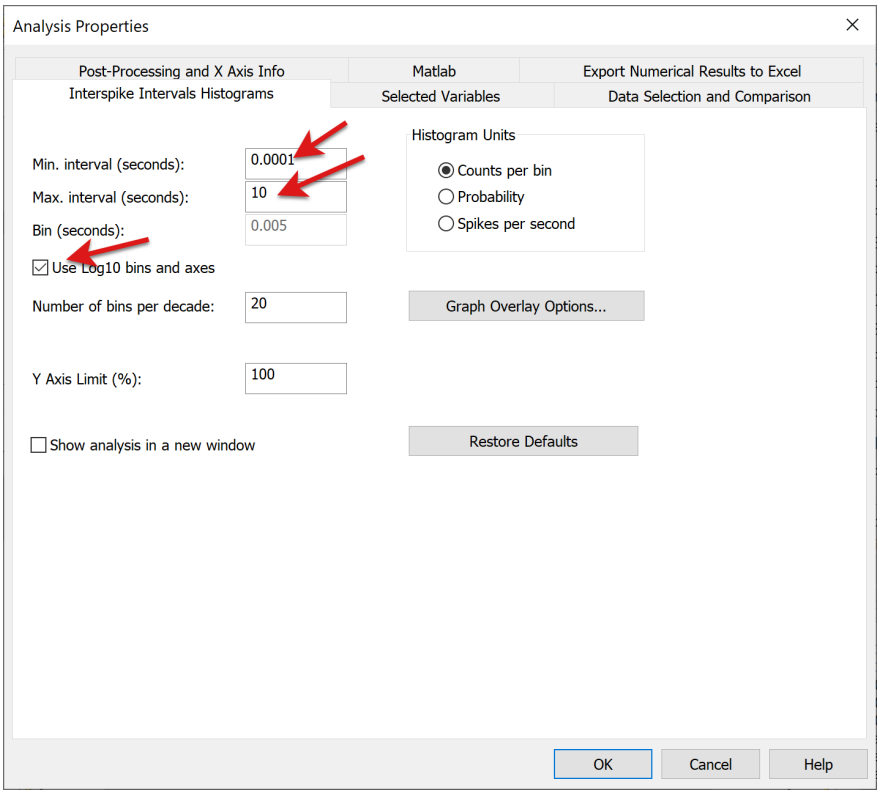
- NeuroExplorer will open a new window with graphical analysis results:



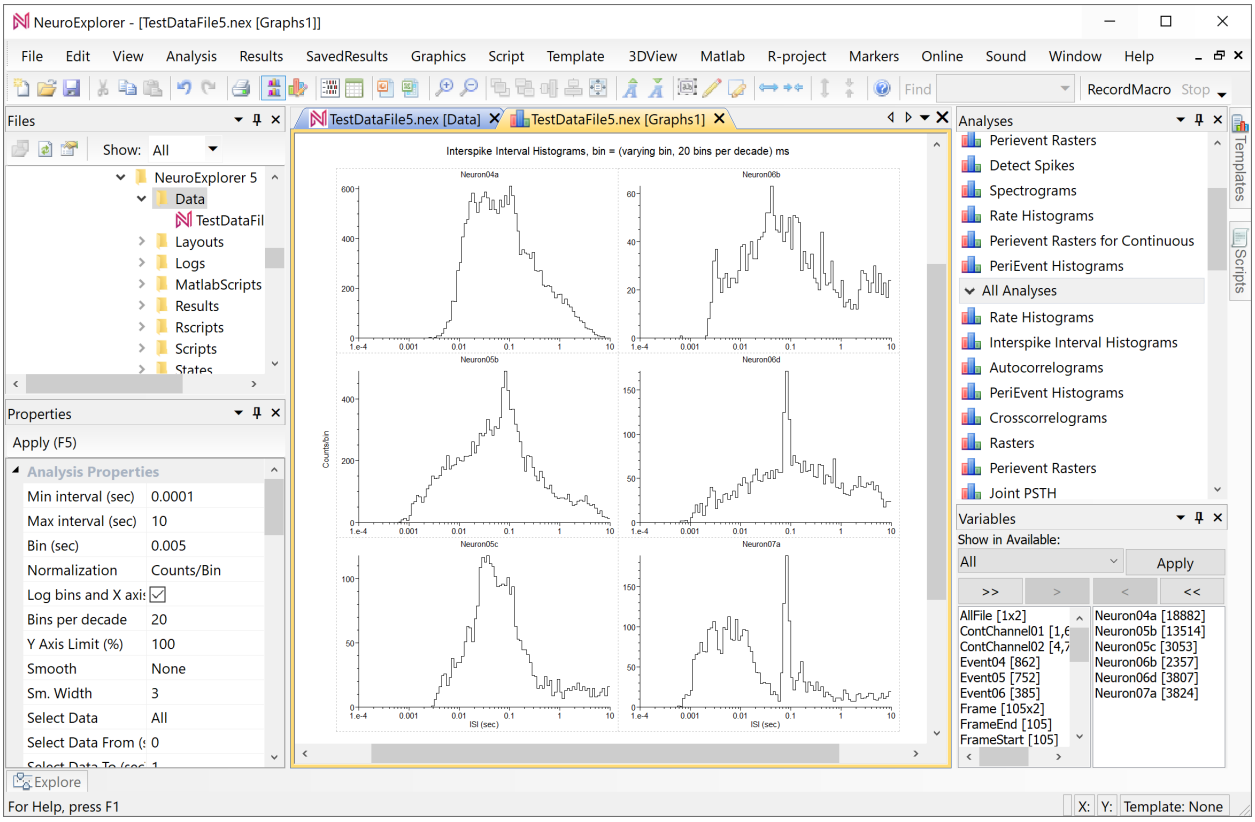
- Right-click in the graphical analysis results window and select **Analysis | Analysis Parameters** menu command:



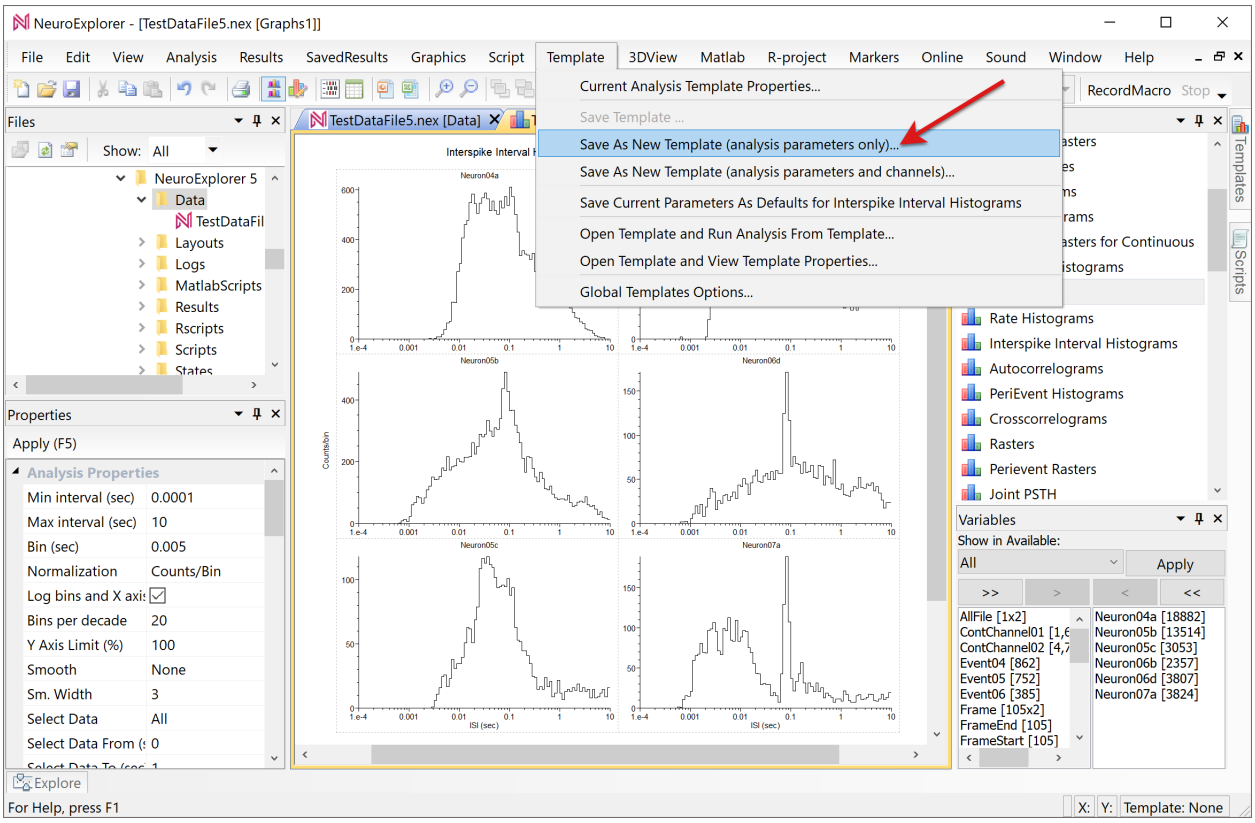
- In the Analysis Properties dialog, change three analysis parameters: set **Min. Interval** to 0.0001, set **Max. Interval** to 10, check **Use Log10 bins and axes** and press OK:



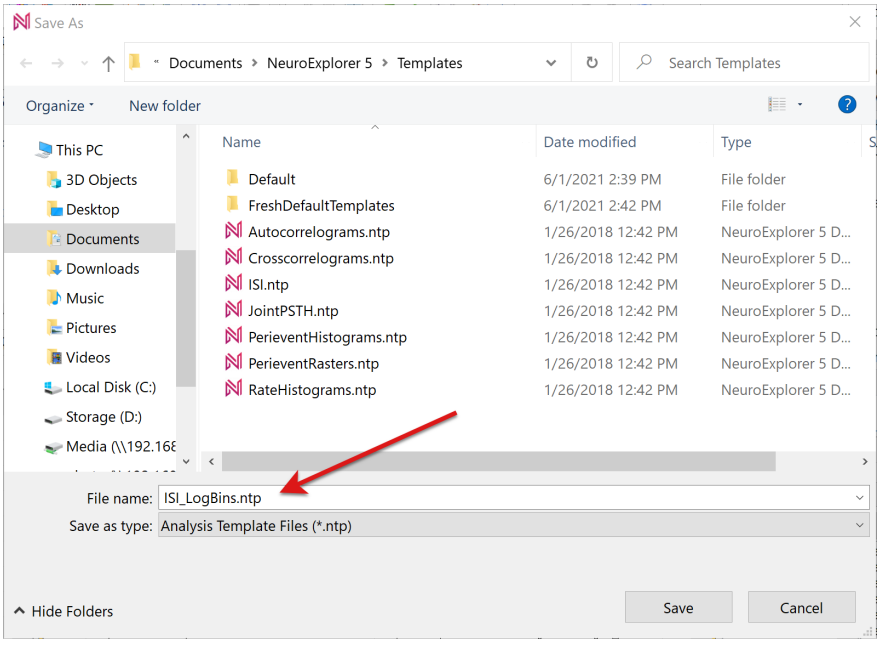
- NeuroExplorer will recalculate Interspike Interval Histograms with logarithmic bins:



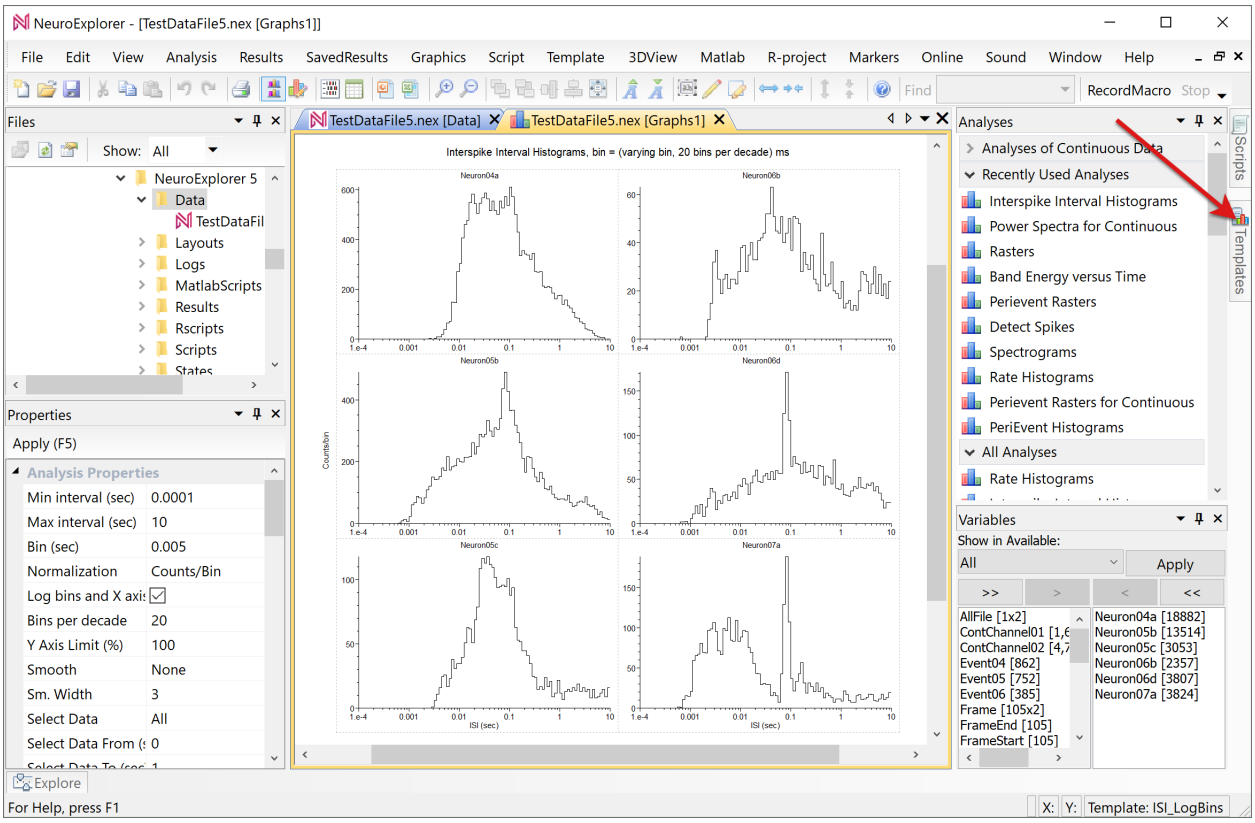
- Select **Template | Save as New Template (analysis parameters only)** menu command



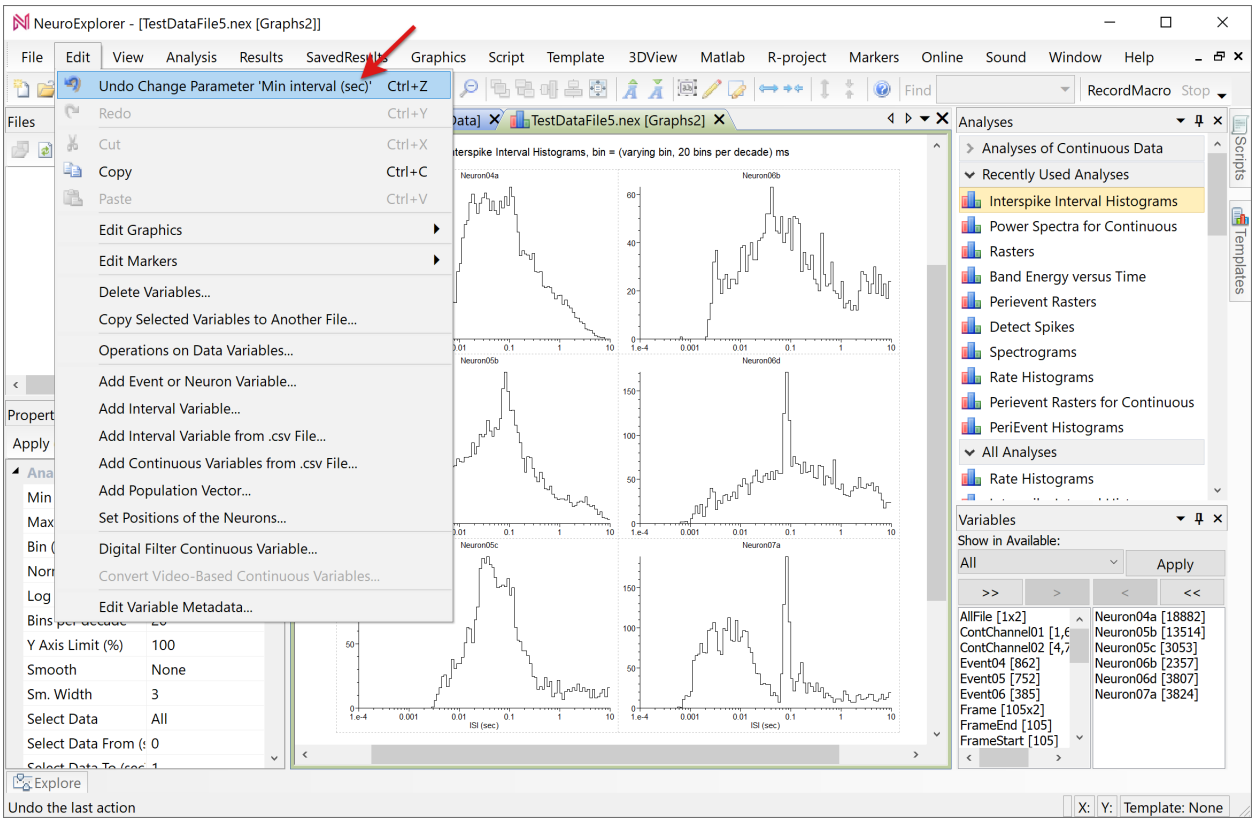
- Specify ISI_LogBins as the new template name



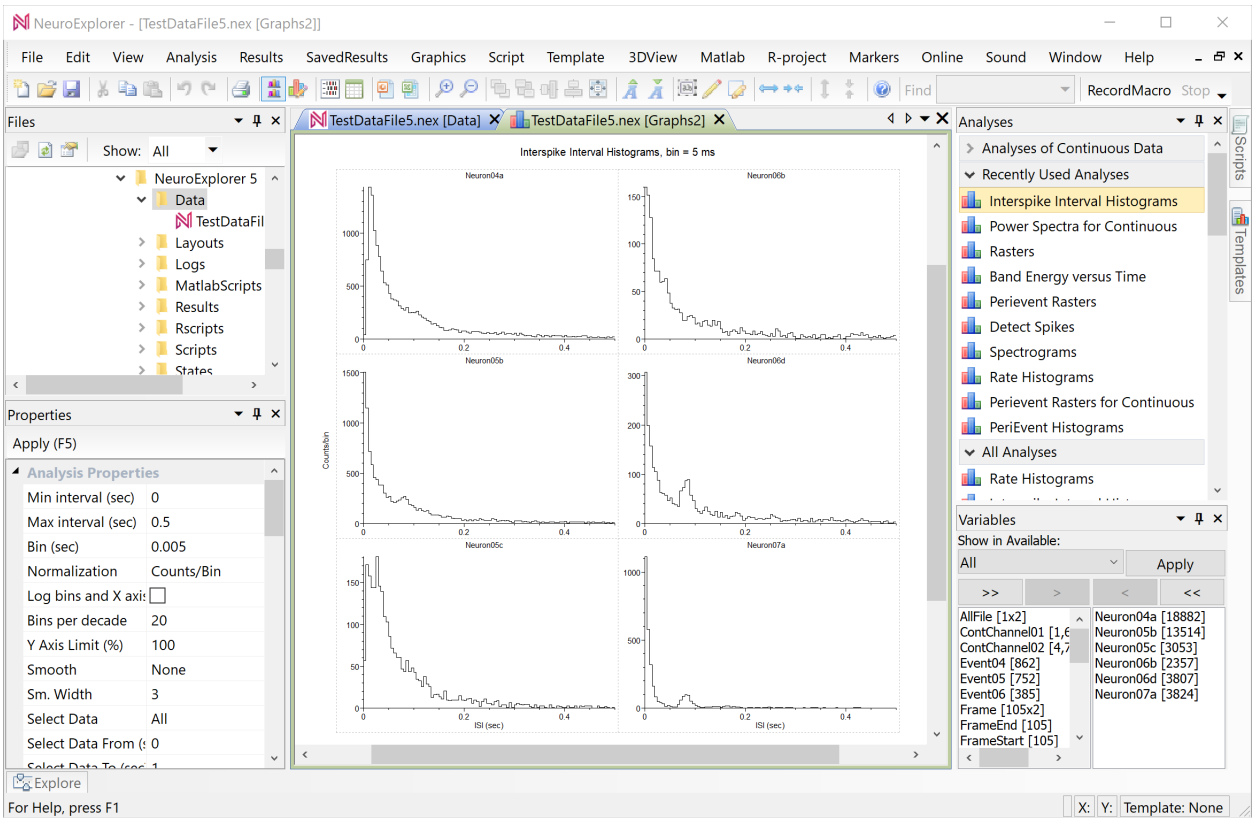
- After you saved the template, the name of the template appears in the Templates Window. To view Templates window, click on the Templates tab in the right toolbar:



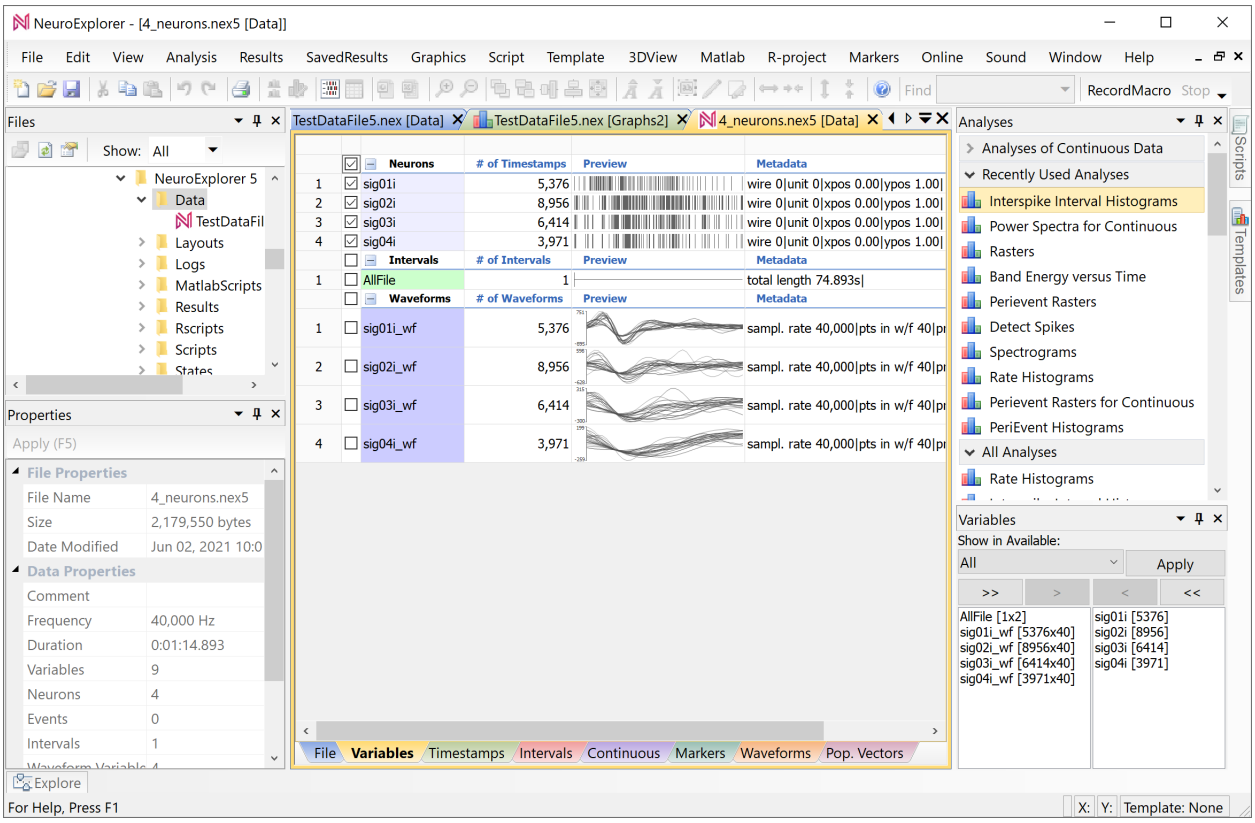
- You can now change parameters in the current graphical analysis results window. For example, you can select Edit | Undo menu command:



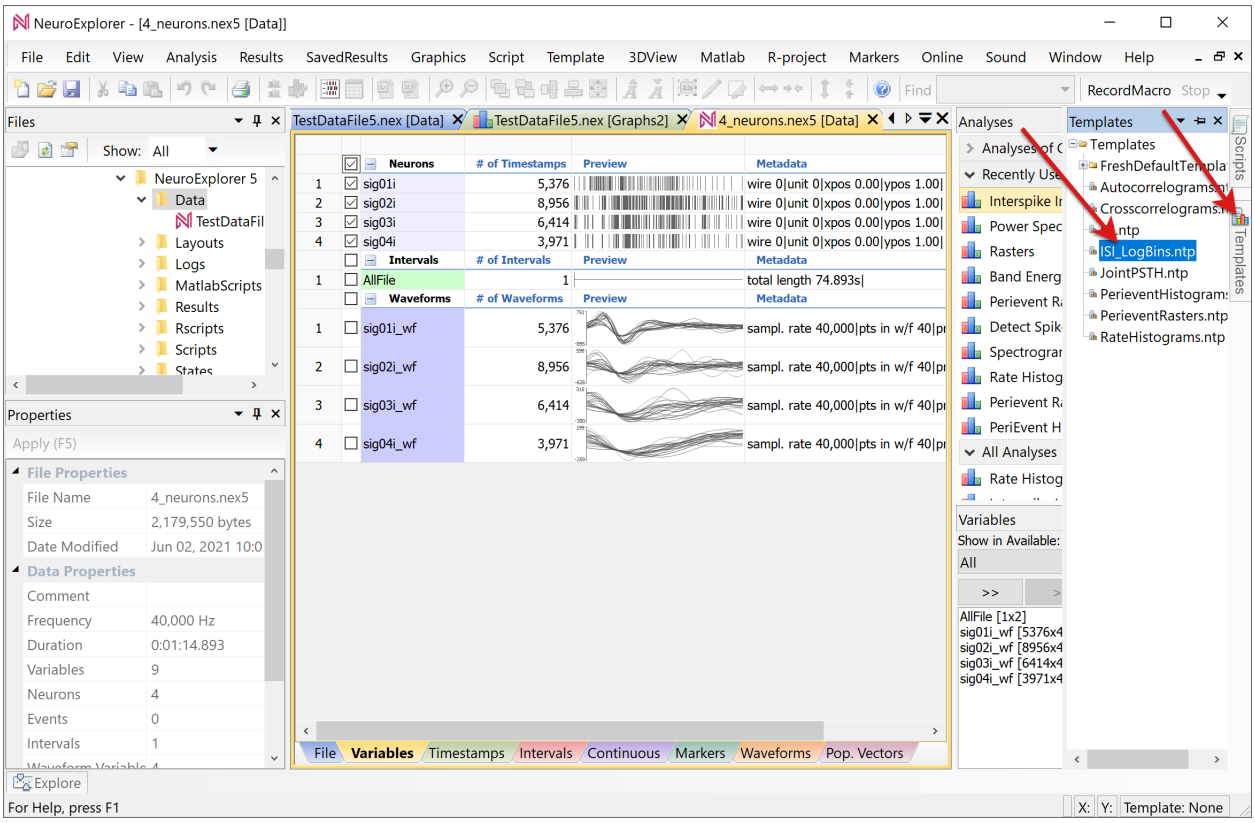
NeuroExplorer will display analysis results with the previous values of the analysis parameters:



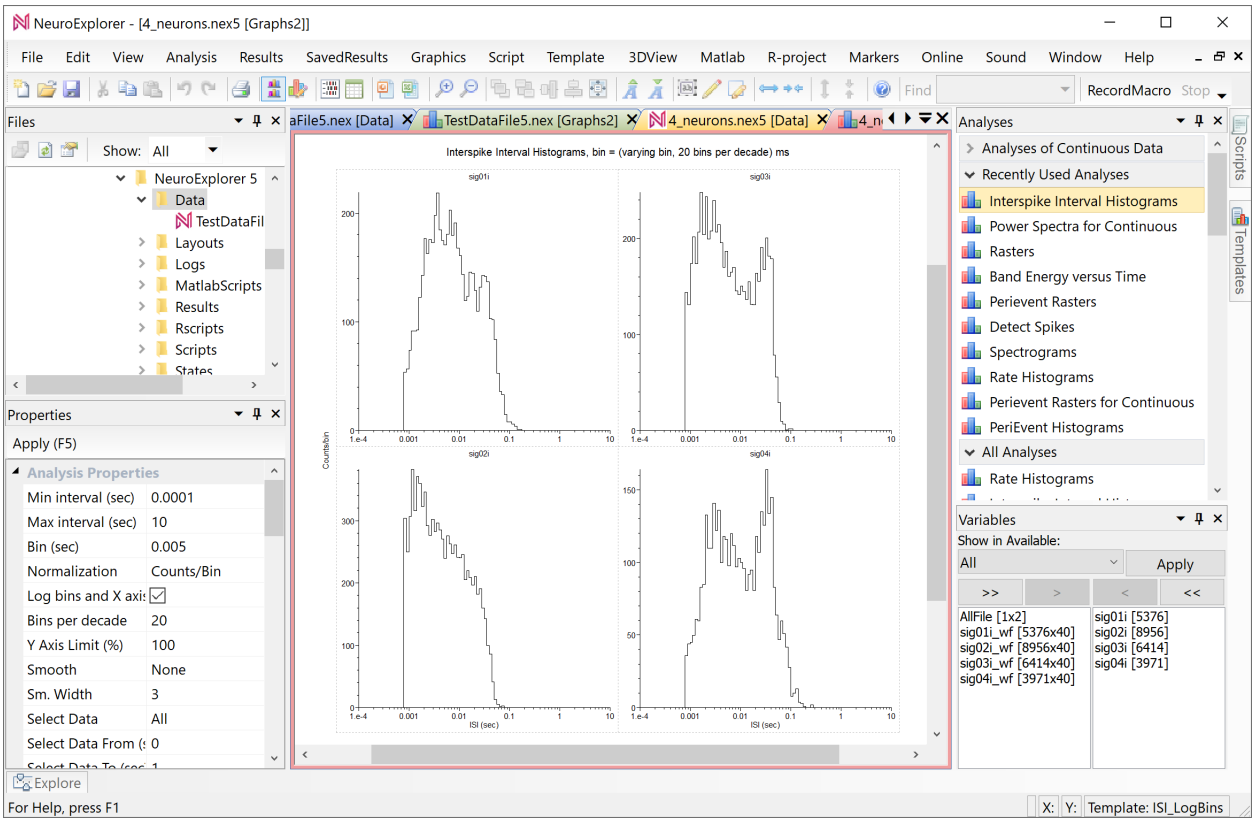
- We can now open a different data file and apply the saved analysis template. Fro example, let's open *4_neurons.nex5* data file:



- To apply our template to the currently opened file, double-click the template name in the Templates Window:



- NeuroExplorer will calculate the analysis results with all the parameters the were set when we saved the template:



You can specify the templates directory by selecting **Templates | Global Templates Properties…** menu command.

CHAPTER

TWO

HOW-TO GUIDES

2.1 Loading Recorded Data

This guide contains an overview of data import methods in NeuroExplorer.

Use **File | Open…** menu command and then select your data file.

If you are using one of *more than 50 supported file formats*, the program will load your data file.

If NeuroExplorer displays an error message about unknown file extension or invalid file format, follow *these instructions*.

If you can load your data in Matlab, you can either *save Matlab data in a .nex5 file*, or *transfer data from Matlab to NeuroExplorer*.

If your data is in Excel, you can *copy and paste your data from Excel to NeuroExplorer*.

2.2 Dealing with Unknown File Format Error

This guide explains how NeuroExplorer imports data files created by data acquisition systems.

NeuroExplorer supports *more than 50 file formats* and tries to identify the data file type from the file extension.

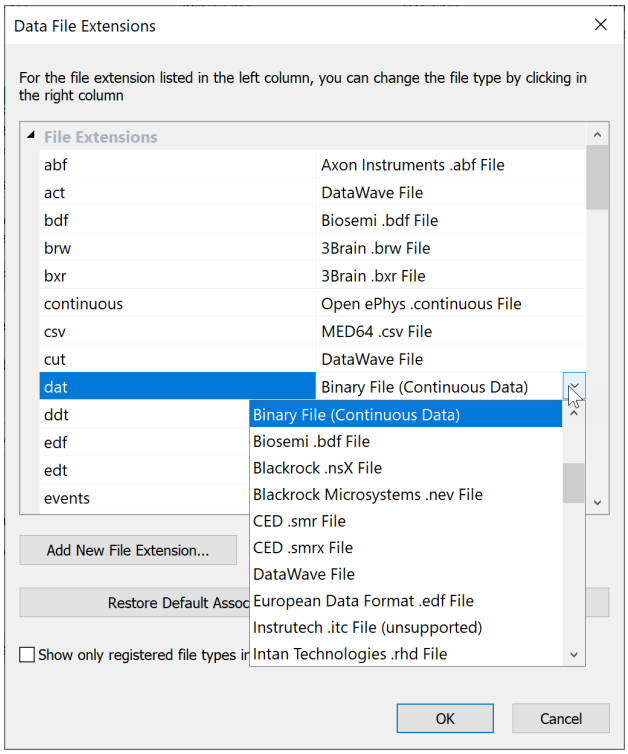
Some file extensions are used by several file formats and NeuroExplorer may be unable to identify what is the file format of the selected file.

For example, a .nev file can be either a Neuralynx data file or Ripple Neuro data file.

Here is what you can do in this situation:

- Use specific **File | Import Data** sub-menu (Plexon Data File, Neuralynx Data File, etc.) to open your data file

- Specify what file format corresponds to a specific file extension
  - Select **View | Data Import Options...** menu command
  - Click the **File Extensions...** button
  - Select a file extension in the left column,
  - Click in the right column and select the file type for the extension:



2.3 Importing Data from Matlab

This guide explains how to transfer data from Matlab to NeuroExplorer.

You can create spike trains (1 x N or N x 1 matrices with timestamps in seconds) or continuous variables in Matlab and transfer them to NeuroExplorer on the fly. Here is how to do this:

- Select **File | New** menu command.
- Select **Matlab | Get Data From Matlab | Open Matlab As Engine** menu command. NeuroExplorer will open Matlab as an engine.

Note: When you run **Open Matlab as Engine** menu command, a new instance of Matlab is opened. This instance will have no variables in its workspace. Therefore, you need to

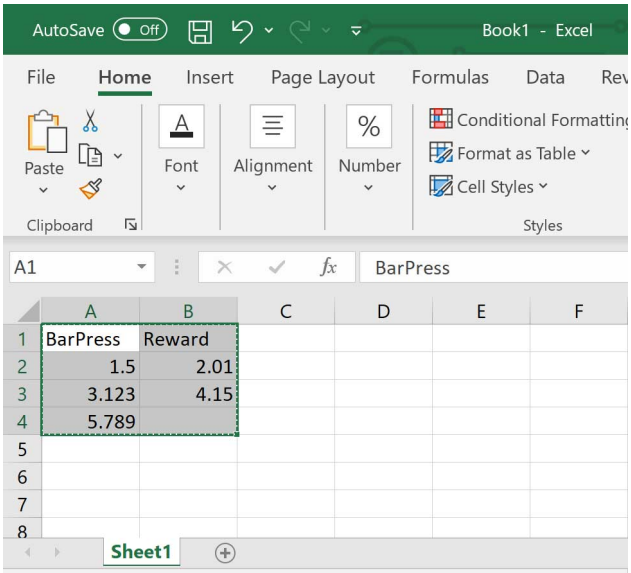
create (or load from a file) the data you want to transfer to NeuroExplorer.

After your data is available in Matlab, select one of the **Matlab | Get Data From Matlab** menu commands to transfer the data to NeuroExplorer.

2.4 Importing Data from Excel

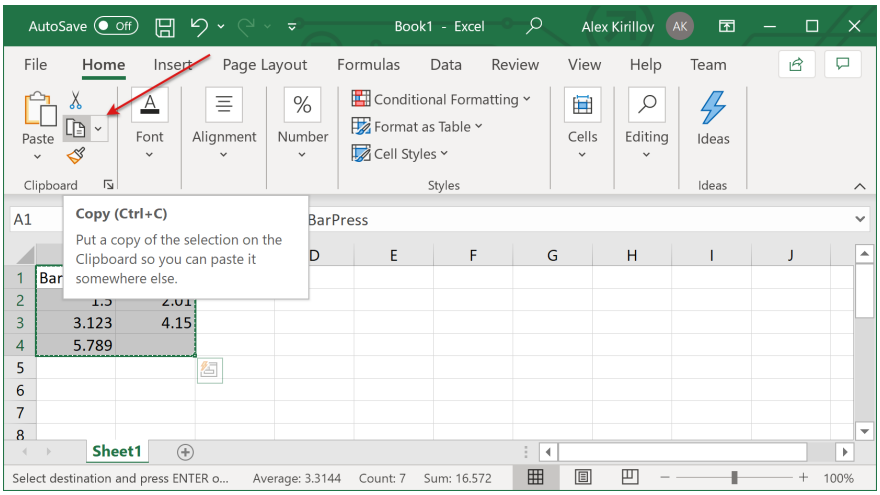
This guide explains how to use Copy/Paste to add data variables to NeuroExplorer.

To paste the following two timestamped variables (`BarPress` and `Reward`) from a spreadsheet to NeuroExplorer:



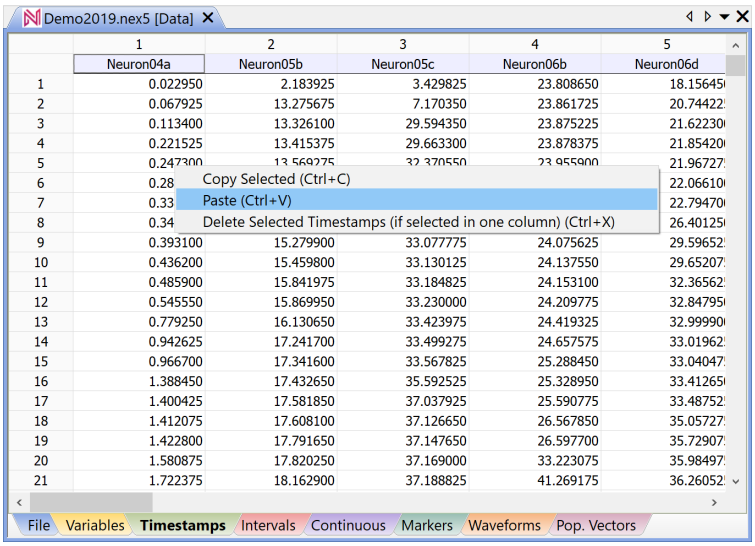
In Excel:

- Select the cell range with both variables (A1 to B4)
- Select **Home | Copy** command



In NeuroExplorer:

- Select the Data view of a file in NeuroExplorer
- Select the **Timestamps** tab of the Data view
- Right-click and select the **Paste** menu command:



NeuroExplorer will create two new Event variables and add them to the file.

Note: Using a similar approach (copy in Excel and paste in NeuroExplorer), you can import continuous variables (right-click in the **Continuous** tab) and time intervals (right-click in the **Intervals** tab).

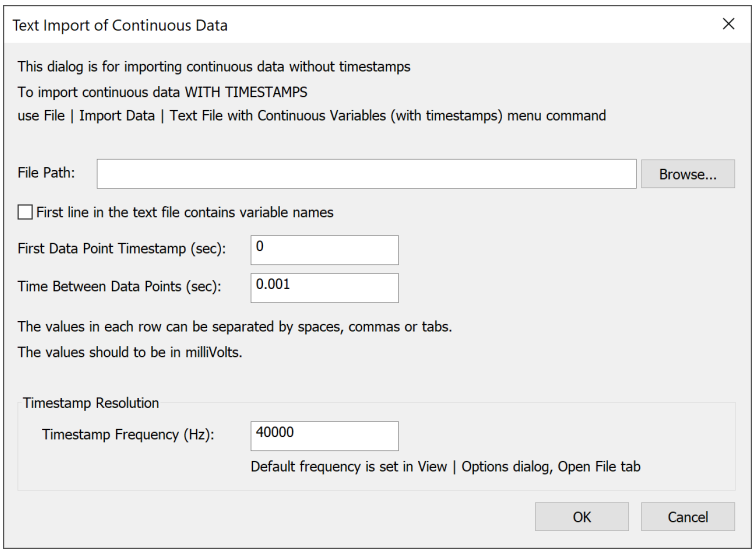
You need to save the file (use **File | Save...** or **File | SaveAs...** menu command) to make this change permanent.

2.5 Importing Continuous Data from Text Files

NeuroExplorer can import continuous data from a multicolumn text file where each column corresponds to a continuous variable. Here is an example:

```
ContChannel1    ContChannel2
114.74609       -63.47656
56.15234        -358.88672
-187.98828      -63.47656
-48.82813       388.18359
-26.85547       285.64453
```

Use **File | Import Data | Text File with Continuous Variables (no timestamps)...** menu command to import such a data file. The following dialog will be shown:



- **First line contains variable names** - if this option is selected, the fields of the first line of the text file are used as variable names. The second line in the file is the first row of data.
- **First Data Point Timestamp** specifies the timestamp of the first data point in each continuous variable.
- **Time Between Data Points** specifies the time step used in calculation of the timestamp for each row of data. For data row **n** (**n** = 1, 2, ...) in the text file, the data row timestamp is calculated using the following formula:

```
DataRowTimestamp = FirstDataPoint + (n-1)*TimeBetweenDataPoints
```

If you have a text file with the timestamps in the first column of the file:

```
Time      ContChannel01
0.000400  114.746094
0.000500  56.152344
0.000600  -187.988281
0.000700  -48.828125
0.000800  -26.855469
```

use **File | Import Data | Text File with Continuous Variables (with timestamps)...** menu command to import such a data file.

2.6 Importing Timestamps from Text Files

NeuroExplorer can import timestamps stored in the following text formats:

1. Multicolumn table of timestamps

```
Neuron01    Neuron02
0.01        0.001
0.3         0.05
0.5         0.1
            0.4
            0.6
```

In this format, each column in the text file contains the timestamps of a neuron. Each column represents a different neuron. The first line of the file contains the names of all the variables.

2. Pairs <name> <timestamp> The text file in this format should contain pairs of the type

```
name timestamp
```

where

`name` is a character string. The first character of the name should be a letter. If the number is used as the name, NeuroExplorer will add "event" at the beginning of the variable name.

`timestamp` is a number representing the neuron firing time (or event time) in seconds or in time ticks.

Here is an example of a text file with the timestamps in seconds:

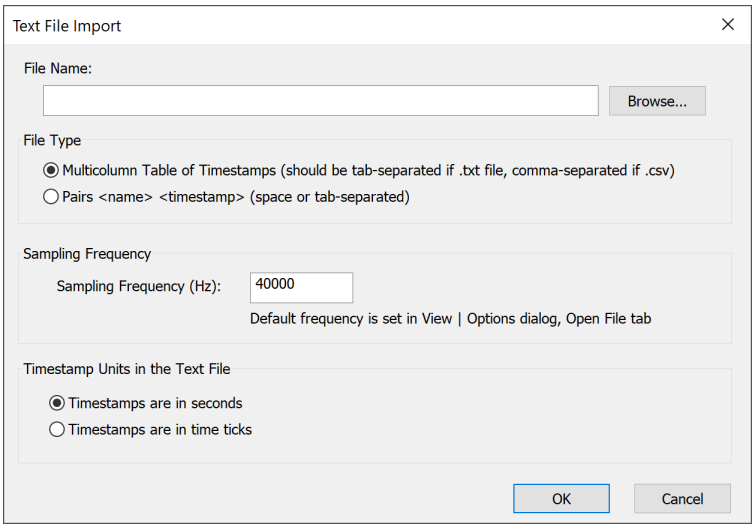
```
Neuron01    0.01
Neuron01    0.3
Neuron02    0.001
Neuron02    0.05
```

(continues on next page)

(continued from previous page)

```
Neuron01    0.5
Neuron02    0.1
Neuron02    0.4
```

Use **File | Import Data | Text File with Timestamps...** menu command to import a text file with timestamps. The following dialog will be shown:



The following options can be specified in Text Import dialog:

- **Sampling Frequency** - this parameter defines the internal representation of the timestamps that NeuroExplorer will use for this file. Internally, the timestamps are stored as integers representing the number of time ticks from the start of the experiment. The time tick is equal to

```
1./Sampling_Frequency
```

- **Timestamp Units** - this parameter defines how the numbers representing the timestamps are treated by NeuroExplorer. If the timestamps are in time ticks, they are stored internally exactly as they are in the text file. If the timestamps are in seconds, NeuroExplorer converts them to time ticks:

```
Internal_Timestamp_In_Ticks = Timestamp_In_Seconds * Sampling_Frequency
```

2.7 How to read and write .nex and .nex5 files in Matlab, Python, C++ and C#

.nex and .nex5 files are binary files with the following general structure:

- File header (number of variables in file (N), etc.)
- Variable1 header (variable name, number of values, etc.)
- Variable2 header (variable name, number of values, etc.)
- …
- VariableN header (variable name, number of values, etc.)
- Variable1 data (timestamps, continuous values, etc.)
- Variable2 data (timestamps, continuous values, etc.)
- …
- VariableN data (timestamps, continuous values, etc.)

The code to read and write .nex and .nex5 files is available in the following zip files:

File Name	Description
HowToReadAndWriteNexAndNex5FilesInMatlab.zip	Matlab code to read and write .nex and .nex5 files
HowToReadAndWriteNexAndNex5FilesInPython.zip	Old Python code to read and write .nex and .nex5 files. See also nex5file Python package documentation
HowToReadAndWriteNexFiles.zip	C++ code to read and write .nex files
HowToReadAndWriteNex5Files.zip	C++ code to read and write .nex5 files
NeuroExplorerNexFileCSharp.zip	C# code to read and write .nex and .nex5 files

README.txt file in each of these zip files contains instructions on how to use the code.

2.8 Dealing with License Key Errors

NeuroExplorer requires either a physical USB license key or a software (cloud) license key to operate.

Sentinel USB license key:

Dinkey USB license key:

Make sure to install the correct version of NeuroExplorer.

If you purchased a USB license key:

- Download and install NeuroExplorer5Setup64.exe.

If you purchased a cloud (software) license key:

- Download and install NeuroExplorerNexCloudSetup.exe.

Sentinel drivers need to be installed so that NeuroExplorer can communicate with Sentinel license keys. These drivers are installed when you run NeuroExplorer setup (NeuroExplorer5Setup64.exe).

Dinkey license keys or cloud license keys do not require Windows drivers.

2.8.1 Error Messages and Troubleshooting

Message	Troubleshooting
Unable to find license key	Follow instructions in *Unable to Find Key Message*.
Sentinel key does not have version 5 license	You are using NeuroExplorer version 3 or version 4 Sentinel key. Follow instructions in *Sentinel Key Does Not Have Version 5 License*.
Unable to read Sentinel key data	Contact NeuroExplorer technical support (support@neuroexplorer.com). Sentinel key may be damaged.
Unable to initialize Sentinel library	Follow instructions in *Make Sure Sentinel Drivers are Installed*.

Unable to Find Key Message

1. If you are using Dinkey license keys:

   follow instructions in *Dinkey License Keys*.

2. If you are using Sentinel keys:

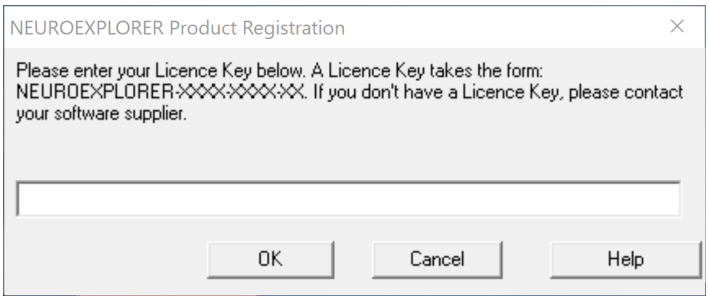
   1. Verify that you are using a NeuroExplorer license key. Make sure that the code `SRB10491` or `JKFZZ` is printed on the key. The newer Sentinel keys also have `Neuroexplore` printed on them:

   2. Follow the instructions in *Make Sure Sentinel Drivers are Installed*.
   3. If the problem persists, you may need to *Use Sentinel Cleanup Utility*.
   4. If you still have problems with the Sentinel key, contact NeuroExplorer technical support (support@neuroexplorer.com).

3. If you purchased a cloud (software) license key:

   1. Download and install NeuroExplorerNexCloudSetup.exe.
   2. In the NeuroExplorer Product Registration dialog, enter the license key code provided to you:

Sentinel Key Does Not Have Version 5 License

The message **Sentinel key does not have version 5 license** indicates that you are using NeuroExplorer version 3 or version 4 Sentinel key.

If you purchased NeuroExplorer version 5, contact NeuroExplorer technical support (support@neuroexplorer.com).

If you would like to upgrade to NeuroExplorer version 5, contact NeuroExplorer support or one of the NeuroExplorer resellers listed in the Purchase Page of the NeuroExplorer web site.

Make Sure Sentinel Drivers are Installed

1. Open *Control Panel* | *Uninstall a program* and verify that Sentinel System Driver Installer 7.6.1 is listed in Currently Installed Programs.
    - If an older version (prior to 7.6.1) of Sentinel Driver installer is listed, remove this version and reboot the computer.
2. To install Sentinel 7.6.1 drivers, download and run Sentinel driver installer Sentinel-SystemDriverInstaller.7.6.1.exe.

Important: Reboot the computer after installing the drivers.

Use Sentinel Cleanup Utility

1. Uninstall NeuroExplorer and all the Sentinel software items listed in *Control Panel* | *Uninstall a program*.
2. Download Sentinel cleanup utility: ssdcleanup.zip.
3. Unzip and run the utility. Make sure you let the utility finish running. It may take several minutes.
4. Reboot the computer.
5. Install the latest version of NeuroExplorer: download and run NeuroExplorer5Setup64.exe.
6. Reboot the computer.

Dinkey License Keys

Beginning February, 2022, NeuroExplorer will use two types of license keys.

1. Sentinel license keys:

2. Dinkey license keys:

Dinkey license keys require NeuroExplorer version 5.400 or later. You can check your version of NeuroExplorer using `Help | About` menu command.

To install the latest version of NeuroExplorer, download and run NeuroExplorer5Setup64.exe and reboot your computer.

Dinkey license keys do not require Windows drivers.

If you have problems with Dinkey license keys, contact NeuroExplorer technical support (support@neuroexplorer.com).

2.9 How to Customize the Analyses Panel

To speed up access to the analyses you use most often, you can rearrange the order of the analyses in the **All Analyses** list. Simply drag the analyses up and down in the list.

There is also a **Recently Used Analyses** list that shows the last 10 analyses that were used.

The layout of the **Analyses Panel** is saved in the file

```
C:\Users\<user_name>\Documents\NeuroExplorer 5\Layouts\AnalysesPanelLayout.
↪layout
```

If you want to standardize the order of analyses on all computers in your lab, you can rearrange the analyses on one of the computers. Then, exit NeuroExplorer and copy the `AnalysesPanelLayout.layout` file to

```
C:\Users\<user_name>\Documents\NeuroExplorer 5\Layouts
```

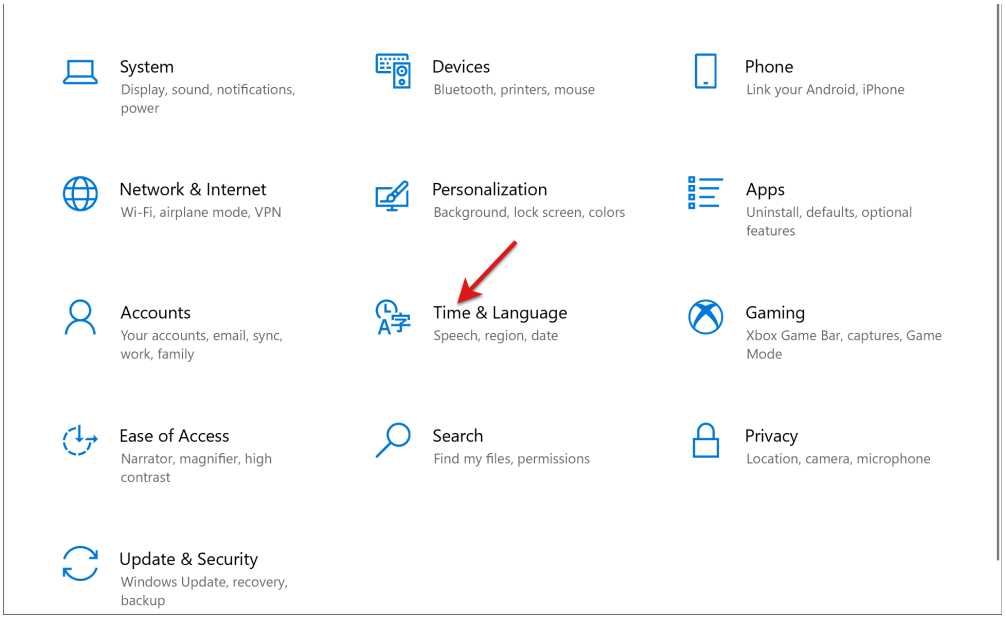
folders on other computers.

2.10 How to fix the program freeze problem when using keyboard with Chinese and Japanese languages

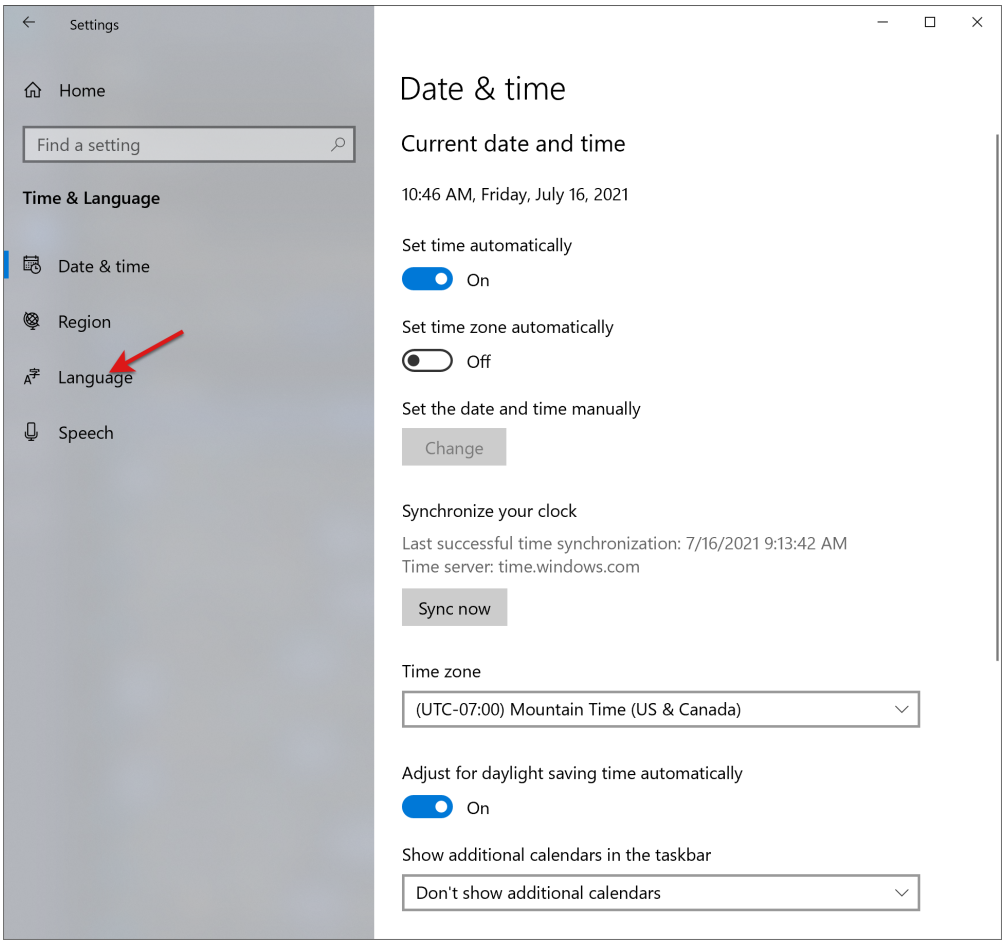
There is a known issue that NeuroExplorer may freeze on Windows 10 with Chinese and Japanese as preferred languages when the user edits the values in the dialog text boxes using the keyboard.

This is the documented problem with the Microsoft Input Method Editor (IME), which is a software component that enables a user to input text in a language that can't be represented easily on a standard QWERTY keyboard.

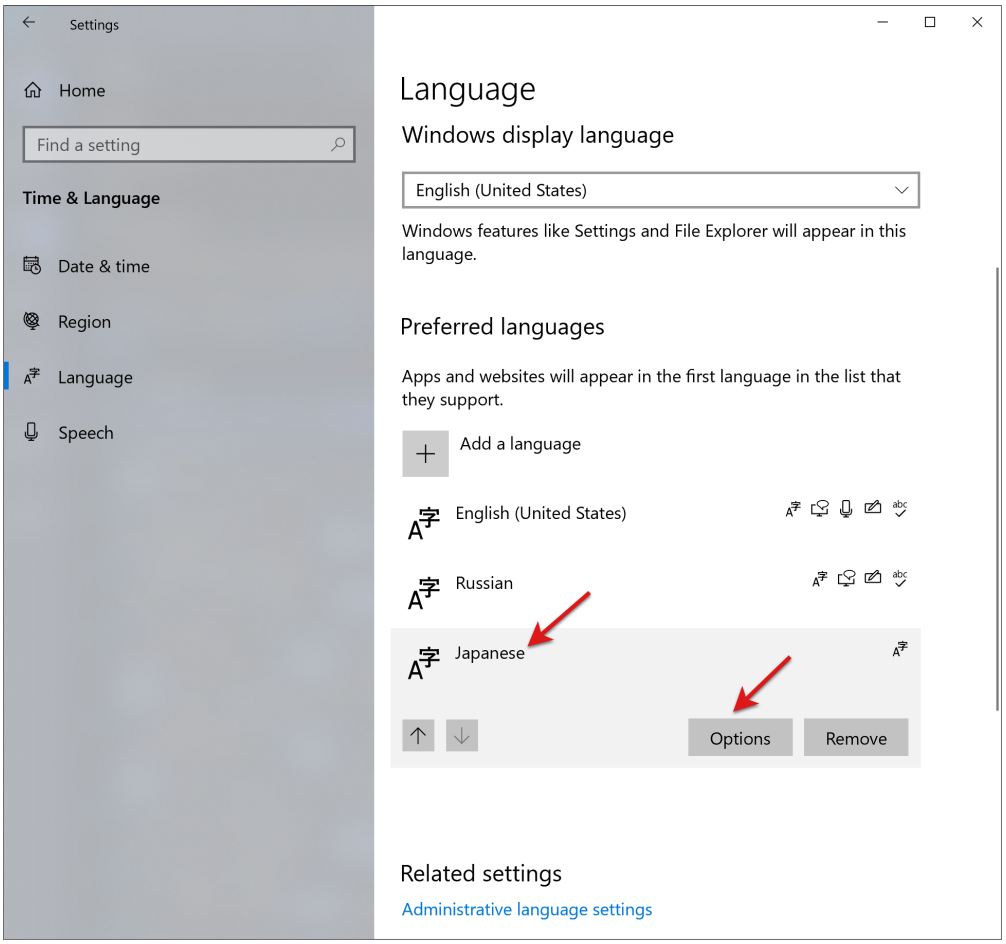
The workaround, suggested by Microsoft is to use the previous version of Input Method Editor. Select Start, type Settings, and select it or press enter. Select **Time & Language**:



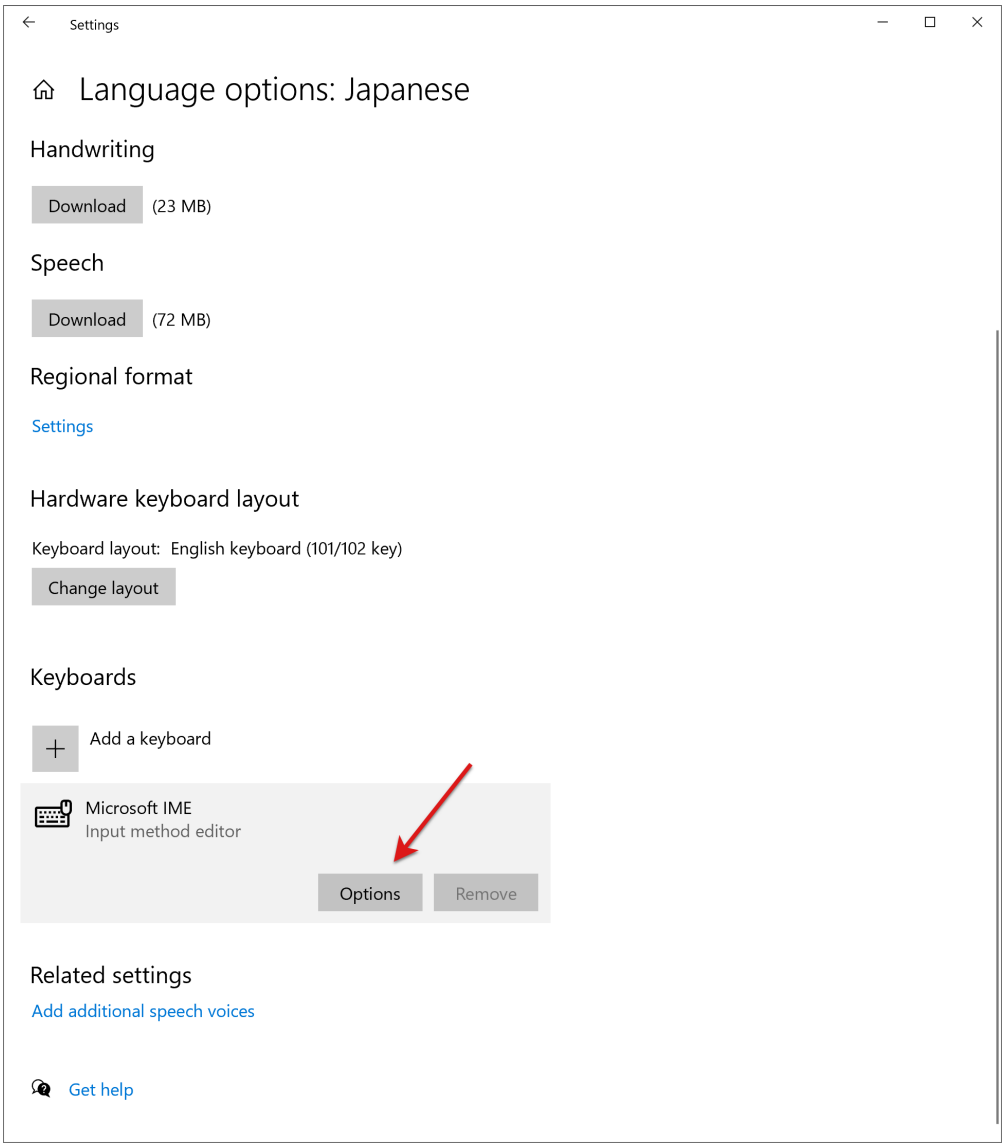
Select **Language**:



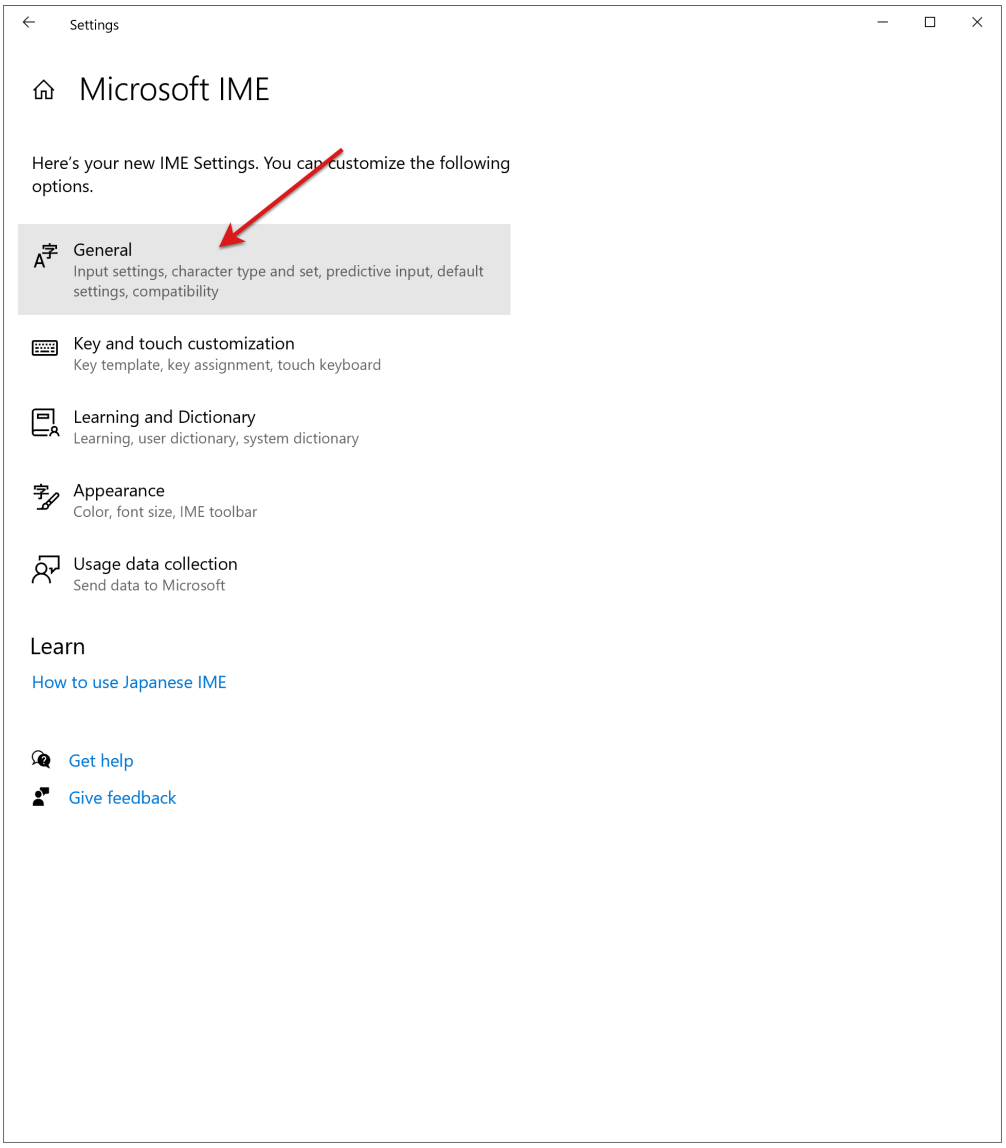
Select your language, for example Japanese, and click **Options**:



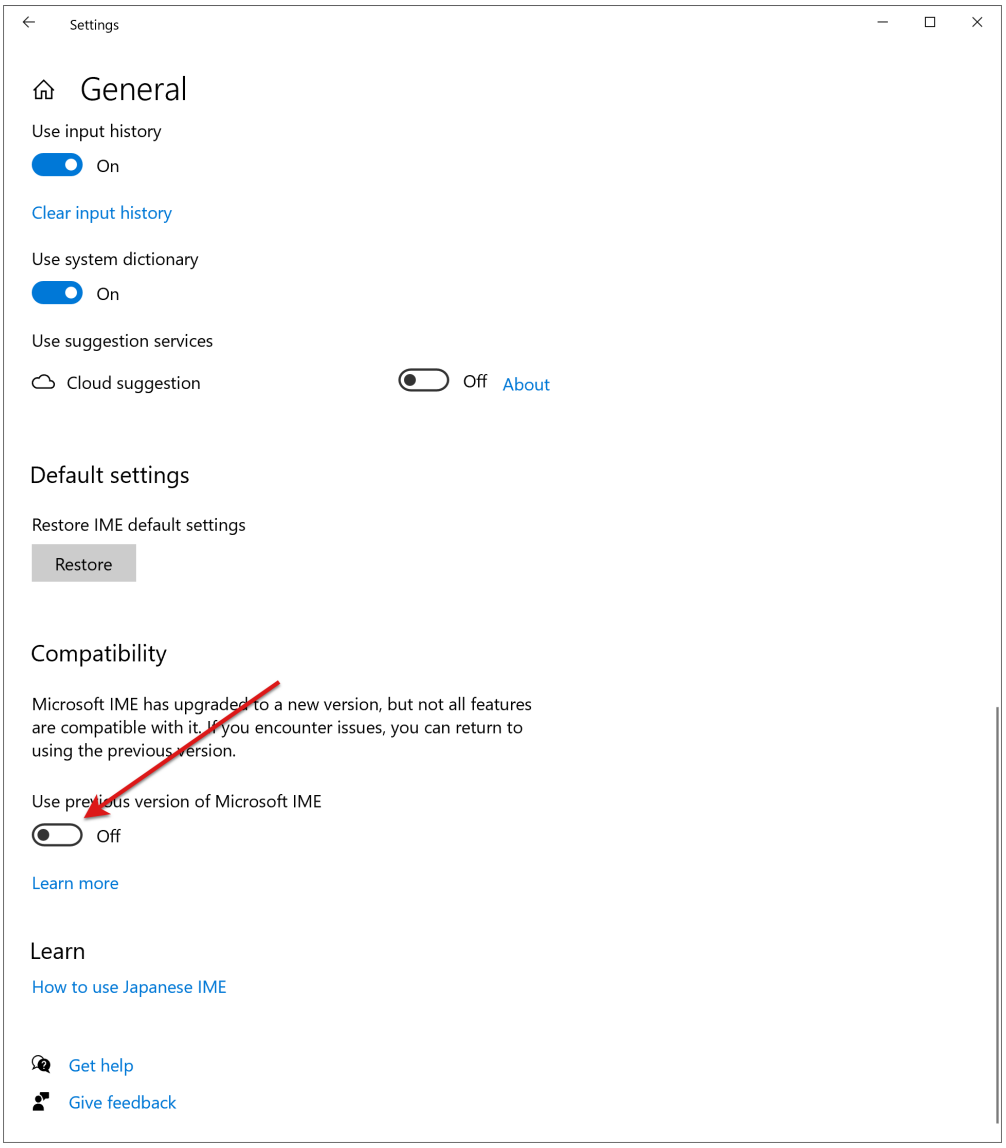
Select Microsoft IME and then click **Options**:



Select **General**:



Scroll down and turn on **Use previous version of Microsoft IME**:


Settings
General
Use input history
On
Clear input history
Use system dictionary
On
Use suggestion services
Cloud suggestion
Off
About
Default settings
Restore IME default settings
Restore
Compatibility
Microsoft IME has upgraded to a new version, but not all features are compatible with it. If you encounter issues, you can return to using the previous version.
Use previous version of Microsoft IME
Off
Learn more
Learn
How to use Japanese IME
Get help
Give feedback

CHAPTER

THREE

REFERENCE

Supported file formats are listed in *File Formats* topic.

See *Analysis Reference* for the list of analyses, their parameters and algorithms.

See *Scripting Reference* to view the documentation on Python and NexScript functions that allow you to automate repetitive operations and customize NeuroExplorer.

See *ActiveX Interface Reference* to view the documentation on how to control NeuroExplorer from Matlab.

3.1 Analysis Reference

3.1.1 Data Types

NeuroExplorer supports several data types: spike trains, behavioral events, time intervals, continuously recorded data and other data types. The topics in this section describe the data types used in NeuroExplorer, list the properties of data types, and show how to view and modify various data types in NeuroExplorer.

Spike Trains

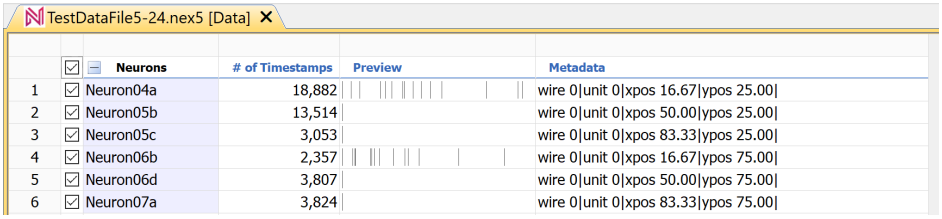
A spike train in NeuroExplorer represents spike timestamps (the times when the spikes occurred). The *Waveform* data type is used to store both the timestamps and the spike waveform values.

Note: *Events* are very similar to spike trains. The only difference between spike trains and events is that spike trains may contain additional information about recording sites, electrode numbers, cluster numbers, etc. (for example, spike trains contain positions of neurons used in the 3D activity "movie" shown via **3D View | 3D Activity Animation** menu command).

Internally, the timestamps are stored as 64-bit signed integers. These integers are usually the timestamps recorded by the data acquisition system and they represent time in the so-called time ticks. For example, the typical time tick for Plexon system is 25 microseconds, so an event recorded at 1 sec will have the timestamp equal to 40000.

Viewers

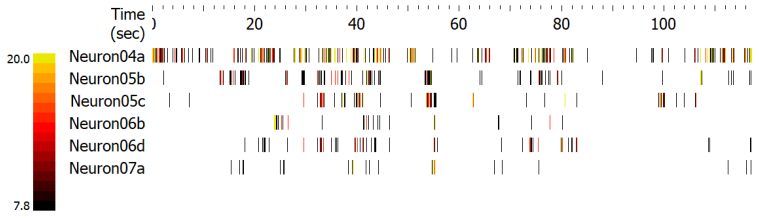
The **Preview** column of the variables panel shows the timestamps in the 1 second interval that starts with the first timestamp:



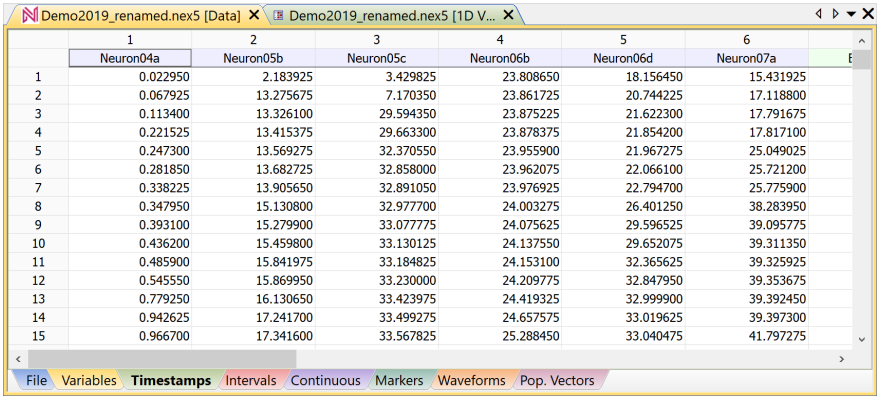
	Neurons	# of Timestamps	Preview	Metadata
1	Neuron04a	18,882		wire 0|unit 0|xpos 16.67|ypos 25.00|
2	Neuron05b	13,514		wire 0|unit 0|xpos 50.00|ypos 25.00|
3	Neuron05c	3,053		wire 0|unit 0|xpos 83.33|ypos 25.00|
4	Neuron06b	2,357		wire 0|unit 0|xpos 16.67|ypos 75.00|
5	Neuron06d	3,807		wire 0|unit 0|xpos 50.00|ypos 75.00|
6	Neuron07a	3,824		wire 0|unit 0|xpos 83.33|ypos 75.00|

The **Metadata** column shows electrode positions, electrode numbers, cluster numbers (unit numbers), etc.

You can view the timestamps of the selected variables in graphical display (**View | 1D Data Viewer** menu command):



Numerical values of the timestamps (in seconds) are shown in the **Timestamps** sheet of the Data view:



	1 Neuron04a	2 Neuron05b	3 Neuron05c	4 Neuron06b	5 Neuron06d	6 Neuron07a
1	0.022950	2.183925	3.429825	23.808650	18.156450	15.431925
2	0.067925	13.275675	7.170350	23.861725	20.744225	17.118800
3	0.113400	13.326100	29.594350	23.875225	21.622300	17.791675
4	0.221525	13.415375	29.663300	23.878375	21.854200	17.817100
5	0.247300	13.569275	32.370550	23.955900	21.967275	25.049025
6	0.281850	13.682725	32.858000	23.962075	22.066100	25.721200
7	0.338225	13.905650	32.891050	23.976925	22.794700	25.775900
8	0.347950	15.130800	32.977700	24.003275	26.401250	38.283950
9	0.393100	15.279900	33.077775	24.075625	29.596525	39.095775
10	0.436200	15.459800	33.130125	24.137550	29.652075	39.311350
11	0.485900	15.841975	33.184825	24.153100	32.365625	39.325925
12	0.545550	15.869950	33.230000	24.209775	32.847950	39.353675
13	0.779250	16.130650	33.423975	24.419325	32.999900	39.392450
14	0.942625	17.241700	33.499275	24.657575	33.019625	39.397300
15	0.966700	17.341600	33.567825	25.288450	33.040475	41.797275

Timestamped Variables in Python and NexScript

Python

In Python, use `Timestamps()` function to get the timestamps of a neuron, event, marker or a waveform variable:

```
import nex
doc = nex.GetActiveDocument()
# get all the timestamps for neuron SPK01a
ts = doc["SPK01a"].Timestamps()
```

To assign the timestamps in Python, use `SetTimestamps()` function:

```
import nex
doc = nex.GetActiveDocument()
# create ScriptGenerated event variable
doc["ScriptGeneratedEvent"] = nex.NewEvent(doc, 0)
# set all the timestamps of the new event
doc["ScriptGeneratedEvent"].SetTimestamps([ 1.0025,  2.5, 34.5])
```

NexScript

You can get access to any timestamp in the current file. For example, to print the value of the third timestamp of the variable `SPK01a`, you can use this script:

```
doc = GetActiveDocument()
Trace(doc["SPK01a"][3])
```

To assign the value 0.5 (sec) to the third timestamp of the variable `SPK01a`, you can use this script:

```
doc = GetActiveDocument()
doc["SPK01a"][3] = 0.5
```

Limitations

NeuroExplorer requires that, in each spike train, all the timestamps are in ascending order, that is

`timestamp[i+1] >= timestamp[i] for all i.`

NeuroExplorer also requires that

`timestamp[i] >= 0 for all i.`

Continuously Recorded Data

The continuous data type is used in NeuroExplorer to store the data that has been continuously recorded (LFP, EEG, EMG, etc.).

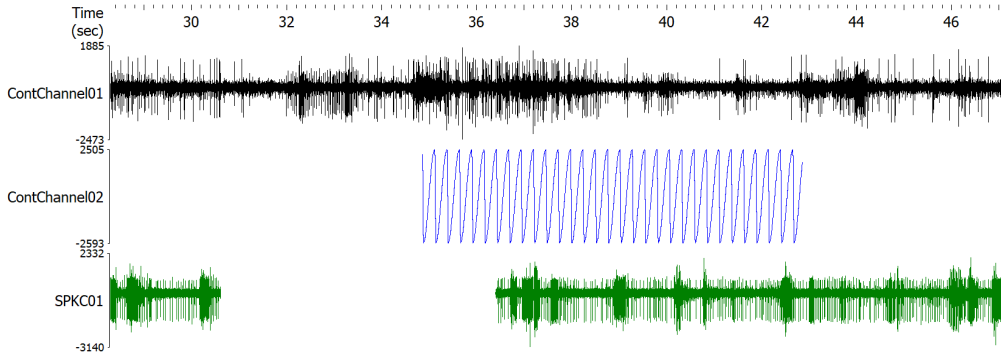
A continuous variable stores the measurements of some signal at the time points `t0`, `t0+dt`, `t0+2*dt`, etc.:

```
t0, value0
t0+dt, value1
t0+2*dt, value2
...
```

where `dt = 1/sampling_rate`.

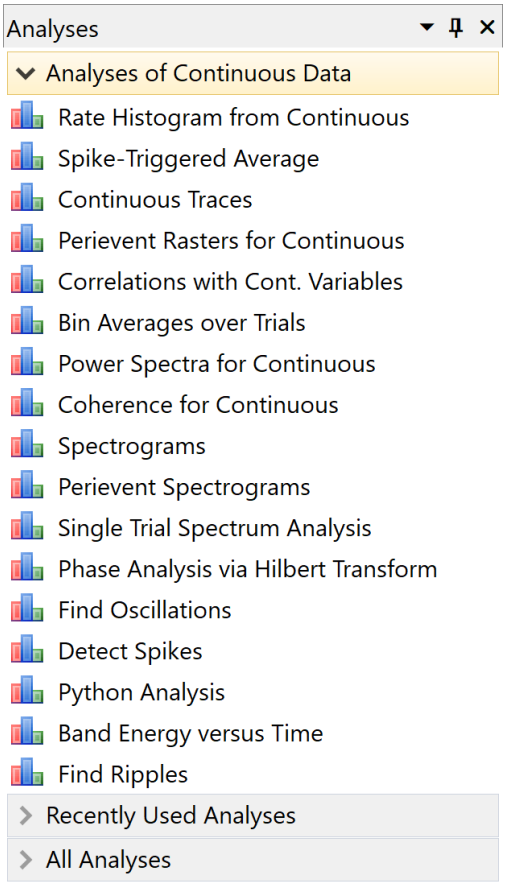
If the data timestamps have gaps, for example, `t0`, `t0+dt`, `t0+2*dt`, `t0+100*dt`, `t0+101*dt`, the signal is considered to have fragments (in this example, there is the first fragment with 3 values starting at `t0`, and the second fragment with 2 values starting at `t0+100*dt`).

The following figure shows continuous variables shown in 1D Viewer:



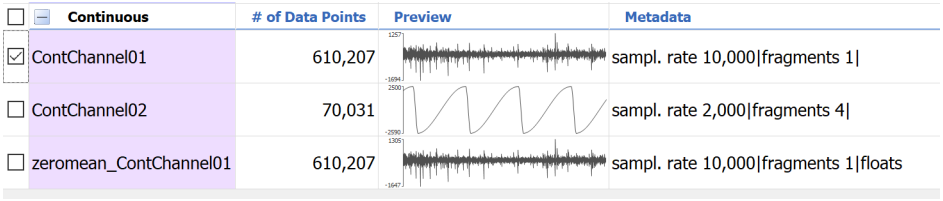
Note that the variable **SPKC001** is a continuous variable with fragments.

Continuous variables can be used in the analyses listed in *Analyses of Continuous Data* folder of *Analyses* panel:



Viewers

When you load a file with continuous data, NeuroExplorer displays the preview of the continuous variables in the **Variables** sheet of the Data view:



The **Preview** column shows the first 1 second of data. The Y axis of the preview column shows min and max of the signal in the first 1 second of data.

The **# of Data Points** column shows the number of measurements. The **Metadata** column shows the sampling rate in Hz and the number of fragments. If the values of a continuous variable is stored as 4-byte floats, the Metadata column shows the word `floats`.

You can view continuous variables in the graphical display (**View | 1D Data Viewer** menu command, see the top figure above).

Numerical values of the continuous variables are shown in the **Continuous** sheet of the Data view.

Continuous Variables in Python and NexScript

Python

In Python, use `ContinuousValues()` function to access the values of a continuous variable:

```
import nex
doc = nex.GetActiveDocument()
# get all the values for variable ContChannel01
values = doc["ContChannel01"].ContinuousValues()
```

To assign the continuous in Python, use `SetContVarStartTimeAndValues()` or `SetContVarTimestampsAndValues()` method:

```
import nex
doc = nex.GetActiveDocument()
doc["ScriptGenerated"] = nex.NewContVarWithFloats(doc, 1000)
doc["ScriptGenerated"].SetContVarStartTimeAndValues(0.5, [1, 2, 22.3])
```

NexScript

`ContVar[i,1]` gives you read-only access to the timestamp of the i-th data point.

`ContVar[i,2]` gives you read-write access to the value of the i-th data point.

For example, the following script prints the timestamp and the value of the fifth data point in variable ContChannel01:

```
doc = GetActiveDocument()
Trace("ts = ", doc.ContChannel01[5,1], "value =" ,doc.ContChannel01[5,2])
```

The following script line assigns the value of 100 to the fifth data point:

```
doc = GetActiveDocument()
doc.ContChannel01[5,2] = 100.
```

Limitations

Internally, each data point of the continuous variable is stored as a 2-byte integer or as a 4-byte floating point number. The maximum number of values (in each continuous variable) in NeuroExplorer is 2 billion.

Events

Event data type in NeuroExplorer is used to represent the series of timestamps recorded from external devices (for example, stimulation times recorded as pulses produced by the stimulus generator). Event data type stores only the times when the external events occurred. A special *Marker* data type is used to store the timestamps together with other stimulus or trial information

---

Note: Events are very similar to spike trains. The only difference between spike trains and events is that spike trains may contain additional information about recording sites, electrode numbers, etc.

Internally, event timestamps are represented exactly as *Spike Trains* timestamps. See *Spike Trains* for more information about the timestamps in NeuroExplorer.

Creating Event Variables via User Interface

You can add events to the data file using **Copy | Paste** command in the Timestamps view (see *Importing Data from Spreadsheets* for details).

You can also create additional events based on the events in the data file. Use **Edit | Operations on Data Variables** menu command and select one of the operations in the Operations dialog.

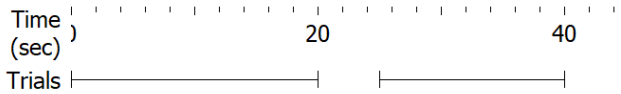
Event Variables in Python and NexScript

See *Spike Trains* for more information about accessing the timestamps using scripting in NeuroExplorer.

Intervals

Intervals are usually used in NeuroExplorer to select the data from the specified time periods (see *Data Selection Options* for details).

Each interval variable can contain multiple time intervals. For example, the `Trials` variable in the following figure has two intervals:



Creating Interval Variables via User Interface

You can create intervals in NeuroExplorer using **Edit | Add Interval Variable** menu command.

You can also create intervals based on existing Event or Interval variables. Use **Edit | Operations on Data Variables** menu command and then select one of the following operations:

- MakeIntervals
- MakeIntFromStart
- MakeIntFromEnd
- IntOpposite
- IntAnd
- IntOr
- IntSize
- IntFind

Interval Variables in Python and NexScript

Python

In Python, use `Intervals()` function to get interval start and end values. For example, the following script:

```
import nex
doc = nex.GetActiveDocument()
intervals = doc['Trials'].Intervals()
print(intervals)
```

will produce the following output:

```
[[0.0, 25.0], [20.0, 40.0]]
```

To add intervals in Python, use `AddInterval()` method:

```
import nex
doc = nex.GetActiveDocument()
doc['MyInterval'] = nex.NewIntEvent(doc)
doc['MyInterval'].AddInterval(0., 100.)
doc['MyInterval'].AddInterval(200., 300.)
```

NexScript

`IntVar[i,1]` gives you the access to the start of the i-th interval,

`IntVar[i,2]` gives you the access to the end of the i-th interval.

For example, the following script creates a new interval variable that has two intervals: from 0 to 100 seconds and from 200 to 300 seconds:

```
doc = GetActiveDocument()
doc["MyInterval"] = NewIntEvent(doc)
AddInterval(doc["MyInterval"], 0., 100.)
AddInterval(doc["MyInterval"], 200., 300.)
```

Limitations

Internally, NeuroExplorer stores the beginning and the end of each interval as a timestamp (see *Spike Trains* for more information about timestamps in NeuroExplorer).

If an interval variable contains multiple time intervals, these time intervals cannot overlap.

Viewers

You can view the intervals of the selected variables in graphical display (**View | 1D Data Viewer** menu command, see figure above).

The intervals (in seconds) are shown in the **Intervals** sheet of the Data view.

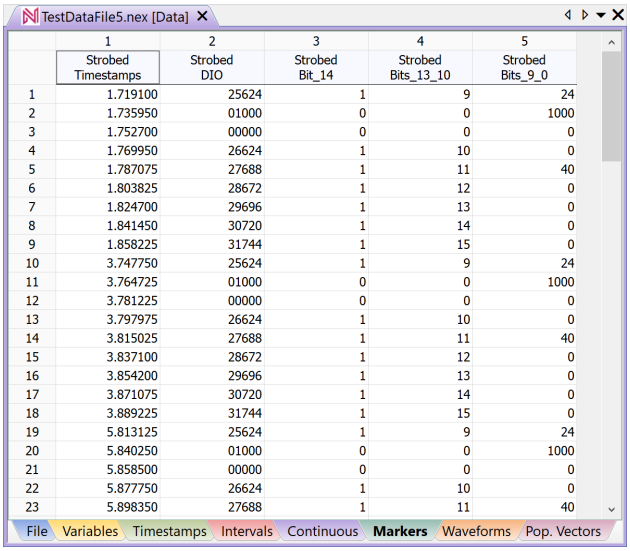
Markers

Marker data type in NeuroExplorer is used to represent the series of timestamps recorded from external devices (for example, stimulation times recorded as pulses produced by the stimulus generator) together with other stimulus or trial information. *Event* data type stores only the times when the external events occurred.

NeuroExplorer creates Marker variables when it imports, for example, strobed events from Plexon data files.

Viewers

You can view both the marker timestamps and additional marker information in the Markers tab of the Data window:


TestDataFile5.nex [Data]

	1 Strobed Timestamps	2 Strobed DIO	3 Strobed Bit_14	4 Strobed Bits_13_10	5 Strobed Bits_9_0
1	1.719100	25624	1	9	24
2	1.735950	01000	0	0	1000
3	1.752700	00000	0	0	0
4	1.769950	26624	1	10	0
5	1.787075	27688	1	11	40
6	1.803825	28672	1	12	0
7	1.824700	29696	1	13	0
8	1.841450	30720	1	14	0
9	1.858225	31744	1	15	0
10	3.747750	25624	1	9	24
11	3.764725	01000	0	0	1000
12	3.781225	00000	0	0	0
13	3.797975	26624	1	10	0
14	3.815025	27688	1	11	40
15	3.837100	28672	1	12	0
16	3.854200	29696	1	13	0
17	3.871075	30720	1	14	0
18	3.889225	31744	1	15	0
19	5.813125	25624	1	9	24
20	5.840250	01000	0	0	1000
21	5.858500	00000	0	0	0
22	5.877750	26624	1	10	0
23	5.898350	27688	1	11	40

File Variables Timestamps Intervals Continuous **Markers** Waveforms Pop. Vectors

Extracting Events

Usually you will need to extract events with the specific trial information from the marker variables. To do this, use **Marker | Split...** and **Marker | Extract...** menu commands.

Marker Variables in Python

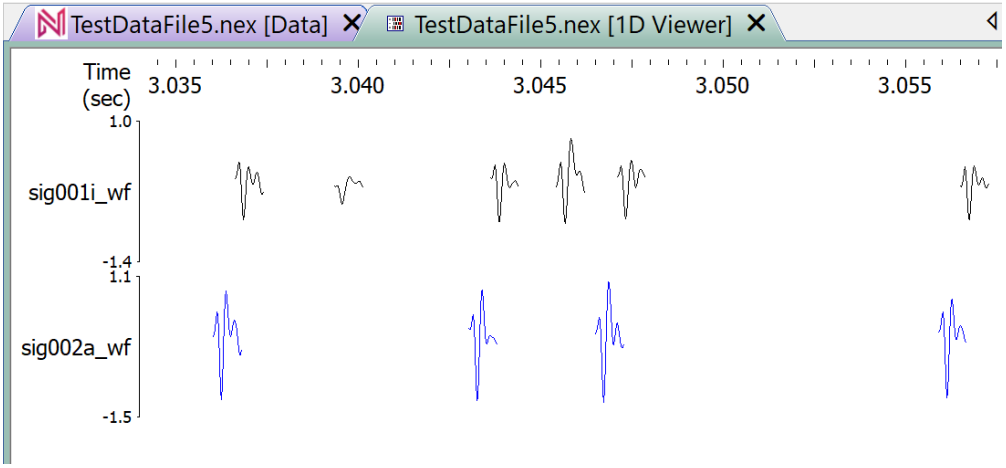
Python

Use `Markers()` function to get marker values of a marker variable:

```
import nex
doc = nex.GetActiveDocument()
markerValues = doc['Strobed'].Markers()
# print the second marker value of the first marker field
print(markerValues[0][1])
```

Waveforms

The waveform data type is used in NeuroExplorer to store the spike waveform values together with the spike timestamps. The following figure shows two waveform variables: sig001i_wf and sig002a_wf:



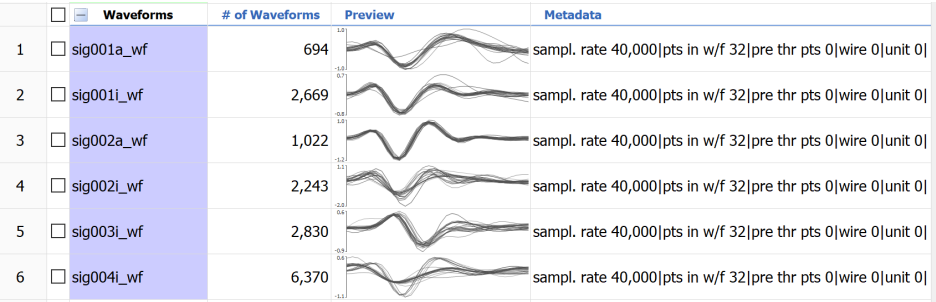
For most of the timestamp analyses, only the waveform timestamps are used. In the following analyses, the waveform shapes are used:

- Rate Histograms
- Rasters
- Perievent Histograms (instead of PST, NeuroExplorer calculates spike-triggered average for a waveform variable)

- Waveform Comparison
- Sort Spikes

Viewers

The **Preview** column of the variables panel shows the first 20 waveforms (superimposed). The Y axis of the preview column shows min and max of the first 20 waveforms.

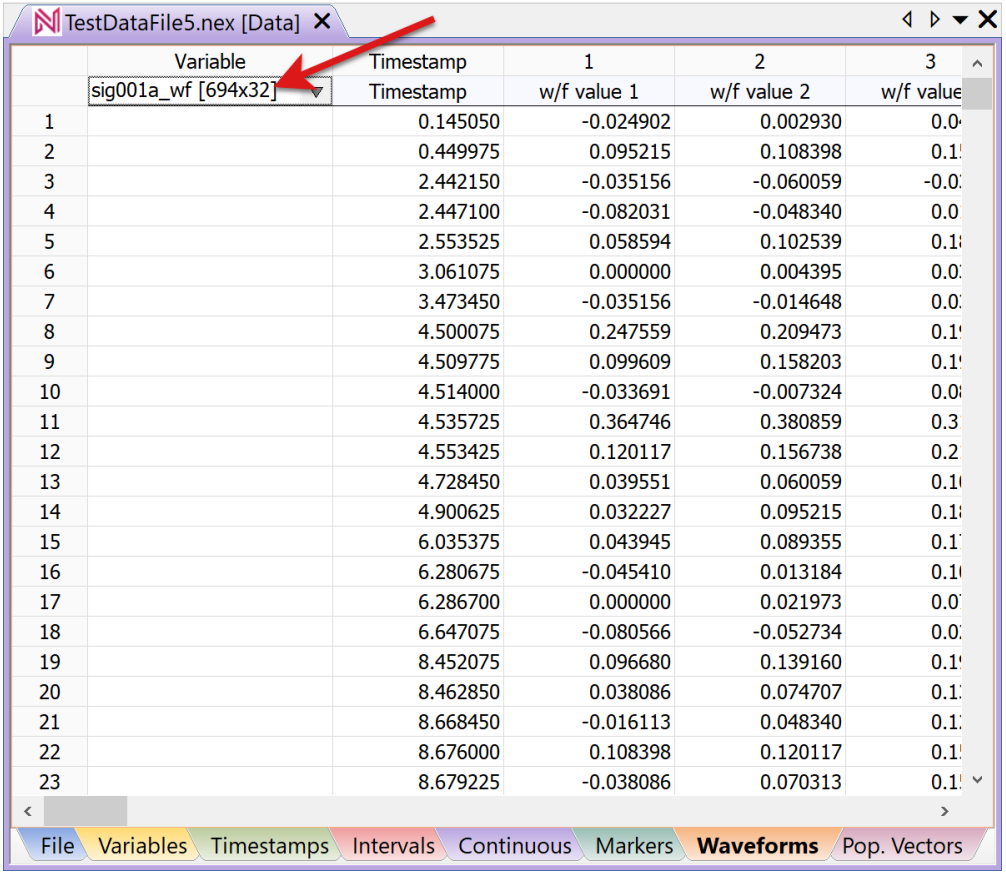


	Waveforms	# of Waveforms	Preview	Metadata
1	sig001a_wf	694		sampl. rate 40,000|pts in w/f 32|pre thr pts 0|wire 0|unit 0|
2	sig001i_wf	2,669		sampl. rate 40,000|pts in w/f 32|pre thr pts 0|wire 0|unit 0|
3	sig002a_wf	1,022		sampl. rate 40,000|pts in w/f 32|pre thr pts 0|wire 0|unit 0|
4	sig002i_wf	2,243		sampl. rate 40,000|pts in w/f 32|pre thr pts 0|wire 0|unit 0|
5	sig003i_wf	2,830		sampl. rate 40,000|pts in w/f 32|pre thr pts 0|wire 0|unit 0|
6	sig004i_wf	6,370		sampl. rate 40,000|pts in w/f 32|pre thr pts 0|wire 0|unit 0|

The **Metadata** column shows the sampling rate, the number of data points in each waveform as well as the electrode positions, electrode numbers, cluster numbers (unit numbers), etc.

You can view the waveforms of the selected variables in the graphical display (**View | 1D Data Viewer** menu command, see the first figure above).

Numerical values of the waveforms and their timestamps are shown in the **Waveforms** sheet of the Data view.



To select a different waveform variable, click in the cell located in the row 1 of the Variable column (the cell shows `sig001a_wf` in the figure above).

Waveform values (in milliVolts) are shown in columns labeled 1, 2, etc.

Waveform Variables in Python

Use the `WaveformValues()` method to get the waveform values of a waveform variable:

```
import nex
doc = nex.GetActiveDocument()
waveforms = doc['sig001a_wf'].WaveformValues()
# print the values of the first waveform
print(waveforms[0])
```

Population Vectors

Population vectors in NeuroExplorer can be used to display the linear combinations of histograms in some of the analyses. For example, you can calculate perievent histograms for each individual neuron recorded in a data file. If you want to calculate the response of the whole population of neurons (that is, create an average PST histogram) you need to use a population vector.

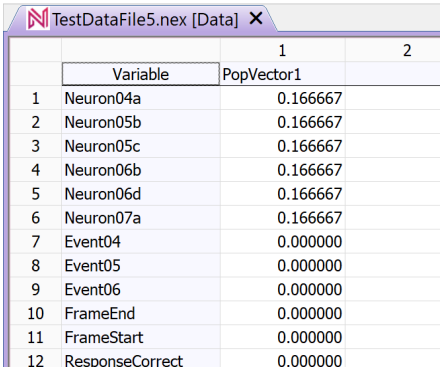
Population vector assigns a weight to each spike train or event variable in the file. You can then use population vectors in the following analyses:

- Rate Histograms
- Interspike Interval Histograms
- Perievent Histograms
- Trial Counts

If you select the population vector for analysis and then run one of the four analyses listed above, the histogram corresponding to the population vector will be calculated as:

```
histogram_of_neuron_1 * weight_1 + histogram_of_neuron_2 * weight_2...
```

where the weights are defined in the population vector. For example, to calculate an average histogram for 6 neurons, the following population vector should be used:


TestDataFile5.nex [Data]

		1	2
	Variable	PopVector1	
1	Neuron04a	0.166667	
2	Neuron05b	0.166667	
3	Neuron05c	0.166667	
4	Neuron06b	0.166667	
5	Neuron06d	0.166667	
6	Neuron07a	0.166667	
7	Event04	0.000000	
8	Event05	0.000000	
9	Event06	0.000000	
10	FrameEnd	0.000000	
11	FrameStart	0.000000	
12	ResponseCorrect	0.000000	

Creating Population Vectors

You can create new population vectors in NeuroExplorer using **Edit | Add Population Vector** menu command.

Note: *Principal Component Analysis* creates a set of population vectors based on the covariances between activities of individual neurons.

3.1.2 Analysis Types

Spike Train Structure

Rate Histograms

Rate histogram displays firing rate versus time.

Parameters

Parameter	Description
XMin/XMax type	An option on how XMin and XMax values are specified.
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Bin	Histogram bin size in seconds.
Normalization	Histogram units (Counts/Bin or Spikes/Second). See *Algorithm* below.
Set Cont. Mean to Zero if Small Bin Count	If the number of continuous data points int a bin is too small, set cont. mean (bin value) to zero. See *Algorithm* below.
Cont. Min Bin Count Percent	Minimum number of continuous data points in a bin as percent of the expected number of data points. See *Algorithm* below.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Smooth histogram	An option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options* .
Matrix Name	Specifies the name of the results matrix in Matlab workspace.

continues on next page

Table 1 – continued from previous page

Parameter	Description
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values.
St. Err. Mean. Hist.	The standard error of mean of the histogram bin values.

Algorithm

The time axis is divided into bins. The first bin is `[XMin, XMin+Bin)`. The second bin is `[XMin+Bin, Xmin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin.

Spike Trains and Events

For each bin, the number of events (timestamps) in this bin is calculated.

For example, for the first bin

```
bin_count = number of timestamps (ts) such that ts >= XMin and ts < XMin + Bin
```

If **Normalization** is Counts/Bin, no further calculations are performed.

If **Normalization** is Spikes/Sec, bin counts are divided by **Bin**.

Continuous Channels

For each bin, the average of continuous signal values in this bin is calculated. Normalization parameter is ignored.

When calculating the average of continuous signal, we may encounter a situation when the number of data points in a bin is very small. For example, with a 1 KHz signal and bin = 1 second, we typically have 1000 data points in each bin. If we have a bin in which we have only one data point, the average of the signal in this bin is equal to the single data point value. This value can be very different from the typical average of 1000 data points. Therefore, to avoid these spurious artifacts, we need to check how many data points are in the bin and set the average to zero if there are too few data points.

Firing Rates

Firing Rates analysis calculates and displays firing rates in specified time intervals or series of time intervals.

Parameters

Parameter	Description
Time Period 1 Type	How period 1 is specified (None, Single Interval or Interval Variable).
Period 1 From (sec)	If period 1 type is Single Interval, interval start in seconds.
Period 1 To (sec)	If period 1 type is Single Interval, interval end in seconds.
Period 1 Interval Var.	If period 1 type is Interval Variable, selected interval variable.
Period 1 Inside/Outside	An option to calculate firing rates inside or outside specified time interval(s).
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options* .
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Period Spec 1	Period 1 specification.
Period Length 1	Length of period 1 in seconds.
Spikes in Period 1	The number of spikes in period 1.
Firing rate in Period 1	Firing Rate in period 1 (spikes per second)

Algorithm

For each specified interval or interval variable (series of time intervals), the number of events (timestamps) within the time interval(s) is calculated. The number of timestamps is then divided by the duration of the intervals(s).

Autocorrelograms

Autocorrelogram shows the conditional probability of a spike at time `t0+t` on the condition that there is a spike at time `t0`.

Parameters

Parameter	Description
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds.
Bin	Bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability or Spikes/Second). See *Algorithm* below.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	**Offset (in seconds, relative to the event specified in** *Create filter around* parameter) for the end of interval for the temporary filter.

continues on next page

Table 5 – continued from previous page

Parameter	Description
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Overlay Graphs	An option to draw several histograms in each graph. This option requires that *Int. filter type* specifies that multiple interval filters will be used (either *Table (row)* or *Table (col)*).
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Draw confidence limits	An option to draw the confidence limits.
Confidence (%)	Confidence level (percent). See *Confidence Limits* for details.
Conf. mean calculation	An option that specifies how the mean firing rate (that is used in the derivation of the confidence limits) is calculated. There are 2 options: *Use data selection* and *Use all file*. See *Confidence Limits* for details.
Conf. display	An option to draw confidence limits either as horizontal lines or as a colored background.
Conf. line style	Line style for drawing confidence limits (used when Conf. display is *Lines*).
Conf. background color	Background color for drawing confidence limits (used when Conf. display is *Colored Background*).
Draw mean freq.	An option to draw a horizontal line representing the expected histogram value for a Poisson spike train. See *Confidence Limits* for details.
Mean line style	Line style for drawing mean frequency.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.

continues on next page

Table 5 – continued from previous page

Parameter	Description
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values.
Conf. Low	Lower confidence level.
Conf. High	Upper confidence level.
Mean	Expected mean value of the histogram.
Norm. Factor	Normalization factor. Bin counts are divided by this value. See Normalization in *Algorithm* below.
Time of Minimum	Position of the histogram minimum (in seconds). If there are multiple bins in the histogram where the bin value is equal to the histogram minimum, this value represents the position of the first such bin.
Time of Maximum	Position of the histogram maximum (in seconds). If there are multiple bins in the histogram where the bin value is equal to the histogram maximum, this value represents the position of the first such bin.

Algorithm

In general, the Autocorrelogram shows the conditional probability of a spike in the spike train at time `t` on the condition that there is a spike at time zero.

The time axis is divided into bins. The first bin is `[XMin, XMin+Bin)`. The second bin is `[XMin+Bin, XMin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin.

Let `ts[i]` be the spike train (each ts is the timestamp).

For each timestamp `ts[k]`:

calculate the distances from this spike to all other spikes in the spike train:

```
d[i] = ts[i] - ts[k]
```

for each `i` except `i` equal to `k`:

if `d[i]` is inside the first bin, increment the bin counter for the first bin:

```
if d[i] >= XMin and d[i] < XMin + Bin
then bincount[1] = bincount[1] + 1
```

if d[i] is inside the second bin, increment the bin counter for the second bin:

```
if d[i] >= XMin+Bin and d[i] < XMin + Bin*2
then bincount[2] = bincount[2] + 1
```

and so on… .

If **Normalization** is **Counts/Bin**, no further calculations are performed.

If **Normalization** is **Probability**, bin counts are divided by the number of spikes in the spike train.

Note that the Probability normalization makes sense only for small values of Bin. For Probability normalization to be valid (so that the values of probability are between 0 and 1), there should be no more than one spike in each bin. For example, if the Bin value is large and for each`` ts[k]`` above there are many `d[i]` values such that `d[i] >= XMin and d[i] < XMin + Bin`, the bin count for the first bin can exceed the number of spikes in the spike train. Then, the probability value `(bincount[1]/number_of_spikes)` could be larger than 1.

If **Normalization** is **Spikes/Sec**, bin counts are divided by `NumSpikes*Bin`, where `NumSpikes` is the number of spikes in the spike train.

Autocorrelograms Versus Time

This analysis shows the dynamics of the autocorrelograms over time. It calculates multiple autocorrelograms using a "sliding window" in time. Each autocorrelogram is shown as a vertical stripe with colors representing the bin counts. Horizontal axis represents the position of the sliding window in time.

Parameters

Parameter	Description
XMin AutoCorr	Autocorrelogram time axis minimum in seconds.
XMax AutoCorr	Autocorrelogram time axis maximum in seconds.
Bin AutoCorr	Autocorrelogram bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability or Spikes/Second). See *Algorithm* below.
Start	Start of the first sliding window in seconds.
Duration	Duration of the sliding window in seconds.
Shift	How much sliding window is shifted each time.
Number of shifts	The number of sliding windows to be used.
Matrix Scale	Specifies color scale for drawing matrix.
X Axis (Sliding window axis alignment)	Allows to specify what values are shown in the sliding window axis. There are two options: Start of Window and Center of Window. Suppose your data has the strongest correlation at 50 seconds. This means that the window that has 50 seconds as its center will show the highest autocorrelation values. If the window width is 20 seconds, it will be the window [40,60]. If you are using 'Start of window' option, you will see the peak at 40 instead of 50 seconds. However, with 'Center of window' option, the peak will be at 50 seconds.
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.

continues on next page

Table 7 – continued from previous page

Parameter	Description
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Color Scale Min	Color scale minimum.
Color Scale Max	Color scale maximum.
Spikes	Number of spikes used

Algorithm

NeuroExplorer calculates here multiple autocorrelograms (see *Autocorrelograms* for more information on how each autocorrelogram is calculated).

Each autocorrelogram is calculated for a different interval (window) in time:

```
``[window_start[i], window_end[i]], i = 1,...,Number_of_Shifts``
```

That is, for each autocorrelogram only the timestamps that are within the window are used.

The following rules are used to calculate the windows:

```
window_start[1] = Start

window_end[1] = Start + Duration

window_start[2] = Start + Shift

window_end[2] = Start + Shift + Duration

...
```

Interspike Interval Histograms

Interspike interval histogram shows the conditional probability of the *first* spike at time `t0+t` after a spike at time `t0`.

Parameters

Parameter	Description
Min Interval	Minimum interspike interval in seconds.
Max Interval	Maximum interspike interval in seconds.
Bin	Bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability or Spikes/Second). See *Algorithm* below.
Log Bins and X Axis	An option to use Log10 interval scale (see *Algorithm* below for details).
Bins per decade	A bin size option if Log option is selected (see *Algorithm* below for details).
Y Axis Limit (%)	An option to 'zoom in' small histogram values. This parameter specifies the maximum of Y axis. If it is 100%, the Y axis maximum is equal to the maximum bin value. If it is, for example, 10%, Y axis maximum is set to 10% of the bin max and the small bin values have better visibility.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.

continues on next page

Table 9 – continued from previous page

Parameter	Description
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Overlay Graphs	An option to draw several histograms in each graph. This option requires that *Int. filter type* specifies that multiple interval filters will be used (either *Table (row)* or *Table (col)*).
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Add Bin Left to Results	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add Bin Middle to Results	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add Bin Right to Results	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values.
St. Err. Mean. Hist.	The standard error of mean of the histogram bin values.
Mean ISI	The mean of the interspike intervals.
St. Dev. ISI	The standard deviation of the interspike intervals.
Coeff. Var. ISI	Coefficient of variation of the interspike intervals.
Median ISI	Median of the interspike intervals.
Mode ISI	Mode of the interspike intervals.

Algorithm

If Use Log Bins and X Axis option is not selected

The time axis is divided into bins. The first bin is `[IntMin, IntMin+Bin)`. The second bin is `[IntMin+Bin, Intmin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin. For each bin, the number of interspike intervals within this bin is calculated.

For example, for the first bin

```
bin_count = number of interspike intervals (isi)
such that isi >= IntMin and isi < IntMin + Bin
```

If **Normalization** is **Counts/Bin**, no further calculations are performed.

If **Normalization** is **Probability**, bin counts are divided by the number of interspike intervals in the spike train.

If **Normalization** is **Spikes/Sec**, bin counts are divided by `NumInt*Bin`, where `NumInt` is the number of interspike intervals in the spike train.

If Use Log Bins and X Axis option is selected

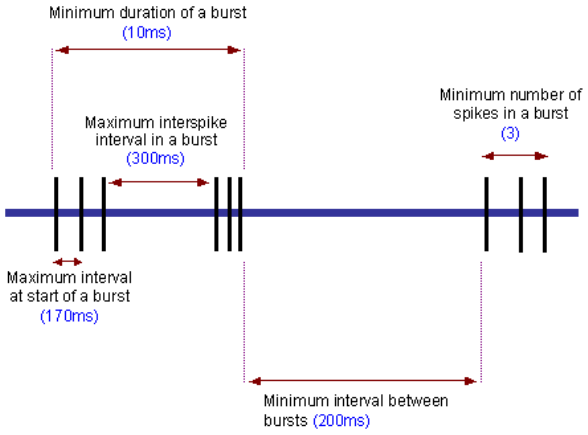
The i-th bin (i=1,2,...) is `[IntMin * 10 ^ ((i -1)/D), IntMin * 10 ^ (i/D))`, where `D` is the **Number of Bins Per Decade.** For each bin, the number of interspike intervals within this bin is calculated. For discussion on using logarithm of interspike intervals, see:

Alan D. Dorval, Probability distributions of the logarithm of inter-spike intervals yield accurate entropy estimates from small datasets. Journal of Neuroscience Methods 173 (2008) 129-139

Burst Analysis

This analysis identifies bursts in spike trains. Burst start and end times, duration of each burst and other burst statistics are calculated.

For each neuronal variable, a new interval variable can also be created that would contain the time intervals corresponding to bursts.



Parameters

Parameter	Description
Algorithm	Selection of MaxInterval, Surprise or Firing Rate Based algorithms.
Max Interval	Maximum interspike interval to start the burst (in seconds, MaxInterval method).
Max End Interval	Maximum interspike interval to end the burst (in seconds, MaxInterval method).
Min Interburst Interval	Minimum interval between bursts (in seconds, MaxInterval method).

continues on next page

Table 11 – continued from previous page

Parameter	Description
Min Burst Duration	Minimum burst duration (in seconds, MaxInterval method).
Min Number of Spikes	Minimum number of spikes in the burst (MaxInterval method).
Min Surprise	Minimum surprise of the burst (Surprise method).
Surprise Method Min. Num. Spikes	Minimum number of spikes in the burst (Surprise method).
Surprise Method Min. Duration	Minimum burst duration (Surprise method).
Surprise Merge Bursts	Merge bursts (Surprise method).
Surprise Min. Interburst Interval (sec)	Merge bursts if interburst interval is less than this parameter (Surprise method).
Firing Rate Method Bin	Firing Rate Method Bin (seconds).
Firing Rate Method Smooth Width	Width of the Gaussian filter used in Firing Rate Method.
Firing Rate Method Threshold	Firing Rate Method Threshold (number of standard deviations).
Firing Rate Method Minimum Number of Spikes in Burst	Firing Rate Method Minimum Number of Spikes in Burst.
Display	Specifies what burst statistics to display.
Add Burst Interval Vars.	An option to create interval variables containing time intervals corresponding to bursts.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Burst rate histogram minimum.
YMax	Burst rate histogram maximum.
Spikes	The number of spikes used in spectrum calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Mean Frequency	Mean firing rate (Spikes/Filter_Length).
Num Bursts	Number of bursts.
Bursts Per Second	Burst rate in bursts per second.
Bursts Per Minute	Burst rate in bursts per minute.
% of Spikes in Bursts	Percent of spikes in bursts.
Mean Burst Duration	Mean duration of burst (in seconds).
St. Dev. of Burst Duration	Standard deviation of burst duration.
Mean Spikes in Burst	Average number of spikes in burst.
St. Dev. of Spikes in Burst	Standard deviation of the number of spikes in burst.
Mean ISI Burst	Mean interspike interval in burst.
St. Dev of ISI in Burst	Standard deviation of interspike intervals in burst.
Mean Freq. in Burst	Mean firing rate `(1/interspike_interval)` in burst.
St. Dev. of Freq. in Burst	Standard deviation of `(1/interspike_interval)` in burst.
Mean Peak Frequency in Burst	Mean of `(1/min_interspike_interval)`, where `min_interspike_interval` is the minimum interspike interval in a burst.
St. Dev. Peak Frequency in Burst	Standard deviation of `(1/min_interspike_interval)`, where `min_interspike_interval` is the minimum interspike interval in a burst.
Mean Interburst Interval	Average length (in seconds) of interburst interval. Interburst interval is the interval from the end of the previous burst to the start of the current burst.
St. Dev. of Interburst Interval	Standard deviation of the interburst intervals.
Mean Burst Surprise	Average burst surprise.
St. Dev. Burst Surprise	Standard deviation of burst surprise.

Algorithm

For each spike train, NeuroExplorer identifies bursts and calculates burst start and end times, burst duration, number of spikes in burst, mean ISI in burst and peak frequency (`1/min_interspike_interval`) in burst. Optionally, creates a new Interval variable and stores all the burst intervals in this variable.

MaxInterval Method

Find all the bursts using the following algorithm:

- Scan the spike train until an interspike interval is found that is less than or equal to **Max. Interval**.
- While the interspike intervals are less than **Max. End Interval,** they are included in the burst.
- If the interspike interval is more than **Max. End Interval**, the burst ends.
- Merge all the bursts that are less than **Min. Interval Between Bursts** apart.
- Remove the bursts that have duration less than **Min. Duration of Burst** or have fewer spikes than **Min. Number of Spikes**.

Surprise Method

1. First, the mean firing rate (`Freq`) and mean interspike interval (`MeanISI`) of the neuron are calculated.

```
Freq = NumberOfSpikes / (FileEndTime - FileStartTime)
MeanISI = 1 / Freq
ISIToStartBurst = MeanISI / 2
ISIToEndBurst = MeanISI
```

2. NeuroExplorer scans the spike train until it finds two sequential ISI's so that each of those ISIs is less than `ISIToStartBurst`. The surprise of the resulting 3-spike sequence is calculated:

   If we assume that a random variable `P` has a Poisson distribution with parameter `Freq` and we also assume that the burst has `N` spikes and the distance from the first to the last spike of the burst is `T`, then the surprise of the burst is:

```
S = - log10 (Probability that P has at least N points in time interval␣
↪of length T)
```

3. NeuroExplorer adds the spikes to the end of the burst until the first ISI that is more than `ISIToEndBurst` and calculates surprise for each of the bursts (with 3 initial spikes, 4 spikes, 5 spikes, etc.). The burst with maximum surprise `Smax` is then selected.
4. NeuroExplorer removes the spikes from the beginning of the burst and calculates the surprise for each of the reduced bursts. The burst with maximum surprise `Smax` is then selected.
5. If `Smax` is more than `MinSurprise` and the number of spikes in the burst is more than **Surprise Method Min. Num. Spikes**, NeuroExplorer adds the burst to the result.

Firing Rate Based Method

1. Firing rate histogram with the specified bin size is calculated.
2. Rate histogram is smoothed using Gaussian smooth filter of the specified width. See *Post-Processing Options* for details of filter design.
3. Mean and Standard Deviation (STD) of the smoothed rate histogram are calculated.
4. Bins that have smooth histogram values of more than `Mean+STD*Firing_Rate_Method_Threshold` are considered to be in the burst. Burst start is the first spike in the burst bins, burst end is the last spike in the burst bins.
5. Bursts that have fewer spikes than the specified **Firing Rate Method Minimum Number of Spikes in Burst** are removed.

Reference

Legendy C.R. and Salcman M. (1985): Bursts and recurrences of bursts in the spike trains of spontaneously active striate cortex neurons. *J. Neurophysiology*, 53(4):926-39.

Joint ISI

Joint ISI Distribution

For each spike, a point is calculated where the X coordinate of the point is the current interspike interval and Y coordinate of the point is the preceding interspike interval. The density of these points is shown in the graph.

Parameters

Parameter	Description
Min Interval	Time axis minimum in seconds.
Max Interval	Time axis maximum in seconds
Bin	Bin size in seconds.
Log Axis Scale	An option to use Log10 axis scales.
Matrix Scale	Matrix color scale.
Smooth Matrix	An option to smooth matrix after calculation.
Smooth Radius	Radius of the smooth filter (in bins).
Smooth Colors	An option to smooth colors using bicubic splines.
Log Color Scale	An option to use Log10 color scale.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Color Scale Min	Color scale minimum
Color Scale Max	Color scale maximum

Algorithm

If Use Log Bins and Axes option is not selected

Matrix of `binCounts[x,y]` is created. Each element of the matrix is initialized as zero.

For each spike that occurred at time `t[i]`, the following values are calculated:

```
Interval_I = t[i] - t[i-1]
binX = (Interval_I - MinInterval)/Bin
Interval_I_Plus_1 = t[i+1] - t[i]
binY = (Interval_I_Plus_1 - MinInterval)/Bin
```

Then, the corresponding bin count is incremented:

```
binCounts[binX, binY] = binCounts[binX, binY] + 1
```

The graph shows binCounts matrix values using color scale.

If Use Log Bins and Axes option is selected

The i-th bin (i=1,2,...) is `[IntMin * 10 ^ ((i -1)/D), IntMin * 10 ^ (i/D))`, where `D` is the **Number of Bins Per Decade.**

```
binX = (log10(Interval_I) - log10(MinInterval))* NumBinsPerDecade
binY = (log10(Interval_I_Plus_1) - log10(MinInterval))* NumBinsPerDecade
```

Reference

For discussion on using logarithm of interspike intervals, see:

Alan D. Dorval, Probability distributions of the logarithm of inter-spike intervals yield accurate entropy estimates from small datasets. Journal of Neuroscience Methods 173 (2008) 129-139

Poincare Maps

For each spike, a point is displayed where the X coordinate of the point is the current interspike interval and Y coordinate of the point is the preceding interspike interval.

Parameters

Parameter	Description
Min Interval	Time axis minimum in seconds.
Max Interval	Time axis maximum in seconds
Log Axis Scale	An option to use Log10 axis scales.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Spikes	The number of spikes used in calculation

continues on next page

Table 16 – continued from previous page

Column	Description
Mean Freq.	Mean firing rate.

Algorithm

For each spike that occurred at time `t[i]`, Poincare plot shows the point with the coordinates

```
(t[i] - t[i-1], t[i-1] - t[i-2]).
```

That is, the `X` coordinate of the point is the current interspike interval and the `Y` coordinate of the point is the preceding interspike interval.

Hazard Analysis

This analysis calculates a hazard function – the conditional probability of a spike at time `t0+t` on the condition that there is a spike at time `t0` and there are no spikes in the interval `[t0, t0+t)`.

Parameters

Parameter	Description
Max IIH Interval (sec)	Maximum interspike interval used in calculation of interspike interval histogram (see Algorithm below).
Hazard Time Max (sec)	Time axis maximum in seconds. This parameter is expected to be smaller than Max IIH Interval (sec). See Algorithm below.
Bin	Bin size in seconds.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).

continues on next page

Table 17 – continued from previous page

Parameter	Description
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Overlay Graphs	An option to draw several histograms in each graph. This option requires that *Int. filter type* specifies that multiple interval filters will be used (either *Table (row)* or *Table (col)).*
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Hazard minimum.
YMax	Hazard maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hazard.	The mean of the hazard values.

Algorithm

First, the program calculates the interspike interval histogram using interspike intervals that are less than **Max IIH Interval**.

The hazard function is then calculated according to the following formula (see **Reference** below):

```
(hazard in bin [t, t+dt]) = (number of intervals in bin [t,t+dt])/(number of␣
↪intervals of length > t)
```

where `dt` is the bin size.

Hazard Time Max (which specifies the last bin of the hazard function) is expected to be smaller than **Max IIH Interval**.

If **Hazard Time Max = Max IIH Interval**, the hazard value for the last bin is always 1. Then the maximum of Y axis is set to 1 and smaller hazard values are scaled down. To avoid this scaling down, specify a smaller value of **Hazard Time Max**.

Reference

N.Sabatier, et.al., Phasic spike patterning in rat supraoptic neurones *in vivo* and *in vitro*, *J. Physiol.* 558.1 (2004) pp 161-180.

CV2 Analysis

CV2 Analysis evaluates variability of interspike intervals by comparing only adjacent interspike intervals. CV2 statistic is less sensitive to rate variations than other measures such as the coefficient of variation of interspike intervals.

Parameters

Parameter	Description
Time Min (sec)	Only the timestamps in time interval [Time Min, Time Max] will be selected for analysis.
Time Max (sec)	Only the timestamps in time interval [Time Min, Time Max] will be selected for analysis.
Max Mean of ISI Pair (sec)	CV2 graph shows ISI variability versus Mean of ISI Pair. Max Mean of ISI Pair specifies maximum of X axis in CV2 graph
Mean of ISI Pair Bin (sec)	Bin size for calculation of CV2 mean and Standard Error of Mean.
Dot Size (pts)	Specifies the size of the dots shown at (Mean of ISI Pair, CV2) coordinates.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
CV2 Min	Y axis minimum.
CV2 Max	Y axis maximum.
Time Min	Minimum of the time interval that was used to select spikes
Time Max	Minimum of the time interval that was used to select spikes
CV2 Mean	Average of all CV2 values.
Spikes	The number of spikes used in calculation.
Mean Firing Rate	Mean firing rate of the spikes used in analysis.

Algorithm

If the spikes in a train occur at times `t[i] (i = 1, 2, ..., N)` then the ISIs will be

```
isi[i] = t[i] - t[i-1] (i = 2, 3, ..., N)
```

CV2 is then calculated as

```
CV2[i] = 2*abs(isi[i+1] - isi[i]) / (isi[i+1] - isi[i])
```

Dots in CV2 graph are drawn at `(X,Y)` locations `(ISIPairMean[i], CV2[i])` where

```
ISIPairMean[i] = (isi[i+1] - isi[i]) / 2
```

X (ISI pair mean) axis is divided into bins. For each bin, the mean of CV2 values and the Standard Error of Mean of CV2 values are calculated.

Reference

Holt, Gary R., William R. Softky, Christof Koch, and Rodney J. Douglas. "Comparison of discharge variability in vitro and in vivo in cat visual cortex neurons." Journal of Neurophysiology 75, no. 5 (1996): 1806-1814.

Power Spectral Densities

This analysis captures the frequency content neuronal rate histograms.

Parameters

Parameter	Description
Max Frequency	Maximum frequency of the spectrum in Hz
Number of values	Number of values in the spectrum.
Window Overlap (%)	Window overlap in percent (from 0 to 90).
Window Preprocessing	Preprocessing to be done for each window before calculating the spectrum of the window.
Windowing Function	Windowing Function to be applied to each window before calculating the spectrum of the window.

continues on next page

Table 21 – continued from previous page

Parameter	Description
Discard Incomplete Windows	Option to discard the last window if it contains less than the expected number of data points (less than NumberOfFrValues*2 points).
Use Multitaper Algorithm	Option to use multi-taper algorithm.
Time-Bandwidth Product	Time-Bandwidth Product when using multi-taper algorithm.
Number of Tapers	Number of Tapers when using multi-taper algorithm.
Show Frequency From	Minimum frequency to be shown in the spectrum graph.
Show Frequency To	Maximum frequency to be shown in the spectrum graph.
Normalization	Spectrum normalization. See *Algorithm* below.
Log Frequency Scale	An option show Log10 frequency scale.
Smooth	Option to smooth the spectrum after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Overlay Graphs	An option to draw several histograms in each graph. This option requires that *Int. filter type* specifies that multiple interval filters will be used (either *Table (row)* or *Table (col)*).
Overlay Options	Specifies line colors and line styles for overlaid graphs.

continues on next page

Table 21 – continued from previous page

Parameter	Description
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Spectrum minimum.
YMax	Spectrum maximum.
Spikes	The number of spikes used in spectrum calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Frequency of Minimum	The position of the minimum of the displayed spectrum (in Hz). If there are multiple bins in the spectrum where the spectrum value is equal to the spectrum minimum, this value represents the frequency of the first such bin.

continues on next page

Table 22 – continued from previous page

Column	Description
Frequency of Maximum	The position of the maximum of the displayed spectrum (in Hz). If there are multiple bins in the spectrum where the spectrum value is equal to the spectrum maximum, this value represents the frequency of the first such bin.

Algorithm

Spectral Densities for Spike Trains and Events

Spectral Densities are defined only for continuously recorded signals, so series of timestamps (neurons and events) need to be converted to continuously recorded signals to calculate the spectra.

NeuroExplorer uses rate histograms to represent spike trains as continuous signals. Rate histogram parameters are calculated using the following formulas:

```
Bin = 1./(2.*Maximum_Frequency)
NumberOfBins = 2*Number_of_Frequency_Values
```

The rate histogram over the whole analysis time period is split up into data segments (or windows) of length `N` (where `N` is `NumberOfBins`), overlapping by `D` points. For example, if overlap is 50%, then `D` is `N/2`.

For each segment, the signal is preprocessed according to Window Preprocessing parameter. For example, if Subtract Mean is selected, `ProcessedSignal[i] = Signal[i] - meanOfSignalInSegment`.

The overlapping segments are then windowed: after the data is split up into overlapping segments, the individual data segments have a window applied to them (that is, `ProcessedWindowedSignal[i] = ProcessedSignal[i]*WindowValue[i]`; the window is specified by the Windowing Function).

Most window functions afford more influence to the data at the center of the segment than to data at the edges, which represents a loss of information. To mitigate that loss, the individual data segments are commonly overlapped in time (as in the above step).

After doing the above, the periodogram is calculated by computing the discrete Fourier transform, and then computing the squared magnitude of the result. The individual periodograms are then time-averaged, which reduces the variance of the individual power measurements. The end result is an array of power measurements vs. frequency "bin".

For multi-taper spectral estimate, several periodograms (tapers) are calculated for each segment. Each taper is calculated by applying a specially designed windowing function (Slepian

function, see https://en.wikipedia.org/wiki/Multitaper). All the tapers for a given segment are then averaged to form the periodogram of the segment. The individual segment periodograms are then time-averaged.

See http://www.spectraworks.com/Help/mtmtheory.html for a discussion on selecting Multi-Taper parameters.

Normalization

If **Normalization** is Raw PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values is equal to the mean squared value of the rate histogram. The formulas (13.4.10) of Numerical Recipes in C are used (squared fft values are divided by `Nb^2` where `Nb` is the number of values in data window, `NumberofBins` above). (Numerical Recipes in C, Press, Flannery, et al. (Cambridge University Press, 1992))

If **Normalization** is % of Total PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values equals 100.

If **Normalization** is Log of PSD (Numerical Recipes), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum[i])
```

where raw_spectrum is calculated as described above in Raw PSD (Numerical Recipes).

If **Normalization** is Raw PSD (Matlab), the power spectrum is normalized as described in https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html (squared fft values are divided by Fs*Nb, where Fs is sampling rate, Nb is the number of values in window, NumberOfBins defined above).

If **Normalization** is Log of PSD (Matlab), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum_matlab[i])
```

where raw_spectrum_matlab is calculated as described above in Raw PSD (Matlab).

Spectrogram Analysis

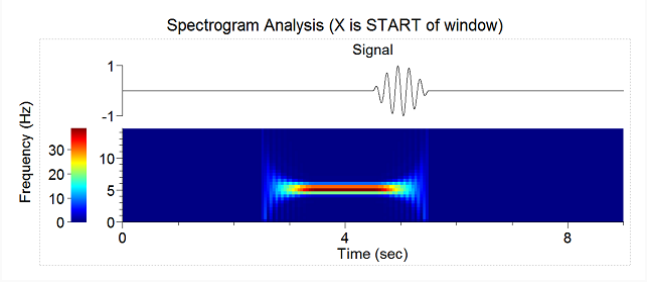
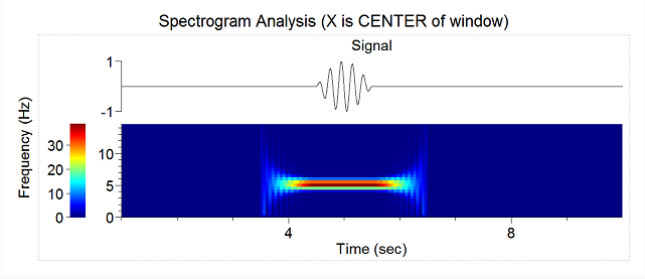
This analysis captures the frequency content of continuous variables or neuronal rate histograms.

Parameters

Parameter	Description
Max. Freq.	Maximum frequency for the spectrograms. If at least one continuous variable is selected, the value of the Maximum Frequency parameter is set as 0.5*Digitizing frequency of the first selected continuous variable. All continuous variables selected for this analysis should have the same digitizing rate.
Number of Fr. Values	Number of frequency values.
Normalization	Spectrum units (see **Normalization** below).
Start	Start of the first window.
Shift Type	Option to specify shift in seconds or as a percent of the sliding window width.
Shift (sec)	If **Shift Type** is Absolute, specifies how much sliding window is shifted each time (in seconds).
Shift (window %)	If **Shift Type** is Window Percent, specifies how much sliding window is shifted each time as a percent of the overall sliding window width.
Number of Shifts	total number of sliding windows.
Window Preprocessing	Preprocessing to be done for each window before calculating the spectrum of the window.
Windowing Function	Windowing Function to be applied to each window before calculating the spectrum of the window.
Use Multitaper Algorithm	Option to use multi-taper algorithm.
Time-Bandwidth Product	Time-Bandwidth Product when using multi-taper algorithm.
Number of Tapers	Number of Tapers when using multi-taper algorithm.
Show Freq. From	An option to show a subset of frequencies. Specifies minimum frequency to be displayed.
Show Freq. To	An option to show a subset of frequencies. Specifies maximum frequency to be displayed.

continues on next page

Table 23 – continued from previous page

Parameter	Description
X Axis	Allows to specify what values are shown in the sliding window X axis. There are two options: Start of Window and Center of Window. Suppose your data has the strongest spectral value at 5 seconds. This means that the window that has 5 seconds as its center will show the highest spectrum values. If the window width is 2 seconds, it will be the window [4,6]. If you are using 'Start of window' option, you will see the peak at 4 instead of 5 seconds. However, with 'Center of window' option, the peak will be at 5 seconds.  
Smooth	Option to smooth the spectrum after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Color Scale Limit (%)	Specifies maximum value for the color scale in percent. Allows you to 'zoom in' small spectrogram values.
Log Vertical Frequency Scale	Option to draw logarithmic vertical frequency scale.
Draw Signal above Spectrogram	Option to draw signal above histogram. The signal is drawn for the time values specified in X Axis. It is recommended to use Center of Window option for X Axis (otherwise, the signal will not be aligned to the spectrogram).
Signal Window Height (%)	The height of the signal window (percent of the total graph window).
Add Freq. Values to Results	An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.

continues on next page

Table 23 – continued from previous page

Parameter	Description
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Y Min	Y axis minimum.
Y Max	Y axis maximum.
Color Scale Min	Color scale minimum.
Color Scale Max	Color scale maximum.
Max Frequency Used	Maximum frequency of the FFTs used.
Rate Histogram Bin	Rate histogram bin size in seconds (if used)
Window width	FFT window width (seconds).
Actual shift	Actual window shift used in analysis.

Algorithm

The power spectrum is calculated for the specified number (**Number of Shifts**) of windows. For each window:

1. For a timestamped variable, the rate histogram is calculated and copied to the Signal array. The following parameters are used:

```
Histogram_Start = Start + Shift * (Window_Number - 1)
Bin = 1./(2.*Maximum_Frequency)
```

(continues on next page)

(continued from previous page)

```
NumberOfBins = 2*Number_of_Frequency_Values
```

2. For a continuous variable,

```
Window_Start = Start + Shift * (Window_Number - 1)
```

`N` values of the continuous variable (starting with the value at Window_Start) are copied to Signal array where `N = 2*Number_of_Frequency_Values`.

3. Signal values are pre-processed according to the specified **Window Preprocessing**.
4. Signal values are multiplied by the coefficients of the specified **Windowing Function**.
5. Discrete FFT of the result is calculated.
6. Power spectrum is calculated from FFT using the formulas defined in (Press et al., Numerical Recipes in C., Cambridge University Press, 1992)

Normalization

If **Normalization** is Raw PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values is equal to the mean squared value of the rate histogram. The formulas (13.4.10) of Numerical Recipes in C are used. (Numerical Recipes in C, Press, Flannery, et al. (Cambridge University Press, 1992))

If **Normalization** is % of Total PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values equals 100.

If **Normalization** is Log of PSD (Numerical Recipes), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum[i])
```

where raw_spectrum is calculated as described above in Raw PSD (Numerical Recipes).

If **Normalization** is Raw PSD (Matlab), the power spectrum is normalized as described in https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html

If **Normalization** is Log of PSD (Matlab), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum_matlab[i])
```

where raw_spectrum_matlab is calculated as described above in Raw PSD (Matlab).

Spike Train Visualizations

Rasters

The raster display shows the spikes as vertical lines (tick) positioned according to the spikes timestamps. For continuous variables, the raster display shows signal versus time graph. This analysis does not generate numerical results since the raster lines are drawn directly from the data.

Parameters

Parameter	Description
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Vert. size	The size of a raster tick (in percent of the graph height).

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum (added for compatibility with other analyses, always 0).
YMax	Y axis maximum (added for compatibility with other analyses, always 1).
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).

Algorithm

The raster display shows the spikes as vertical lines positioned according to the spikes timestamps. For continuous variables, the raster display shows signal versus time graph.

Cumulative Activity Graphs

The cumulative activity display shows a stepwise function which makes a jump at the moment of spike and stays constant between the spikes.

Parameters

Parameter	Description
XMin	Time axis minimum in seconds
XMax	Time axis maximum in seconds

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Spikes	The number of spikes used in calculation
Mean Freq.	Mean firing rate.

Algorithm

The cumulative activity display shows the stepwise function, which makes a jump at the moment of spike and stays constant between the spikes.

Instant Frequency

The instant frequency display shows the instantaneous frequency of the spike train (at the end of each interspike interval, the inverted interspike interval is drawn as a vertical line or as a point).

Parameters

Parameter	Description
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds.
Graph Style	An option to draw vertical lines or points.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Spikes	The number of spikes used in calculation
Mean Freq.	Mean firing rate.

Algorithm

The instant frequency display shows the instantaneous frequency of the spike train.

If vertical lines option is selected, or each spike that occurred at time `t[i]` it draws a vertical line

from point `(t[i], 0.)` to point `(t[i], 1/(t[i] - t[i-1]))`,

where `t[i-1]` is the time of the preceding spike.

In other words, at the end of each interspike interval, the inverted interspike interval is drawn as a vertical line.

If graph style draw points option is selected, or each spike that occurred at time `t[i]` it draws a circle

at `(t[i], 1/(t[i] - t[i-1]))`,

where `t[i-1]` is the time of the preceding spike.

Interspike Intervals vs. Time

The intervals vs. time graph displays interspike intervals against time (at the end of each interspike interval, the point is drawn with the Y coordinate equal to the interspike interval).

Parameters

Parameter	Description
XMin	Time axis minimum in seconds
XMax	Time axis maximum in seconds

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Spikes	The number of spikes used in calculation
Mean Freq.	Mean firing rate.

Algorithm

The intervals vs. time graph displays interspike intervals against time.

For each spike that occurred at time `t[i]` it draws the point with the coordinates

```
(t[i], t[i] - t[i-1]),
```

where `t[i-1]` is the time of the preceding spike.

In other words, at the end of each interspike interval, the point is drawn with the Y coordinate equal to the interspike interval.

Dependencies between Channels

Crosscorrelograms

Crosscorrelogram shows the conditional probability of a spike at time `t0+t` on the condition that there is a reference event at time `t0`.

Compared to the Perievent Histograms analysis (that calculates the same probabilities), Crosscorrelograms analysis allows to use shift-predictors.

Parameters

Parameter	Description
Reference Type	Specifies if the analysis will use a single or multiple reference events.
Reference	Specifies a reference neuron or event (or a group of reference neurons or events).
XMin	Histogram time axis minimum in seconds.
XMax	Histogram time axis maximum in seconds
Bin	Bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability, Spikes/Second or Z-score). See *Algorithm* below.
No Selfcount	An option not to count reference events if the target event is the same as the reference event. Prevents a histogram to have a huge peak at zero when calculating crosscorrelogram versus itself.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Count bins in filter	An option to normalize each bin individually if the interval filter is used. See *Algorithm* section below for detailed discussion.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.

continues on next page

Table 33 – continued from previous page

Parameter	Description
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Overlay Graphs	An option to draw several histograms in each graph. This option requires that *Int. filter type* specifies that multiple interval filters will be used (either *Table (row)* or *Table (col)*).
Overlay Options	Specifies line colors and line styles for overlaid graphs.
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Shift-predictor	An option to use shift-predictor. Shift-predictor requires that an interval filter is used in the analysis. See *Using Shift-Predictor* for details.
Shift times	How many times to calculate shift predictor. See *Using Shift-Predictor* for details.
Subtract predictor	An option to subtract shift-predictor from crosscorrelogram. See *Using Shift-Predictor* for details.
Predictor line style	Specifies the line used to draw shift-predictor (if subtract predictor is not selected).
Draw confidence limits	An option to draw the confidence limits.
Confidence (%)	Confidence level (percent). See *Confidence Limits* for details.
Conf. mean calculation	An option that specifies how the mean firing rate (that is used in the derivation of the confidence limits) is calculated. There are 3 options: *Use data selection*, *Use all file* and *Use pre-ref data*. See *Confidence Limits* for details. Note that *Use pre-ref data* option can only be used for a stimulation type data, i.e. when the distance between any two consecutive reference events is larger than *XMax-XMin*. See *Confidence Limits* for details.
Conf. display	An option to draw confidence limits either as horizontal lines or as a colored background.
Conf. line style	Line style for drawing confidence limits (used when Conf. display is *Lines*).
Conf. background color	Background color for drawing confidence limits (used when Conf. display is *Colored Background*).

continues on next page

Table 33 – continued from previous page

Parameter	Description
Draw mean freq.	An option to draw a horizontal line representing the expected histogram value for a Poisson spike train. See *Confidence Limits* for details.
Mean line style	Line style for drawing mean frequency.
Draw Cusum	An option to draw a cumulative sum graph above the histogram. See *Cumulative Sum Graphs* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Background	An option on how to calculate the histogram background for the peak and through analysis. See *Peak and Trough Statistics* below.
Peak width	Peak width (the number of bins in peak) in the peak and through analysis. See *Peak and Trough Statistics* below.
Left shoulder	Specifies the left shoulder value (in seconds) in the peak and through analysis. See *Peak and Trough Statistics* below.
Right shoulder	Specifies the right shoulder value (in seconds) in the peak and through analysis. See *Peak and Trough Statistics* below.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	Reference variable name.
NumRefEvents	The number of reference events used in calculation.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values.
St. Err. Mean. Hist.	The standard error of mean of the histogram bin values.
Conf. Low	Lower confidence level.
Conf. High	Upper confidence level.
Mean	Expected mean value of the histogram. If Z-score normalization is specified, this value is zero and the expected mean value of the histogram with Counts/Bin normalization is shown in Z-score Mean.
Norm. Factor	Normalization factor. Bin counts are divided by this value. See Normalization in *Algorithm* below.
Z-score mean	Expected mean value of the histogram before Z-score normalization (in other words, confidence mean). See *Confidence Limits* for details .
Mean Before Ref.	Mean of the histogram before the reference event (i.e. for all the bins before zero on time axis).
Bins Before Ref.	The number of bins before the bin that contains zero on time axis.
Zero Bin	The index of the bin that contains zero on time axis.
Background Mean	The mean of the histogram background for the peak and trough analysis. See *Peak and Through Statistics* below.
Background Stdev	The standard deviation of the histogram background for the peak and trough analysis. See *Peak and Trough Statistics* below.
Peak Z-score	Peak Z-score. See *Peak and Trough Statistics* below.
Peak/Mean	Histogram peak value divided by the background mean value.
Peak Position	Peak position (in seconds).
Peak Half Height	The Y value of the half height of the peak, i.e. histogram_background_mean + (peak-mean)/2.

continues on next page

Table 34 – continued from previous page

Column	Description
Peak Width at Half Height	Peak width at the peak half height level (in seconds).
Trough Z-score	Trough Z-score. See *Peak and Trough Statistics* below.
Trough/Mean	Histogram trough value divided by the background mean value.
Trough Position	Trough position (in seconds).
Trough Half Height	The Y value of the half height of the trough, i.e. histogram_background_mean + (trough-mean)/2.
Trough Width at Half Height	Trough width at the trough half height level (in seconds).

Algorithm

Crosscorrelogram shows the conditional probability of a spike at time `t0+t` on the condition that there is a reference event at time `t0`.

The time axis is divided into bins. The first bin is `[XMin, XMin+Bin)`. The second bin is `[XMin+Bin, XMin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin.

Let `ref[i]` be the array of timestamps of the reference event, `ts[i]` be the spike train (each ts is the timestamp).

For each timestamp `ref[k]`:

1) calculate the distances from this event (or spike) to all the spikes in the spike train:

```
d[i] = ts[i] - ref[k]
```

2) for each `i`:

if `d[i]` is inside the first bin, increment the bin counter for the first bin:

```
if d[i] >= XMin and d[i] < XMin + Bin
then bincount[1] = bincount[1] +1
```

if `d[i]` is inside the second bin, increment the bin counter for the second bin:

```
if d[i] >= XMin+Bin and d[i] < XMin + Bin*2
then bincount[2] = bincount[2] +1
```

and so on… .

If **Normalization** is **Counts/Bin**, no further calculations are performed.

If **Normalization** is **Probability**, bin counts are divided by the number of reference events.

Note that the Probability normalization makes sense only for small values of Bin. For Probability normalization to be valid (so that the values of probability are between 0 and 1), there should be no more than one spike in each bin. For example, if the Bin value is large and for each `ref[k]` above there are many `d[i]` values such that `d[i] >= XMin and d[i] < XMin + Bin`, the bin count for the first bin can exceed the number of reference events. Then, the probability value `(bincount[1]/number_of_reference_events)` could be larger than 1.

If **Normalization** is **Spikes/Sec**, bin counts are divided by `NumRefEvents*Bin`, where NumRefEvents is the number of reference events.

If **Normalization** is **Z-score**,

```
bin_value = (bin_count - Confidence_mean)/sqrt(Confidence_mean)
```

where `Confidence_mean` is the expected mean bin count calculated according to **Conf. mean calculation** parameter. Please note that bin counts are assumed to have Poisson distribution. Therefore, the standard deviation is equal to the square root of expected mean and Z-score can be considered to have Normal distribution only for large values (more than 10) of the `Confidence_mean`.

If the option **Count Bins In Filter** is selected, for normalization Spikes/Second, NeuroExplorer will divide bin count by `NumTimesBinWasInFilter*Bin` instead of `NumRefEvents*Bin`. The problem is that when the interval filter is used, bins close to `XMin` and to `XMax` may often (when a reference event is close to the beginning or to the end of the interval in the interval filter) be positioned outside the filter and therefore will not be used for many reference events. Hence, the bins close to 0.0 will be used in analysis more often than the bins close to `XMin` and `XMax`. If the option **Count Bins In Filter** is selected, NeuroExplorer will count the number of times each bin was used in the calculation and use this count, `NumTimesBinWasInFilter`, (instead of the number of reference events) to normalize the histogram.

Peak and Trough Statistics

NeuroExplorer calculates histogram peak statistics the following way:

- Maximum of the histogram is found
- If the histogram contains several maxima with the same value, peak statistics are not calculated
- Otherwise, the center of the bin, where the histogram reaches maximum, is shown as **Peak Position** in the Summary of Numerical results
- The mean `M` and standard deviation `S` of the bin values of the histogram background are calculated:

  - If **Background** parameter is set as *Bins outside peak/trough*, bins outside peak and trough (i.e., bins that are more than PeakWidth/2 away from the bin with the histogram maximum and the bin with the histogram minimum) are used to calculate `M` and `S`
  - If **Background** parameter is set as *Shoulders*, bins that are to the left of the Left Shoulder or to the right of Right Shoulder parameters are used to calculate `M` and `S`

- The value `M` (mean of the background bin values) is shown as **Background Mean** in the Summary of Numerical results
- The value `S` (standard deviation of the background bin values) is shown as **Background Stdev** in the Summary of Numerical results
- The value `(HistogramMaximum - M)/S` is shown as **Peak Z-score**
- The value `(HistogramMaximum + M)/2` is shown as **Peak Half Height**
- Histogram intersects a horizontal line drawn at Peak Half Height at time points `TLeft` and `TRight`. `(TRight - TLeft)` is shown as **Peak Width**

Histogram trough statistics are calculated in a similar way. The only difference is that histogram minimum instead of histogram maximum is analyzed.

Perievent Histograms

Perievent Histogram shows the conditional probability of a spike at time `t0+t` on the condition that there is a reference event at time `t0`. For continuous variables, this analysis calculates event-triggered averages.

Parameters

Parameter	Description
Reference Type	Specifies if the analysis will use a single or multiple reference events.
Reference	Specifies a reference neuron or event (or a group of reference neurons or events).
XMin	Histogram time axis minimum in seconds.
XMax	Histogram time axis maximum in seconds
Bin	Bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability, Spikes/Second or Z-score). See *Algorithm* below.

continues on next page

Table 35 – continued from previous page

Parameter	Description
No Selfcount	An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating Perievent Histogram versus itself).
Ignore Ref. with Missing Data	An option not to use reference events `(t)` that have missing continuous data in time interval `(t+XMin,t+XMax)`. If this option is not specified and continuous values are missing for part of `(t+XMin,t+XMax)`, the missing values are replaced by zero values. This may distort the resulting Perievent Histogram (spike-triggered average).
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Overlay Graphs	An option to draw several histograms in each graph. This option requires that *Int. filter type* specifies that multiple interval filters will be used (either *Table (row)* or *Table (col)*).
Overlay Options	Specifies line colors and line styles for overlaid graphs.
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Draw confidence limits	An option to draw the confidence limits.

continues on next page

Table 35 – continued from previous page

Parameter	Description
Confidence (%)	Confidence level (percent). See *Confidence Limits* for details.
Conf. mean calculation	An option that specifies how the mean firing rate (that is used in the derivation of the confidence limits) is calculated. There are 3 options: *Use data selection*, *Use all file* and *Use pre-ref data*. See *Confidence Limits* for details. Note that *Use pre-ref data* option can only be used for a stimulation type data, i.e. when the distance between any two consecutive reference events is larger than `XMax-XMin`. See *Confidence Limits <ConfidenceLimitsforPerievent>* for details.
Conf. display	An option to draw confidence limits either as horizontal lines or as a colored background.
Conf. line style	Line style for drawing confidence limits (used when Conf. display is *Lines*).
Conf. background color	Background color for drawing confidence limits (used when Conf. display is *Colored Background*).
Draw mean freq.	An option to draw a horizontal line representing the expected histogram value for a Poisson spike train. See *Confidence Limits* for details.
Mean line style	Line style for drawing mean frequency.
Draw Cusum	An option to draw a cumulative sum graph above the histogram. See *Cumulative Sum Graphs* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.
Background	An option on how to calculate the histogram background for the peak and through analysis. See *Peak and Trough Statistics* below.
Peak width	Peak width (the number of bins in peak) in the peak and through analysis. See *Peak and Trough Statistics* below.
Left shoulder	Specifies the left shoulder value (in seconds) in the peak and through analysis. See *Peak and Trough Statistics* below.
Right shoulder	Specifies the right shoulder value (in seconds) in the peak and through analysis. See *Peak and Trough Statistics* below.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options* .
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.

continues on next page

Table 35 – continued from previous page

Parameter	Description
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	Reference variable name.
NumRefEvents	The number of reference events used in calculation.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values.
St. Err. Mean. Hist.	The standard error of mean of the histogram bin values.
Conf. Low	Lower confidence level.
Conf. High	Upper confidence level.
Mean	Expected mean value of the histogram. If Z-score normalization is specified, this value is zero and the expected mean value of the histogram with Counts/Bin normalization is shown in Z-score Mean.
Norm. Factor	Normalization factor. Bin counts are divided by this value. See Normalization in *Algorithm* below.
Z-score mean	Expected mean value of the histogram before Z-score normalization (in other words, confidence mean). See *Confidence Limits* for details .
Mean Before Ref.	Mean of the histogram before the reference event (i.e. for all the bins before zero on time axis).

continues on next page

Table 36 – continued from previous page

Column	Description
Bins Before Ref.	The number of bins before the bin that contains zero on time axis.
Zero Bin	The index of the bin that contains zero on time axis.
Background Mean	The mean of the histogram background for the peak and trough analysis. See *Peak and Through Statistics* below.
Background Stdev	The standard deviation of the histogram background for the peak and trough analysis. See *Peak and Trough Statistics* below.
Peak Z-score	Peak Z-score. See *Peak and Trough Statistics* below.
Peak/Mean	Histogram peak value divided by the background mean value.
Peak Position	Peak position (in seconds).
Peak Half Height	The Y value of the half height of the peak, i.e. `histogram_background_mean + (peak-mean)/2`.
Peak Width at Half Height	Peak width at the peak half height level (in seconds).
Trough Z-score	Trough Z-score. See *Peak and Trough Statistics* below.
Trough/Mean	Histogram trough value divided by the background mean value.
Trough Position	Trough position (in seconds).
Trough Half Height	The Y value of the half height of the trough, i.e. `histogram_background_mean + (trough-mean)/2`.
Trough Width at Half Height	Trough width at the trough half height level (in seconds).

Algorithm

Neurons and Events

Perievent Histogram shows the conditional probability of a spike at time `t0+t` on the condition that there is a reference event at time `t0`.

The time axis is divided into bins. The first bin is `[XMin, XMin+Bin)`. The second bin is `[XMin+Bin, XMin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin.

Let `ref[i]` be the array of timestamps of the reference event, `ts[i]` be the spike train (each ts is the timestamp).

For each timestamp `ref[k]`:

1) calculate the distances from this event (or spike) to all the spikes in the spike train:

```
d[i] = ts[i] - ref[k]
```

2) for each `i`:

if `d[i]` is inside the first bin, increment the bin counter for the first bin:

```
if d[i] >= XMin and d[i] < XMin + Bin
then bincount[1] = bincount[1] +1
```

if `d[i]` is inside the second bin, increment the bin counter for the second bin:

```
if d[i] >= XMin+Bin and d[i] < XMin + Bin*2
then bincount[2] = bincount[2] +1
```

and so on... .

If **Normalization** is **Counts/Bin**, no further calculations are performed.

If **Normalization** is **Probability**, bin counts are divided by the number of reference events.

Note that the Probability normalization makes sense only for small values of Bin. For Probability normalization to be valid (so that the values of probability are between 0 and 1), there should be no more than one spike in each bin. For example, if the Bin value is large and for each `ref[k]` above there are many `d[i]` values such that `d[i] >= XMin and d[i] < XMin + Bin`, the bin count for the first bin can exceed the number of reference events. Then, the probability value `(bincount[1]/number_of_reference_events)` could be larger than 1.

If **Normalization** is **Spikes/Sec**, bin counts are divided by `NumRefEvents*Bin`, where NumRefEvents is the number of reference events.

If **Normalization** is **Z-score**, `bin_value = (bin_count - Confidence_mean)/sqrt(Confidence_mean)`, where `Confidence_mean` is the expected mean bin count calculated according to **Conf. mean calculation** parameter. Please note that bin counts are assumed to have Poisson distribution. Therefore, the standard deviation is equal to the square root of expected mean and Z-score can be considered to have Normal distribution only for large values (more than 10) of the `Confidence_mean`.

Continuous Variables

For a continuous variable, the mean of the variable values is calculated for each bin around reference event. These mean values are then averaged across all the timestamps of the reference event.

Peak and Trough Statistics

NeuroExplorer calculates histogram peak statistics the following way:

- Maximum of the histogram is found
- If the histogram contains several maxima with the same value, peak statistics are not calculated
- Otherwise, the center of the bin, where the histogram reaches maximum, is shown as **Peak Position** in the Summary of Numerical results
- The mean `M` and standard deviation `S` of the bin values of the histogram background are calculated:
  - If **Background** parameter is set as *Bins outside peak/trough*, bins outside peak and trough (i.e., bins that are more than PeakWidth/2 away from the bin with the histogram maximum and the bin with the histogram minimum) are used to calculate `M` and `S`
  - If **Background** parameter is set as *Shoulders*, bins that are to the left of the Left Shoulder or to the right of Right Shoulder parameters are used to calculate `M` and `S`
- The value `M` (mean of the background bin values) is shown as **Background Mean** in the Summary of Numerical results
- The value `S` (standard deviation of the background bin values) is shown as **Background Stdev** in the Summary of Numerical results
- The value `(HistogramMaximum - M)/S` is shown as **Peak Z-score**
- The value `(HistogramMaximum + M)/2` is shown as **Peak Half Height**
- Histogram intersects a horizontal line drawn at Peak Half Height at time points `TLeft` and `TRight`. `(TRight - TLeft)` is shown as **Peak Width**

Histogram trough statistics are calculated in a similar way. The only difference is that histogram minimum instead of histogram maximum is analyzed.

PSTH Versus Time

This analysis shows the dynamics of the perievent histogram over time. It calculates multiple PST histograms using a "sliding window" in time. Each histogram is shown as a vertical stripe with colors representing the bin counts. Horizontal axis represents the position of the sliding window in time.

Parameters

Parameter	Description
Reference	Specifies a reference neuron or event.
XMinPSTH	Histogram time minimum in seconds.
XMaxPSTH	Histogram time maximum in seconds
BinPSTH	Histogram bin size in seconds.
Start	Start of the first sliding window in seconds.
Duration	Duration of the sliding window in seconds.
Shift	How much sliding window is shifted each time.
Number of shifts	The number of sliding windows to be used.
Normalization	Histogram units (Counts/Bin, Probability or Spikes/Second). See *Algorithm* below.
No Selfcount	An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).
Sliding window axis direction	Direction of the sliding window axis (horizontal or vertical).
Sliding window axis alignment	Allows to specify what values are shown in the sliding window axis. There are two options: Start of Window and Center of Window. Suppose your data has the strongest correlation at 50 seconds. This means that the window that has 50 seconds as its center will show the highest PST values. If the window width is 20 seconds, it will be the window [40,60]. If you are using 'Start of window' option, you will see the peak at 40 instead of 50 seconds. However, with 'Center of window' option, the peak will be at 50 seconds.
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.

continues on next page

Table 37 – continued from previous page

Parameter	Description
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	Reference variable name.
YMin	PSTH minimum.
YMax	PSTH maximum.
Color Scale Min	PSTH minimum.
Color Scale Max	PSTH maximum.
Spikes	The number of spikes in the variable

Algorithm

NeuroExplorer calculates multiple PST histograms (see *Perievent Histogram* for more information on how each PSTH is calculated). Each histogram is calculated for a different interval (window) in time:

```
[window_start[i], window_end[i]], i = 1,...,Number_of_Shifts
```

That is, for each histogram only the timestamps that are within the window are used.

The following rules are used to calculate the windows:

```
window_start[1] = Start

window_end[1] = Start + Duration

window_start[2] = Start + Shift

window_end[2] = Start + Shift + Duration

...
```

Trial Bin Counts

Trial bin counts analysis computationally is essentially the same as the perievent histogram. The difference is that the bin counts are saved for each reference event. When used with continuous data, for each bin and reference vent, the average of the continuous values within the bin is calculated.

Parameters

Parameter	Description
Reference	Specifies a reference neuron or event.
XMin	Histogram time axis minimum in seconds.
XMax	Histogram time axis maximum in seconds
Bin	Bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability or Spikes/Second). See *Algorithm* below.
No Selfcount	An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Color Scale Min	Color scale minimum.
Color Scale Max	Color scale maximum.
Reference	Reverence event.
NumRefEvents	Number of reference events.
Bin001Mean	Mean of all the values for Bin 1.
Bin001SdDev	Standard deviation of all the values for Bin 1

Algorithm

Trial bin counts analysis computationally is essentially the same as the perievent histogram. The difference is that the bin counts are saved for each reference event. When used with continuous data, for each bin and reference vent, the average of the continuous values within the bin is calculated.

The time axis is divided into bins. The first bin is `[XMin, XMin+Bin)`. The second bin is `[XMin+Bin, XMin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin.

Let `ref[i]` be the array of timestamps of the reference event, `ts[i]` be the spike train (each ts is the timestamp).

For each timestamp `ref[k]`:

1) calculate the distances from this event (or spike) to all the spikes in the spike train:

```
d[i] = ts[i] - ref[k]
```

2) for each `i`:

if `d[i]` is inside the first bin, increment the bin counter for the first bin:

```
if d[i] >= XMin and d[i] < XMin + Bin
then bincount[1] = bincount[1] +1
```

if `d[i]` is inside the second bin, increment the bin counter for the second bin:

```
if d[i] >= XMin+Bin and d[i] < XMin + Bin*2
then bincount[2] = bincount[2] +1
```

and so on… .

If **Normalization** is **Counts/Bin**, no further calculations are performed.

If **Normalization** is **Spikes/Sec**, bin counts are divided by **Bin.**

Perievent Rasters

This analysis shows the timestamps of the selected variable relative to the timestamps of the reference variable.

Parameters

Parameter	Description
Reference Type	Specifies if the analysis will use a single or multiple reference events.
Reference	Specifies a reference neuron or event (or a group of reference neurons or events).
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Markers	An option to display additional events (as markers - triangles, squares, etc.) in the perievent raster.
Background Intervals	An option to draw intervals in the raster background.
Trial Order	This option allows you to specify how the raster lines are drawn. The raster lines (corresponding to reference events) can be drawn in the order of reference events (with the first event on top, the last at the bottom of the perievent raster), or in the reverse order (the first event at the bottom of the raster display, the last event at the top). This option is ignored if *Sort trials* parameter is not *None*.
Sort trials	An option to sort raster lines. By default, the raster lines (corresponding to reference events) are drawn in the order specified by the Trial Order parameter. This option allows you to display reference events in a different order. For example, in behavioral experiments, you can sort the trials according to the latency of the response.
Sort event	Specifies how to identify the timestamps (in Sort ref. variable) that will be used in sorting.

continues on next page

Table 41 – continued from previous page

Parameter	Description
Sort ref.	Specifies event variable that will be used to sort trials.
Histogram	An option to draw a histogram.
Bin	Histogram bin size in seconds.
Normalization	Histogram units (Counts/Bin, Probability or Spikes/Second). See *Algorithm* below.
No Selfcount	An option not to count reference events if the target event is the same as the reference event. Prevents a histogram to have a huge peak at zero when calculating PSTH versus itself.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Smooth histogram	An option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	Reference variable name.
NumRefEvents	The number of reference events used in calculation.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values.
St. Err. Mean. Hist.	The standard error of mean of the histogram bin values.

Algorithm

Spike Trains

For the perievent histogram calculation,

Let `ref[i]` be the array of timestamps of the reference event, `sortref[i]` be the array of timestamps of the sort event.

If **Sort Trials** is selected, the following algorithm is used:

1) for each timestamp `ref[k]`, NeuroExplorer finds the smallest `sortref[i]`, such that

```
sortref[i] > ref[k]
```

2) the distance between two events is calculated:

```
dist[k] = sortref[i] - ref[k]
```

3) the trials `ref[k]` are sorted array sorted in ascending or descending order of `dist[k]`.

Continuous Variables

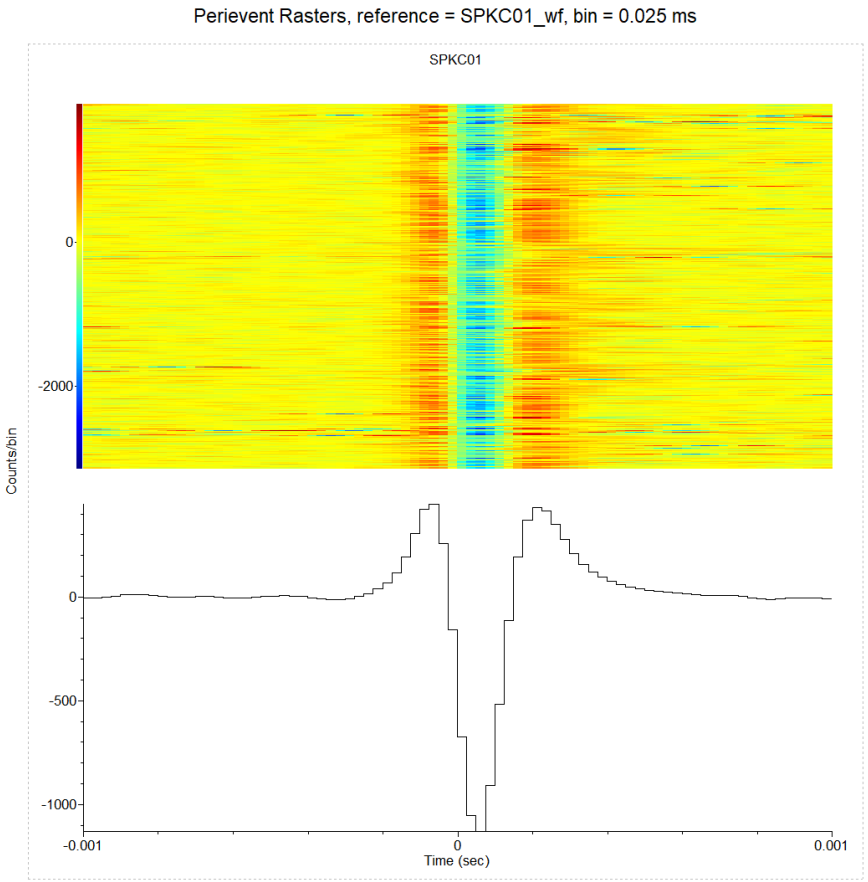
Continuous variables can also be used in this analysis. However, you may prefer to use **PeriEvent Rasters for Continuous Variables** analysis.

For each `ref[k]`, NeuroExplorer calculates a series of bins. The first bin is:

`[ref[k]+XMin, ref[k]+XMin+Bin]`

the second bin is `[ref[k]+XMin+Bin, ref[k]+XMin+Bin+Bin]`, etc.

Then, the **average value** of a continuous variable is calculated within each of the bins and this average value is displayed using the color scale. If bin does not contain any timestamps of the continuous variable, the previous value of the continuous variable is used.



Perievent Firing Rates

This analysis is similar to Perievent Histograms, but instead of counting numbers of spikes in bins, this analysis uses method based on convolution of a spike train with a fixed kernel function.

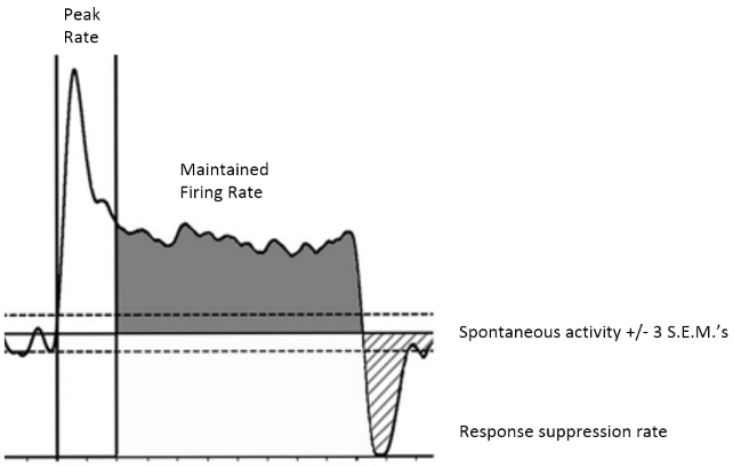


Figure provided by Dr. Maureen McCall, University of Louisville School of Medicine.

Parameters

Parameter	Description
Reference	Specifies a reference neuron or event.
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Kernel Type	The type of kernel to be used (Boxcar, Triangle, Gaussian or Exponential)
Kernel Width (sec)	Kernel width in seconds
Draw Variation	An option to draw variation of firing across reference events as a colored background.
Variation (st. err. mean)	Number of standard errors of mean in the colored background around firing rate mean.
Variation Background Color	The color to be used for drawing of the colored background around firing rate mean.
Draw Mean and Standard Error of Mean	An option to draw expected mean firing rate and plus/minus Variation * S.E.M.
Calculate Response Statistics	An option to calculate response statistics.
Spontaneous Activity Interval Filter	The interval variable used to calculate spontaneous activity mean and S.E.M.

continues on next page

Table 43 – continued from previous page

Parameter	Description
Peak Time Frame Start (seconds)	The start of the time interval (relative to reference) used to calculate peak firing rate.
Peak Time Frame End (seconds)	The end of the time interval (relative to reference) used to calculate peak firing rate.
Maintained Firing Rate Time Frame Start (seconds)	The start of the time interval (relative to reference) used to calculate maintained firing rate.
Maintained Firing Rate Time Frame End (seconds)	The end of the time interval (relative to reference) used to calculate maintained firing rate.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
XMin	Time axis minimum.
XMax	Time axis maximum.
YMin	Y axis minimum.

continues on next page

Table 44 – continued from previous page

Column	Description
YMax	Y axis maximum.
FiringRateMin	Firing rate minimum.
FiringRateMax	Firing rate maximum.
NumRefEvents	The number of reference events used in calculation.
NumSpikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
ResponsePresent	1 if firing rate exceeded `SpontaneuousMean + Variation*S.E.M.`
PeakRate	Peak firing rate (max. of firing rate in Peak Time Frame).
PeakPosition	Peak firing rate position in seconds.
MaintainedRate	Mean firing rate in Maintained Time Frame.
ResponseDuration	Response duration in seconds.
SupressionStart	Suppression start time in seconds.
SupressionEnd	Suppression end time in seconds.

Algorithm

The firing rate is estimated using convolution of a spike train with a fixed kernel function:

`FiringRate(t) = sum of K(t-t[i])`, where `K(t)` is a kernel function, `t[i]` are spike occurrence times.

The following kernel functions are used

```
Boxcar(t) = 1/(2*sqrt(3)*w), t in interval [-sqrt(3)*w, sqrt(3)*w]

Triangle(t) = (sqrt(6)*w - abs(t))/(6*w*w), t in interval [-sqrt(6)*w,␣
↪sqrt(6)*w]

Gaussian(t) = exp(-t*t/(2*w*w))/(sqrt(2*PI)*w), t in interval [-5*w, 5*w]

Exponential(t) = exp(-sqrt(2)*abs(t)/w/(sqrt(2)*w), t in interval [-5*w, 5*w]
```

`w` is a kernel width parameter (square root of the integral of `t square` multiplied by kernel equals `w`). It is specified as the **Kernel Width** analysis parameter.

For each reference event at time `Ref[k]`, `FiringRate(t)` is calculated for the spikes within time interval

```
[Ref[k]+\ **XMin**, Ref[k]+\ **XMax**]
```

The average firing rate across all reference event is then calculated and displayed.

To calculate response statistics, spontaneous mean and standard error of mean (S.E.M) are calculated:

For each interval in interval variable (specified in Spontaneous Activity Interval Filter), the mean firing rate is calculated. Spontaneous mean is the average of mean rates over all the intervals. S.E.M. is the standard error of mean over all the intervals.

If perievent firing rate exceeded `SpontaneuousMean + Variation*S.E.M.`, ResponsePresent is set to to 1.

ResponseDuration is the time when perievent firing rate drops below SpontaneuousMean.

SupressionStart is the time when perievent firing rate drops below `SpontaneuousMean - Variation*S.E.M.`

SupressionEnd is the time when perievent firing rate returns to SpontaneuousMean.

SuppressionRate is the minimum firing rate in the interval `[SupressionStart, SupressionEnd]`

References

Nawrot, Martin, Ad Aertsen, and Stefan Rotter. Single-trial estimation of neuronal firing rates: from single-neuron spike trains to population activity. Journal of neuroscience methods 94.1 (1999): 81-92.

Shigeru Shinomoto. Estimating the Firing Rate. Analysis of Parallel Spike Trains. Springer Series in Computational Neuroscience, Volume 7, 2010, pp 21-35

Joint PSTH

Joint PSTH matrix shows the correlations of the two neurons around the reference events.

Parameters

Parameter	Description
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Bin	Bin size in seconds.
Reference	Specifies a reference neuron or event.
Bottom Neuron	The neuron of event shown along the horizontal axis (the vertical axis shows a neuron selected for analysis).

continues on next page

Table 45 – continued from previous page

Parameter	Description
Diag	The width (in seconds) of the area in the scatter matrix around its main diagonal used to calculate the main diagonal histogram.
Normalization	Scatter matrix normalization.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Smooth	Option to smooth the histograms (not the scatter matrix) after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

continues on next page

Table 45 – continued from previous page

Parameter	Description
Matrix Scale	An option on how to draw the scatter matrix (color or black and white).
Font size (% matrix)	Font size in percent of scatter matrix height.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum (added for compatibility with other analyses, always 0).
YMax	Y axis maximum (added for compatibility with other analyses, always 1).
Spikes	The number of spikes of the neuron shown on the vertical axis used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Hist. norm.	Normalization factor. Histogram bin counts are divided by this value.
Color scale minimum	Scatter matrix color scale minimum.
Color scale maximum	Scatter matrix color scale maximum.

Algorithm

The main Joint PSTH matrix shows the correlations of the bin counts of two neurons around the reference events. Histograms to the left of and below the matrix are standard perievent histograms for two specified neurons. The first histogram to the right of the matrix shows the correlations of near-coincident spikes around the reference events. The far-right histogram shows correlations of firings of two neurons around reference events.

Reference

See the following paper for details on Joint PSTH analysis:

M. H. J. Aertsen, G. L. Gerstein, M. K. Habib and G. Palm. Dynamics of Neuronal Firing Correlation: Modulation of "Effective Connectivity" J. Neurophysiol., Vol. 61, pp. 900-917, 1989.

Epoch Counts

This analysis is very similar to Perievent Histograms. The only distinction is that, instead of calculating bin counts for consecutive bins of the same size, Epoch Counts analysis can calculate bin counts for bins (epochs) of any size. Epochs can be of different lengths and can overlap.

Parameters

Parameter	Description
Reference Type	Specifies if the analysis will use a single or multiple reference events.
Reference	Specifies a reference neuron or event (or a group of reference neurons or events).
Epochs	A table of epochs in seconds.
No Selfcount	An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each epoch) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each epoch) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.

continues on next page

Table 47 – continued from previous page

Parameter	Description
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	The name of the reference event.
NumRefEvents	The number of reference events.
YMin	Minimum epoch count for this variable
YMax	Maximum epoch count for this variable
Filter Length	The length of the interval filter.

Algorithm

Let `ref[i]` be the array of timestamps of the reference event, `ts[i]` be the spike train (each ts is the timestamp) and epochs are specified as `EpochStart[j], EpochEnd[j]`.

For each timestamp `ref[k]`:

1) calculate the distances from this event (or spike) to all the spikes in the spike train:

```
d[i] = ts[i] - ref[k]
```

2) for each `i`:

if `d[i]` is inside the first epoch, increment the counter for the first epoch:

```
if d[i] >= EpochStart[1] and d[i] < EpochEnd[1]
then epochcount[1] = epochcount[1] +1
```

if `d[i]` is inside the second epoch, increment the counter for the second epoch:

```
if d[i] >= EpochStart[2] and d[i] < EpochEnd[2]
then epochcount[2] = epochcount[2] +1
```

and so on... .

Correlations With Continuous Variable

This analysis calculates crosscorrelations between a continuously recorded variable and a neuronal firing rate, or crosscorrelations between a continuously recorded variable and another continuous variable.

Parameters

Parameter	Description
Reference	Specifies a reference continuous variable.
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds.
Bin Type	An option to specify how the bin value is determined. If this option is *Auto*, NeuroExplorer will use the bin equal to the A/D sampling interval of the reference variable.
Bin	Bin size in seconds. Used if *Bin Type* is set to *User defined*.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Smooth	Option to smooth the histogram after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.

continues on next page

Table 49 – continued from previous page

Parameter	Description
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	Reference variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Mean Hist.	Mean of all the correlation values for this variable.
St. Dev. Hist	Standard deviation of all the correlation values for this variable.
St. Err. Mean Hist	Standard error of mean of all the correlation values for this variable

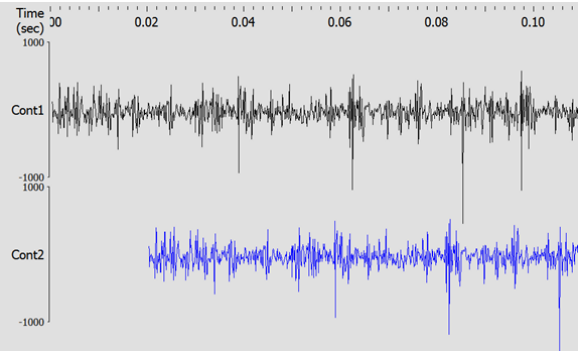
Algorithm

This analysis calculates standard correlograms for two (continuously recorded) vectors. For spike trains and events, NeuroExplorer first calculates the rate histograms with the specified bin size, and then calculates the correlations between the reference variable and the rate histogram.

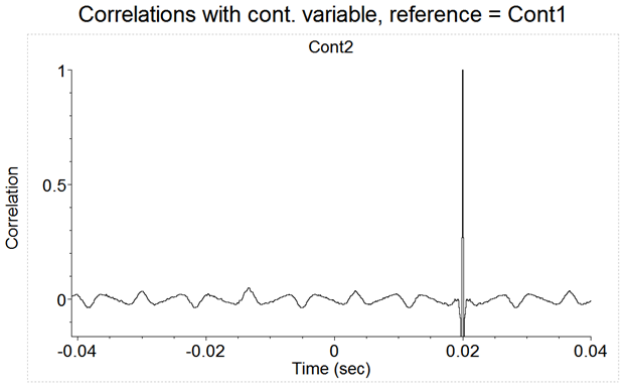
If `x[i], i=1,...N` is the vector of all the values of the reference variable with sampling rate `SR` and `y[i], i=1,...,N` is the vector of all the values of another continuous variable or values of spike train rate histogram, then

```
correlation[lag_n_seconds] = Pearson correlation between vectors

{ x[1], x[2], ..., x[N-lag_in_samples] }

and

{ y[1+lag_in_samples], y[2+lag_in_samples], ..., y[N] }

where lag_in_samples = lag_in_seconds*SR.
```

Let's consider the following example. We have continuous variable Cont1. We create continuous variable Cont2 by running shift operation on Cont1 with shift=0.02s.



For each value of time t, we have `Cont1[t] = Cont2[t+0.02]`. This means that if reference is Cont1 and `lag_in_seconds = 0.02s`, we will calculate Pearson correlation of identical vectors. That is, our correlation result will have the highest value close to 1.0 at 0.02 seconds.



Coherence Analysis

Coherence is a measure of the degree of relationship, as a function of frequency, between two time series.

Parameters

Parameter	Description
Reference	Specifies a reference variable.
Max. Freq.	Frequency maximum (Hz).
Number of Fr. Values	Number of frequency values.
Window Overlap	Percent of window overlaps when calculating FFTs.
Window Preprocessing	Preprocessing to be done for each window before calculating the spectrum of the window.
Windowing Function	Windowing Function to be applied to each window before calculating the spectrum of the window.
Discard Incomplete Windows	Option to discard the last window if it contains less than the expected number of data points (less than `NumberOfFrValues*2` points).
Use Multitaper Algorithm	Option to use multi-taper algorithm.
Time-Bandwidth Product	Time-Bandwidth Product when using multi-taper algorithm.
Number of Tapers	Number of Tapers when using multi-taper algorithm.
Show Freq. From	An option to show a subset of frequencies. Specifies minimum frequency to be displayed.
Show Freq. To	An option to show a subset of frequencies. Specifies maximum frequency to be displayed.
Calculate	Specifies whether coherence values or coherence phases are calculated.
Confidence Level (%)	Confidence level (percent) for the coherence values. Can only be calculated if a single-taper algorithm is used. See Confidence Level Calculation below for details.
Draw confidence level	An option to draw the confidence level. Confidence level is not drawn if coherence phases are specified in Calculate parameter or if a multi-taper algorithm is used.
Conf. line style	Line style for drawing confidence level.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.

continues on next page

Table 51 – continued from previous page

Parameter	Description
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a center of each frequency bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each frequency bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Y Min	Y axis minimum.
Y Max	Y axis maximum.
Spikes	The number of spikes used (for neurons or events).
Rate Hist Bin	Bin size of the rate histograms used for the calculations (for neurons or events)
Filter Length	The length of the interval filter.
Mean Freq.	Mean firing rate (for neurons or events).
Confidence Level	Confidence level for the coherence values.

Algorithm

Neurons and Events

Coherence is defined only for continuously recorded signals, so series of timestamps (neurons and events) need to be converted to continuously recorded signals to calculate coherence.

NeuroExplorer uses rate histograms to represent spike trains as continuous signals. Rate histogram parameters are calculated using the following formulas:

`Bin = 1./(2.*Maximum_Frequency)`

`NumberOfBins = 2\*Number_of_Frequency_Values`

The rate histogram over the whole analysis time period is split up into data segments (or windows) of length `N` (where `N` is NumberOfBins), overlapping by `D` points. If overlap is 50%, then `D` is `N/2`.

For each segment, the signal is preprocessed according to Window Preprocessing parameter. For example, if Subtract Mean is selected, `ProcessedSignal[i] = Signal[i] - meanOfSignalInSegment`.

The overlapping segments are then windowed: after the data is split up into overlapping segments, the individual data segments have a window applied to them (that is, `ProcessedWindowedSignal[i] = ProcessedSignal[i]*WindowValue[i]`; the window is specified by the Windowing Function).

Most window functions afford more influence to the data at the center of the segment than to data at the edges, which represents a loss of information. To mitigate that loss, the individual data segments are commonly overlapped in time (as in the above step).

For two variables `X` (reference) and `Y` (target) the following entities are calculated. FFTs of the data segments (after preprocessing) are calculated. Then, individual and cross-densities are calculated:

```
Pxx = FFT(X)*Conj(FFT(X)),

Pyy = FFT(Y)*Conj(FFT(Y))

Pxy = FFT(X)*Conj(FFT(Y)).
```

Here `Conj(z)` is a complex conjugate of `z`. `Pxx`, `Pyy` and `Pxy` values are averaged across all intervals and coherence values are calculated as

```
Mean(Pxy)*Mean(Pxy) / (Mean(Pxx)*Mean(Pyy)).
```

Coherence phase values are calculated as phase of `Mean(Pxy)`.

Continuous Variables

For continuous variables, if both the reference and the target variables have the same digitizing frequency, the FFTs of variable values (after applying preprocessing and tapering window) are calculated and then resampled to the specified frequency steps. If the reference is a timestamped variable or two continuous variables have different digitizing rates, the values of continuous variables are averaged within the specified bins and then FFTs of these averages are calculated.

Calculation of the Confidence Level

Confidence levels can only be calculated if a single-taper algorithm is used.

Confidence levels are calculated as described in Kattla and Lowery (2010) (see Reference blow). The confidence level `Z` is calculated as

```
Z = 1 - pow(alpha, 1/(w*L - 1))
```

where

`pow(x,y)` returns `x` to the power of `y`,

```
alpha = 1 - Confidence_Level*0.01,
```

`L` is the number of overlapped windows

`w` is the correction due to the Hanning window tapering (see eq. (5) and (6) in Kattla and Lowery (2010)).

Reference

Kattla S, Lowery MM. Fatigue related changes in electromyographic coherence between synergistic hand muscles. Exp Brain Res. 2010 Apr; 202(1):89-99. Epub 2009 Dec 12.

Perievent Spectrograms

This analysis captures the frequency content of continuous variables or neuronal rate histograms in the time windows around reference events.

The idea of perievent spectrogram analysis is that you may have stimuli applied to your neural network and the neurons or LFPs exhibit slight variations of their spectral properties in some periods right after the stimuli. You may not see the variation when analyzing the response to the single stimulus, but when you average the spectrograms over all the stimuli, you may get a clear picture.

Parameters

Parameter	Description
Reference	Reference event.
Skip ref. events with missing cont. data	An option to skip reference events when calculating PeriEvent Spectrograms for continuous variables. Use this option when continuous variables have gaps (time periods without data points). See **Algorithm** below for details.
Max. Freq.	Maximum frequency for the spectrograms. If at least one continuous variable is selected, the value of the Maximum Frequency parameter is set as 0.5*Digitizing frequency of the first selected continuous variable. All continuous variables selected for this analysis should have the same digitizing rate.
Number of Fr. Values	Number of frequency values.
Normalization	Spectrum units (see **Normalization** in Spectrogram Analysis).
Start	Start of the first window (relative to the reference event).
Shift Type	Option to specify shift in seconds or as a percent of the sliding window width.
Shift (sec)	If **Shift Type** is Absolute, specifies how much sliding window is shifted each time (in seconds).
Shift (window %)	If **Shift Type** is Window Percent, specifies how much sliding window is shifted each time as a percent of the overall sliding window width.
Number of Shifts	total number of sliding windows.
Show Freq. From	An option to show a subset of frequencies. Specifies minimum frequency to be displayed.
Show Freq. To	An option to show a subset of frequencies. Specifies maximum frequency to be displayed.

continues on next page

Table 53 – continued from previous page

Parameter	Description
X Axis	Allows to specify what values are shown in the sliding window X axis. There are two options: Start of Window and Center of Window. Suppose your data has the strongest spectral value at 5 seconds. This means that the window that has 5 seconds as its center will show the highest spectrum values. If the window width is 2 seconds, it will be the window [4,6]. If you are using 'Start of window' option, you will see the peak at 4 instead of 5 seconds. However, with 'Center of window' option, the peak will be at 5 seconds.
Log Vertical Frequency Scale	Option to draw logarithmic vertical frequency scale.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options* .
Smooth	Option to smooth the spectrum after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a center of each frequency bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each frequency bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options* .
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Y Min	Y axis minimum.
Y Max	Y axis maximum.
Color Scale Min	Color scale minimum.
Color Scale Max	Color scale maximum.
Max Frequency Used	Maximum frequency of the FFTs used.
Rate Histogram Bin	Rate histogram bin size in seconds (if used).
Number of Ref. Events	Number of reference events.
Window width	FFT window width (seconds).
Actual shift	Actual window shift used in analysis.
Number of Bins In Shift	Number of rate histogram bins in shift (if rate histograms were used)

Algorithm

Perievent spectrogram is an average of multiple regular spectrograms. For each timestamp of the reference event, a regular spectrogram is calculated, then these spectrograms are averaged over all the selected reference timestamps.

- For a continuous variable, for each timestamp of the reference event, the following parameters are used:

  ```
  Window_Start = Reference_Timestamp + Start + Shift * (Window_Number
  - 1)
  ```

  `N` values of the continuous variable (starting with the value at `Window_Start`) are copied to Signal array where `N = 2*Number_of_Frequency_Values`.

- Signal values are pre-processed according to the specified **Window Preprocessing**.
- Signal values are multiplied by the coefficients of the specified **Windowing Function**.
- Discrete FFT of the result is calculated.
- Power spectrum is calculated from FFT using the formulas defined in (Press et al., Numerical Recipes in C., Cambridge University Press, 1992)

If **Skip ref. events with missing cont. data** option is selected, and there is no continuous data for any part of time interval `[RefTimestamp+StartTime, RefTimestamp+StartTime+Shift*NumberOfShifts]`, the spectrogram for `RefTimestamp` is not calculated.

Regularity Analysis

This analysis estimates the regularity of neuronal firing after the stimulation.

Parameters

Parameter	Description
Reference	Specifies a reference event or spike train.
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds.
Bin	Bin size in seconds. Used if *Bin Type* is set to *User defined*.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
ISI Graph	This parameter determines how the mean ISI values is displayed in the graph.
SD Graph	This parameter determines how the standard deviation of ISI is displayed in the graph.
CV Graph	This parameter determines how the CV (coefficient of variation) values are displayed in the graph.
CV Graph Max	This parameter determines the scale for the CV values in the graph. CV values are scaled from **zero** (bottom of the graph) to **CV Graph Max** (top of the graph).
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
NumRefEvents	Number of reference events.
YMin	Histogram minimum.
YMax	Histogram maximum.
Spikes	The number of spikes used in calculation.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Freq.	Mean firing rate (Spikes/Filter_Length).
Mean Hist.	The mean of the histogram bin values (ISI values).
St. Dev. Hist.	The standard deviation of the histogram bin values.
St. Dev. ISI	The mean value of the SD ISI graph.
CV	The mean value of the CV ISI graph.

Algorithm

This analysis estimates the regularity of neuronal firing after the stimulation. The time axis is divided into bins with the first bin starting at the stimulus sync pulse. Interspike intervals are computed and interval values are placed in time bins according to the latency of the first spike in the interval (if the end of the ISI is more than XMax, the interval is not used in the analysis). Then the mean and standard deviation in each bin are calculated. CV is equal to standard deviation for the bin divided by the mean ISI for the bin.

Let `ref[i]` be the array of timestamps of the reference event,

`ts[i]` be the spike train (each ts is the timestamp)

For each timestamp `ref[k]`:

1a) calculate the distances from this event (or spike) to all the spikes in the spike train:

```
d[i] = ts[i] - ref[k]
```

1b) calculate the interspike interval

```
isi[i] = ts[i+1] - ts[i]
```

1c) for each i:

if `d[i]` is inside the first bin (`d[i] >= XMin and d[i] < XMin + Bin`) and `d[i] + isi[i] < XMax`, add `isi[i]` to the series of intervals for the first bin.

if `d[i]` is inside the second bin (`d[i] >= XMin+Bin and d[i] < XMin + Bin*2`) and `d[i] + isi[i] < XMax`, add `isi[i]` to the series of intervals for the second bin

and so on... .

2) for each bin, calculate mean and standard deviation of the series of interspike intervals for this bin.

Reference

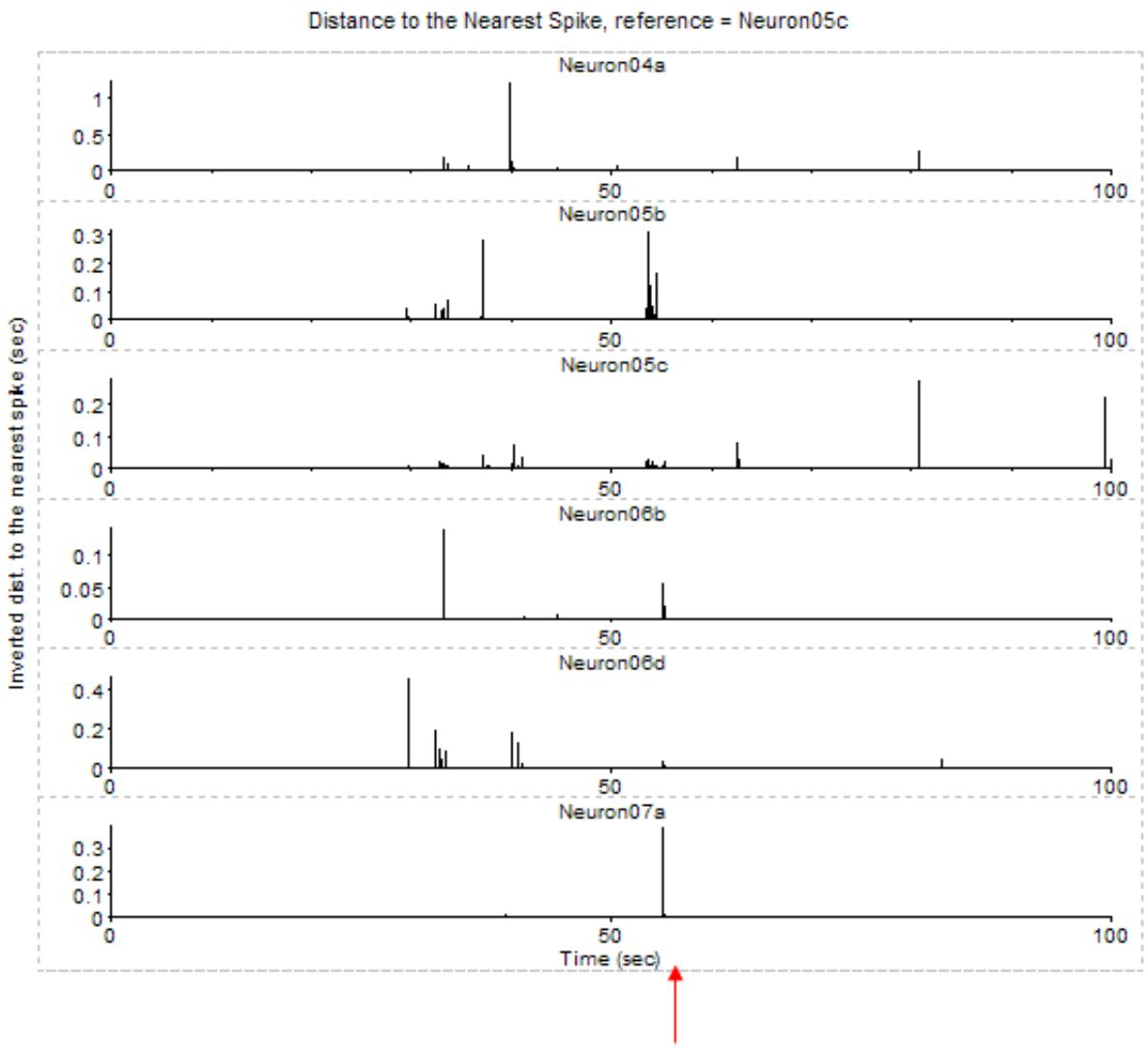
E.D. Young, J.-M. Robert and W.P. Shofner. Regularity and latency of units in ventral cochlear nucleus: implications for unit classification and generation of response properties. J. Neurophysiology, Vol. 60, 1988, 1-29.

Synchrony vs. Time

For each spike, this graph shows the distance from this spike to the closest spike (timestamp) in the reference event.

This analysis is useful in identifying the moments in time when several neurons are in sync with the reference neuron. To do this, you need to select Inverted Distance. Then, high values will indicate spikes that are close to each other. Also, it is useful to specify that the graphs are shown in 1 column (in Num. Columns in Properties panel).

The red arrow in the picture below shows a moment in time where several neurons have almost-synchronous spikes with the reference.



Parameters

Parameter	Description
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Reference	Reference event.
Distance	An option to draw linear or inverted distance

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Spikes	The number of spikes used in calculation
Mean Freq.	Mean firing rate.

Algorithm

For each spike that occurred at time t[i], this graph shows the distance from this spike to the closest spike (timestamp) in the reference event:

`dist = min(abs(t[i] - ref[j])),`

where `ref[j]` is a timestamp of the reference event

If **Distance** is **Linear**, the vertical line is drawn

from point `(t[i], 0.)` to point `(t[i], dist)`.

If **Distance** is **Inverted**, the vertical line is drawn

from point `(t[i], 0.)` to point `(t[i], 1/dist)`.

Analysis of Head Direction Cells

This analysis estimates the frequency of neuronal firing as a function of a head direction of an animal. The head direction should be described by two pairs of continuously recorded variables (head base X and Y and nose X and Y).

Parameters

Parameter	Description
Head Base X Position	Continuous variable that describes X position of the head base of the animal.

continues on next page

Table 59 – continued from previous page

Parameter	Description
Head Base Y Position	Continuous variable that describes Y position of the head base of the animal.
Nose X Position	Continuous variable that describes X position of the nose of the animal.
Nose Y Position	Continuous variable that describes Y position of the nose of the animal.
Bad Position Value	When LED is obscured by the wires, both position variables may suddenly jump to 0 and then jump back to the previous position. All the data points where BaseX, BaseY, NoseX or NoseY is equal to Bad Bad Position Value are ignored.
Min Distance between LEDs	All the data points where the distance between base and nose LEDs is smaller than this value are ignored.
Direction Bin (degrees)	The bin for head directions. Head direction is measured in degrees (from 0 to 360). This parameter represents the size of the head direction bin in degrees.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Histogram minimum.
YMax	Histogram maximum.
Mean Hist.	The mean of the histogram bin values.
St. Dev. Hist.	The standard deviation of the histogram bin values

Algorithm

- Position points that have coordinates that are within 0.001 of the specified 'bad' X and X values are removed
- Head direction is calculated for each position
- Direction bins are `[0, b)`, `[b, 2*b)`, `[2*b, 3*b)`, etc., where `b` is **Direction Bin**
- The intervals in time are calculated where head direction is in one of the direction bins
- The number of of spikes in all the intervals where head direction was in a specific bin (for example, the head direction was from 0 to 10 degrees) is calculated
- The number of spikes is then divided by the duration of the intervals for this bin to get the firing rate at the bin's head direction

Firing Phase

This analysis determines phase relationships between single cell and continuous signal activity.

Parameters

Parameter	Description
Zero Phase Event	Specifies an event variable containing timestamps of the starts of the oscillation cycles. Usually, this variable is created by running *Find Oscillations* analysis.

continues on next page

Table 61 – continued from previous page

Parameter	Description
Oscillation Epochs	Specifies an interval variable that specifies beginning and end of each oscillatory episode. Usually, this variable is created by running *Find Oscillations* analysis.
Number of Phase Bins	Number of bins in [0,360] degrees phase range.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Smooth	Option to smooth the result after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
NumSpikes	The number of spikes used in calculation.
CyclesUsed	The number of oscillation cycles used in calculation

Algorithm

For each interval of the Oscillation Epochs variable:

- Timestamps of Zero Phase Event in this interval are selected
- For each oscillation cycle (a pair of selected Zero Phase Event timestamps `[zero_phase[i], zero_phase[i+1]`):
- Spikes in the oscillation cycle `[zero_phase[i], zero_phase[i+1])` are selected
- For each selected spike, spike phase is calculated as

```
360*(spike_time - zero_phase[i)/(zero_phase[i+1] - zero_phase[i])
```

A histogram of spike phases is calculated. Histogram bin counts are divided by the number of spikes in all oscillation cycles.

References

Klausberger et al. Brain-state- and cell-type-specific firing of hippocampal interneurons in vivo. Nature. 2003 Feb 20;421(6925):844-8

Place Cell Analysis

This analysis estimates the frequency of neuronal firing as a function of position of an animal. The animal position should be described by two continuously recorded variables.

Parameters

Parameter	Description
X Position	Continuous variable that describes X position of the animal.
Y Position	Continuous variable that describes Y position of the animal.
Fix Positions	This option specifies whether NeuroExplorer needs to fix problems that often occur in recording of position variables. For example, when LED is obscured by the wires, both position variables suddenly jump to 0 and then jump back to the previous position.

continues on next page

Table 63 – continued from previous page

Parameter	Description
Fix Threshold	*Fix Positions* parameter. See *Algorithm* section below for details.
Fix Bad X	*Fix Positions* parameter. The 'bad' value of X variable. See *Algorithm* section below for details.
Fix Bad Y	*Fix Positions* parameter. The 'bad' value of Y variable. See *Algorithm* section below for details.
X Min	X axis minimum in X Position units.
X Max	X axis maximum in X Position units.
Y Min	Y axis minimum in Y Position units.
Y Max	Y axis maximum in Y Position units.
Number of Bins (X)	Number of bins along X axis.
Number of Bins (Y)	Number of bins along Y axis.
Display	This parameter determines what kind of Place Cell Analysis display is shown (Path, Firing Positions, Time Spent (in each cell), Number of Visits (to each cell), Spike Counts (for each cell), Firing Rates (for each cell), Firing Fields).
Min. Time Spent	Minimum time spent. This parameter is used only with Firing Rates display. If the animal spent less time in the cell than **Min Time Spent**, the firing rate for the cell is set to zero.
Min Visits	Minimum number of visits. This parameter is used only with Firing Rates display. If the animal visited the cell less than **Min Visits** number of times, the firing rate for the cell is set to zero.
Firing Rate Bin	Bin size (in seconds) to calculate firing rates.
Min Std. Dev. in Field	Minimum standard deviation in the field. See *Algorithm* section below for details.
Min. Pixels in Field	Minimum number of pixels in the field. See *Algorithm* section below for details.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Int. filter type	Specifies if the analysis will use a single or multiple interval filters.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
Create filter on-the-fly	Specifies if a temporary interval filter needs to be created (and used to preselect data).
Create filter around	Specifies an event that will be used to create a temporary filter.

continues on next page

Table 63 – continued from previous page

Parameter	Description
Start offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the start of interval for the temporary filter.
End offset	Offset (in seconds, relative to the event specified in *Create filter around* parameter) for the end of interval for the temporary filter.
Fix overlaps	An option to automatically merge the overlapping intervals in the temporary filter.
Smooth Matrix	An option to smooth the matrix after the calculation. See *Post-Processing Options* for details.
Smooth Radius	The radius of the smoothing filter (in cells) See *Post-Processing Options* for details.
Smooth Colors	An option to smooth colors (blend colors across cells to make smooth color transitions). This option may require considerable computation time.
Bins with Num Visits Less than Min	An option on how to display cells where the number of visits is less than the specified minimum (*Min Visits* parameter).
Bins with Time Spent Less than Min	An option on how to display cells where the time spent is less than the specified minimum (*Min. Time Spent* parameter).
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Spikes	The number of spikes used in calculation.

continues on next page

Table 64 – continued from previous page

Column	Description
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Position ISI mode	Average interval between two consecutive position data points (in seconds).
Mean Firing Rate	Mean firing rate (mean of all the firing rates for all the cells).
St. Dev. Firing Rate	Standard deviation of the firing rate (st. dev. of all the firing rates for all the cells).
Firing Fields	The number of firing fields.
Field N Num Cells	The number of cells in the firing field N.
Field N Centroid X	X coordinate of the field N centroid.
Field N Centroid Y	Y coordinate of the field N centroid.
Field N Peak Rate	Maximum firing rate among the cells inside the firing field N.
Field N Spikes	Number of spikes in field N.
Field N Time Spent	Total time spend in the field N.
Field N Rate	Average firing rate in field N (*Field N Spikes* divided by *Field N Time Spent*).
Field N Coherence	Coherence measures the local smoothness of the firing rate distribution in the field. It is the Pearson correlation of the firing rate in each cell (pixel) with the firing rate averaged over the pixel and its 8 nearest neighbors.

Algorithm

If **Fix Positions** is set to **Use 4 Neighbors**, NeuroExplorer will analyze X and Y position variables and do the following:

```
for each point x[t], calculate the average of its 4 neighbors:

aver = (x[t-2]+x[t-1]+x[t+1]+x[t+2])/4

if abs(x[t] - aver) > FixThreshold, then assign x[t] the average of its␣
↪neighbors:

x[t] = aver
```

If **Fix Positions** is set to **Ignore Bad**, NeuroExplorer will ignore position points that have coordinates that are within 0.001 of the specified 'bad' X and X values.

If **Fix Positions** is set to **Interpolate**, for each position point that has coordinates that are within 0.001 of the specified 'bad' X and X values, the X and Y values are interpolated using the closest previous valid position and the closest next valid position.

The following steps are then performed:

The position space is divided into cells with width (**XMax** - **XMin**)/**Number of Bins (X)** and height (**YMax** - **YMin**)/**Number of Bins (Y).**

For each cell, the number of visits to this cell and the time spent in the cell are calculated.

For each of the neuron firing times, the position of the animal is calculated using linear interpolation of animal positions before and after the spike.

For each cell, the number of times the neuron fired in this cell is calculated.

With **Firing Rates** display, if the animal spent less time in the cell than **Min Time Spent** or visited the cell less than **Min Visits** number of times, the firing rate for the cell is set to zero. Otherwise the number of times the neuron fired in this cell is divided by the time the animal spent in the cell producing the firing rate for the cell.

Firing Field a set of at least **Min. Pixels in Field** contiguous cells where each cell in the field shared at least one side with another cell in the field and each cell had a firing rate greater than (Session_mean + *Min Std. Dev. in Field* * Session_st_dev), where Session_mean is the firing rate of the complete recording session, Session_st_dev is the standard deviation of the mean firing rate. Session_mean and Session_st_dev are calculated the following way: the recording session is divided into time bins of size **Firing Rate Bin**. For each bin i, neuron firing rate R[i] is calculated. Session_mean is the mean of array R[i], Session_st_dev is the standard deviation of array R[i].

Centroid of field location is the arithmetic mean center of a firing field's spatial firing rate distribution determined after the coordinates of each cell in the field was multiplied by the firing rate in the cell.

Coherence measures the local smoothness of the firing rate distribution in the field. It is the Pearson correlation of the firing rate in each cell with the firing rate averaged over the cell and its 8 nearest neighbors.

Reference

A. Fentona, Gyorgy Csizmadiab, and Robert U. Muller. Conjoint Control of Hippocampal Place Cell Firing by Two Visual Stimuli I. The Effects of Moving the Stimuli on Firing Field Positions. *The Journal of General Physiology*, Volume 116, Number 2, August 1, 2000 191-210.

Principal Component Analysis

This analysis calculates eigenvalues and eigenvectors (principal components) of the matrix of covariances of rate histograms. The analysis creates *population vectors* corresponding to the principal components.

Parameters

Parameter	Description
Bin	Bin size in seconds.
Vectors Prefix	A string specifying how the population vector names will be generated. For example, if prefix is **pca1**, the vector names will be **pca1_01**, **pca1_02**, etc.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name and PCA statistics name.
pca1_N	The weight of the variable in the N-th eigenvector. The rows at the bottom of the table also show Eigenvalue of this eigenvector, % of variance it explains, and the cumulative percent of the variance explained by this and preceding eigenvectors.
Corr with VAR	Correlation with the specified variable.

Numerical Results

The **Results** sheet shows for each neuron the weights this neuron has in all the eigenvectors.

Algorithm

Step 1

Rate histograms are calculated for each of the selected neurons.

The time axis is divided into bins. The first bin is `[XMin, XMin+Bin)`. The second bin is `[XMin+Bin, Xmin+Bin*2)`, etc. The left end is included in each bin, the right end is excluded from the bin.

For each bin, the number of events (spikes) in this bin is calculated.

For example, for the first bin

```
bin_count = number of timestamps (ts) such that ts >= XMin and ts < XMin + Bin
```

Bin counts are calculated in such a way for all the selected variables to produce in a matrix `bin_count[i, j]`, where `i` is the neuron number, `j` is the bin number.

Step 2

The matrix of covariances between neurons `c[t, s]` is calculated:

```
c[t, s] = correlation between vectors bin_count[t, *] and bin_count[s, *], s,␣
↪t = 1, ...,number_of_selected_neurons.
```

Step 3

The eigenvalues and eigenvectors are calculated for the matrix `c[t, s]`. The eigenvectors (principal components) are sorted according to their eigenvalues. The first principal component has the largest eigenvalue.

Each principal component becomes a new population vector in the data file.

Reverse Correlation

This analysis is used for estimation of receptive fields in vision research. The analysis calculates the average visual stimulus that preceded the spike.

Parameters

Parameter	Description
Reference	the variable that contains timestamps of the stimuli.
Stim. Sequence File	The path of the text file that contains the sequence of images used for stimulation. See *Algorithm* below for file format description.
Char. per pixel	The number of characters per pixel in image file.
Pixels in Row	The number of pixels in a row for each stimulus image.
Rows in Image	The number of rows of pixels in each stimulus image.
Cache Stim. Data	An option to cache stimulus image data. If this option is selected, the image data is loaded once and stored in memory (until a different stimulus sequence file is specified). To clear cache, run this analysis once with this option disabled.
XMin (deg)	Minimum of the X axis of the average stimulus display (in degrees).
XMax (deg)	Maximum of the X axis of the average stimulus display (in degrees).

continues on next page

Table 67 – continued from previous page

Parameter	Description
YMin (deg)	Minimum of the Y axis of the average stimulus display (in degrees).
YMax (deg)	Maximum of the Y axis of the average stimulus display (in degrees).
Time Min (sec)	Time minimum in seconds.
Time Max (sec)	Time maximum. For example, if Time Min = -0.4 and Time Max = 0, NeuroExplorer will analyze all the stimuli that were presented up to 400 msec before each spike.
Time Bin (sec)	Time bin.
Display	This option specifies what view of the average stimulus will be displayed. See *Algorithm* below.
Show Slice At	This option specifies the slice that will be shown. The units and valid range depend on the Display option See *Algorithm* below.
Z Min	Color scale minimum.
Z Max	Color scale maximum.
Smooth Colors	An option to smooth colors of the average stimulus matrix.
Matrix Scale	An option on what color scale to use when drawing the matrix.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.
Save 3D Matrix	An option to save 3D matrix of numerical results in a .csv file. 3D matrix (in X vs. Y display case) is saved starting from the first slice (at Time Min), then the second slice, etc. Within the slice, the first row corresponds to Y Min, the first column corresponds to X Min.
Save File	The name of the file that will contain 3D matrix data.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
RefCount	Number of reference events (and images used)
Color Scale Min	Minimum of color scale.
Color Scale Max	Maximum of color scale.
Spikes	The number of spikes used in calculation.

Algorithm

Stimulus File Format

NeuroExplorer assumes that the sequence of images used for stimulation is saved in a text file. The file has the following format:

- each line of the file represents a row of pixels in the stimulus image
- each pixel is represented by an integer
- each pixel has the same number of characters used for its description
- images are saved in the file in the order they were presented
- images can be (optionally) separated by blank lines

Here is an example of an image file with 2 images (15 pixels per row, 12 rows per image):

```
0  1  1  1  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

(continues on next page)

(continued from previous page)

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  1  1  1  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

The computational algorithm works the following way. For each spike of a selected neuron, NeuroExplorer identifies the images that precede the spike and are within the specified time interval (between Tmin and Tmax). For example, if Tmin = -0.4 and Tmax = 0, NeuroExplorer will analyze all the stimuli that were presented up to 400 msec before each spike.

Then, NeuroExplorer calculates the average of the presented stimuli over all the spikes. The result is a 3-dimensional matrix (with X, Y, and Time dimensions, time axis is divided into bins of the specified size).

Display option identifies what 2-dimensional view of this matrix is presented:

X vs. Y display shows an average stimulus image for the specified time slice

X vs. Time display shows average X pixels of the stimuli for the specified Y value

Y vs. Time display shows average Y pixels of the stimuli for the specified X value

Reference

Izumi Ohzawa, Gregory C. DeAngelis, and Ralph D. Freeman (1996). Encoding of binocular disparity by simple cells in the cat's visual cortex. J. Neurophysiol. 75: 1779-1805.

Analyses of Continuous Data

Band Energy versus Time

This analysis calculates energy of a specified frequency band over time.

Parameters

Parameter	Description
XMin (sec)	Analysis is run on the data from a specific time range [XMin, XMax]. XMin specifies time range minimum in seconds.
XMax (sec)	Analysis is run on the data from a specific time range [XMin, XMax]. XMax specifies time range maximum in seconds.
Window Width (sec)	Band energies are calculated in consecutive time windows. This parameter specifies the width of each window in seconds.
Use Custom Window Shift	By default (if **Use Custom Window Shift** is not checked), window shift is equal to **Window Width (sec)** (so the analysis is done over consecutive non-overlapping windows). If **Use Custom Window Shift** option is selected, the shift is specified by **Window Shift (sec)** parameter.
Window Shift (sec)	If **Use Custom Window Shift** is not checked, window shift is equal to **Window Width (sec)** and this parameter is ignored. If **Use Custom Window Shift** option is selected, the shift is specified by **Window Shift (sec)** parameter.
Spectrum Normalization	Spectrum units (see **Normalization** below).
Band Min. Freq. (Hz)	Specifies minimum of the frequency band in Hz (for example, 3 Hz for theta band).
Band Max. Freq. (Hz)	Specifies maximum of the frequency band in Hz (for example, 6 Hz for theta band).
Band Energy Measure	Specifies how the band energy is measured.
Create Continuous Variables	An option to create continuous variables with band energy.
Result Prefix	Prefix for the created continuous variables with band energy.
Select Data	If Select Data is *From Time Range*, only the data from the specified time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options* .
Smooth	Option to smooth the result after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.

continues on next page

Table 69 – continued from previous page

Parameter	Description
Add to Results / Bin middle	An option to add an additional vector (containing a center of each frequency bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each frequency bin) to the matrix of numerical results.
Result Prefix	The prefix for generating names of the continuous variables. For example, if we analyze variable LFP and the prefix is Theta, the generated continuous variable will be named Theta_LFP.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options* .
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Algorithm

For each window, the spectrum of the signal in this window is calculated. For the values of the specified frequency band, average, sum, percent of these values or the area below the spectrum curve (for *Integral* option of the *Band Energy Measure*) is calculated.

The Integral measure (the area below the spectrum curve) is calculated using the Simpson rule (similar to *simpson* function of the Python *scipy.integrate* package). See Compute the average bandpower of an EEG signal.

Normalization

If **Normalization** is Raw PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values is equal to the mean squared value of the rate histogram. The formulas (13.4.10) of Numerical Recipes in C are used. (Numerical Recipes in C, Press, Flannery, et al. (Cambridge University Press, 1992))

If **Normalization** is % of Total PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values equals 100.

If **Normalization** is Log of PSD (Numerical Recipes), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum[i])
```

where raw_spectrum is calculated as described above in Raw PSD (Numerical Recipes).

If **Normalization** is Raw PSD (Matlab), the power spectrum is normalized as described in https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html

If **Normalization** is Log of PSD (Matlab), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum_matlab[i])
```

where raw_spectrum_matlab is calculated as described above in Raw PSD (Matlab).

Coherence Analysis for Continuous Variables

Coherence is a measure of the degree of relationship, as a function of frequency, between two time series.

Parameters

Parameter	Description
Reference	Specifies a reference variable.
Number of Fr. Values	Number of values in the spectrum.
Window Overlap (%)	Window overlap in percent (from 0 to 90).
Window Preprocessing	Preprocessing to be done for each window before calculating the spectrum of the window.
Windowing Function	Windowing Function to be applied to each window before calculating the spectrum of the window.
Discard Incomplete Windows	Option to discard the last window if it contains less than the expected number of data points (less than NumberOfFrValues*2 points).
Use Multitaper Algorithm	Option to use multi-taper algorithm.
Time-Bandwidth Product	Time-Bandwidth Product when using multi-taper algorithm.
Number of Tapers	Number of Tapers when using multi-taper algorithm.
Show Freq. From	An option to show a subset of frequencies. Specifies minimum frequency to be displayed.

continues on next page

Table 70 – continued from previous page

Parameter	Description
Show Freq. To	An option to show a subset of frequencies. Specifies maximum frequency to be displayed.
Calculate	Specifies whether coherence values or coherence phases are calculated.
Confidence Level (%)	Confidence level (percent) for the coherence values. Can only be calculated if a single-taper algorithm is used. See Confidence Level Calculation below for details.
Draw confidence level	An option to draw the confidence level. Confidence level is not drawn if coherence phases are specified in Calculate parameter or if a multi-taper algorithm is used.
Conf. line style	Line style for drawing confidence level.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Add to Results / Bin left	An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.
Add to Results / Bin middle	An option to add an additional vector (containing a center of each frequency bin) to the matrix of numerical results.
Add to Results / Bin right	An option to add an additional vector (containing a right edge of each frequency bin) to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Y Min	Y axis minimum.
Y Max	Y axis maximum.
Filter Length	The length of the interval filter.
Confidence Level	Confidence level for the coherence values

Algorithm

For two variables `X` (reference) and `Y` (target) the following entities are calculated. FFTs of the data segments (after preprocessing) are calculated. Then, individual and cross-densities are calculated:

```
Pxx = FFT(X)*Conj(FFT(X)),

Pyy = FFT(Y)*Conj(FFT(Y))

Pxy = FFT(X)*Conj(FFT(Y)).
```

Here `Conj(z)` is a complex conjugate of `z`. `Pxx`, `Pyy` and `Pxy` values are averaged across all intervals and coherence values are calculated as

```
Mean(Pxy)*Mean(Pxy) / (Mean(Pxx)*Mean(Pyy)).
```

Coherence phase values are calculated as phase of `Mean(Pxy)`.

Calculation of the Confidence Level

Confidence levels can only be calculated if a single-taper algorithm is used.

Confidence levels are calculated as described in Kattla and Lowery (2010) (see Reference blow). The confidence level `Z` is calculated as

```
Z = 1 - pow(alpha, 1/(w*L - 1))
```

where

`pow(x,y)` returns `x` to the power of `y`,

`alpha = 1 - Confidence_Level*0.01`,

`L` is the number of overlapped windows

`w` is the correction due to the Hanning window tapering (see eq. (5) and (6) in Kattla and Lowery (2010)).

Reference

Kattla S, Lowery MM. Fatigue related changes in electromyographic coherence between synergistic hand muscles. Exp Brain Res. 2010 Apr; 202(1):89-99. Epub 2009 Dec 12.

Find Oscillations

This analysis identifies episodes of oscillatory activity in the specified frequency band in recorded analog signals.

Parameters

Parameter	Description
XMin (sec)	Analysis is run on the data from a specific time range [XMin, XMax]. XMin specifies time range minimum in seconds.
XMax (sec)	Analysis is run on the data from a specific time range [XMin, XMax]. XMax specifies time range maximum in seconds.
Window Width (sec)	Frequency power ratios are calculated in consecutive time windows. This parameter specifies the width of each window in seconds.
Use Custom Window Shift	By default (if **Use Custom Window Shift** is not checked), window shift is equal to **Window Width (sec)** (so the analysis is done over consecutive non-overlapping windows). If **Use Custom Window Shift** option is selected, the shift is specified by **Window Shift (sec)** parameter.
Window Shift (sec)	If **Use Custom Window Shift** is not checked, window shift is equal to **Window Width (sec)** and this parameter is ignored. If **Use Custom Window Shift** option is selected, the shift is specified by **Window Shift (sec)** parameter.
Main Band Min Freq (Hz)	Specifies minimum of the main frequency band in Hz (for example, 3 Hz for theta band).
Main Band Max Freq (Hz)	Specifies maximum of the main frequency band in Hz (for example, 6 Hz for theta band).

continues on next page

Table 72 – continued from previous page

Parameter	Description
Method	Specifies what method to use to identify periods with oscillations. There are two method options: **Use Ratio to Other Band** or **Use Percent of Main Band**.
Second Band Min Freq (Hz)	Specifies minimum of the second frequency band in Hz (for example, 2 Hz for delta band). If Method is **Use Percent of Main Band**, this parameter is ignored.
Second Band Max Freq (Hz)	Specifies maximum of the second frequency band in Hz (for example, 3 Hz for delta band). If Method is **Use Percent of Main Band**, this parameter is ignored.
Min Power Ratio	Ratio of the power of the main frequency to the power of the second frequency must exceed this value. If Method is **Use Percent of Main Band**, this parameter is ignored.
Min Percent	Percent of spectrum energy of the main band must exceed this value. If Method is **Use Ratio to Other Band**, this parameter is ignored.
Min Number of Windows	If Method is Use Ratio to Other Band, the Ratio must exceed **Min Power Ratio** in a number of consecutive windows. This parameter specifies minimum number of consecutive windows. If Method is Use Percent of Main Band, the Percent of spectrum energy of the main band must exceed **Min Percent** in a number of consecutive windows. This parameter specifies minimum number of consecutive windows.
Filter Type	The type of the frequency filter.
Filter Order	The order of the FIR frequency filter. The order should be 4 or higher. If FIR filter order is odd, order+1 is used.
IIR Filter Order	The order of the IIR Butterworth frequency filter. 2 is recommended.
Result Prefix	The prefix for generating names of the results. For example, if we analyze variable LFP and the prefix is Theta, the results will have names LFP_Theta_Epochs, LFP_Theta_Filtered and LFP_Theta_ZeroPhase.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.

continues on next page

Table 72 – continued from previous page

Parameter	Description
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Analysis Results

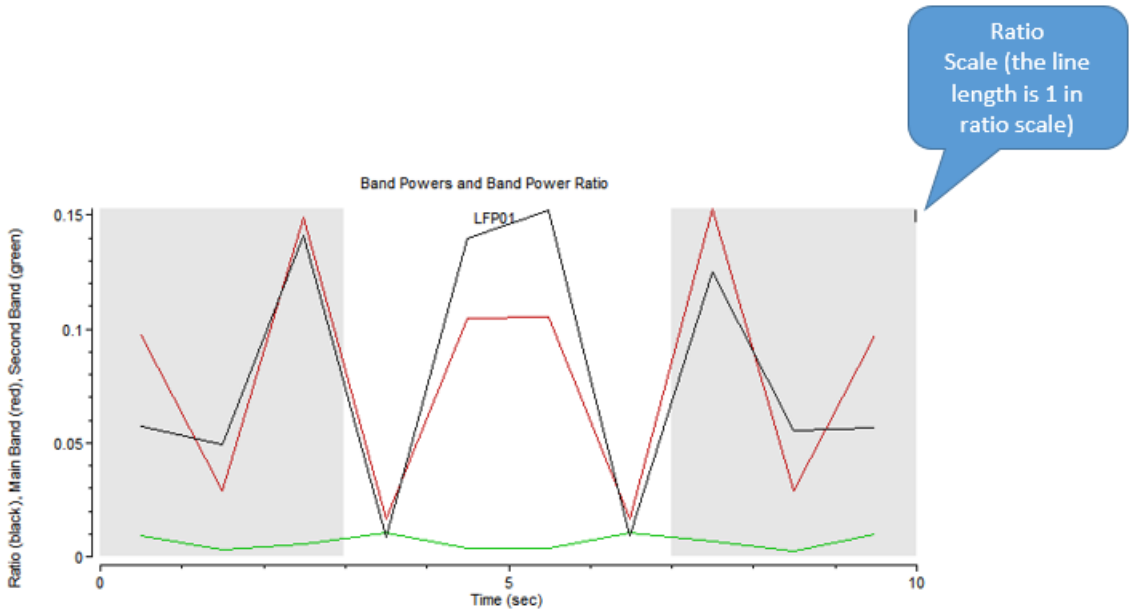
This analysis adds several new variables to the data file. For example, if we analyze continuous variable LFP and the **Result Prefix** is Theta, this analysis will add the following new variables to the file:

- LFP_Theta_Epochs (interval variable that specifies beginning and end of each oscillatory episode)
- LFP_Theta_ZeroPhase (event variable containing timestamps of the starts of the oscillation cycles)
- LFP_Theta_Filtered (continuous variable containing band-filtered LFP)

Variables LFP_Theta_ZeroPhase and LFP_Theta_Epochs can then be used in *Firing Phase* analysis.

Graphical results display the values of energy for each band (red line for the main band energy and green line for the second band energy) as well as the values of power ratio (black line).

Y axis shows the scale for the band energy values. The scale for the power ratio is shown as a vertical line in the upper-right corner of the graph:



The gray background areas show the identified epochs of oscillatory activity.

Algorithm

For each window, the spectrum of the signal in this window is calculated.

If Method is **Use Ratio to Other Band**:

- The power of each band is calculated as an average spectrum value for the frequencies in the band.
- If Min Power Ratio is 4 and Min Number of Windows is 3, a ratio greater than 4 in at least 3 consecutive windows is identified as a start of an oscillatory epoch. The subsequent windows are added to the epoch if the band power ratio for each window is greater than 4.

If Method is **Use Percent of Main Band**:

- The percent or spectrum values in the Main Band is calculated.
- If Min Percent is 2 and Min Number of Windows is 3, a percent of main band greater than 2 in at least 3 consecutive windows is identified as a start of an oscillatory epoch. The subsequent windows are added to the epoch if the percent of the main band for each window is greater than 2.

After the oscillatory epochs are identified, the continuous variable is band-filtered within these epochs (with filter band specified as [Main Band Min Freq, Main Band Max Freq]). Hilbert transform is then applied to the filtered signal. The jumps of the phase of the Hilbert transform from 360 degrees to zero are identified as starts of the oscillation cycles.

References

Klausberger et al. Brain-state- and cell-type-specific firing of hippocampal interneurons in vivo. Nature. 2003 Feb 20;421(6925):844-8

Find Ripples

This analysis finds ripples in continuous channels using the algorithm described in Buzsaki lab GitHub repository function FindRipples.m.

Graphical display produced by this analysis shows ripple positions and durations.

This analysis can also add interval variables containing time intervals corresponding to ripples.

Parameters

Parameter	Description
Filter Frequency From	Minimum frequency (Hz) of the band-pass filter.
Filter Frequency To	Maximum frequency (Hz) of the band-pass filter.
Smooth Filter Width (seconds)	Width of the boxcar smooth filter.
Threshold to Start and End Ripple (number of standard deviations)	Threshold value to start and end ripples (see Algorithm below).
Minimum Ripple Peak Value (number of standard deviations)	Minimum ripple peak value (see Algorithm below).
Minimum Time Between Ripples (seconds)	Minimum time between ripples.
Minimum Ripple Duration (seconds)	Minimum ripple duration.
Maximum Ripple Duration (seconds)	Maximum ripple duration.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.

continues on next page

Table 73 – continued from previous page

Parameter	Description
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Number of Ripples	Number of ripples found.
Ripples per Seconds	Number of ripples per second.
Mean Ripple Duration	Mean ripple duration.
Mean Ripple Amplitude	Mean ripple amplitude.
Mean Peak Amplitude	Mean peak amplitude.
Mean Peak Frequency	Mean peak frequency.

Algorithm

1) Signal is bandpass filtered in `[Freq_From, Freq_To]` frequency range using the 3rd order Butterworth filter.
2) The result of 1) is squared.
3) The result of 2) is smoothed using boxcar filter of the specified length.
4) The result of 3) is normalized to unity standard deviation: `r4 = (r3-mean(r3))/std(r3)`.
5) Ripple starts are time points where `r4` crosses threshold going up.
6) Ripple ends are time points where `r4` crosses threshold going down.
7) Incomplete ripples are removed.
8) Ripples with short inter-ripple periods are merged.
9) Ripples with a maximum value less than `Minimum_Ripple_Peak_Value` are discarded.
10) Ripples that are way too long are discarded.
11) Ripples that are too short are discarded.
12) For each ripple, the following values are calculated:

- Ripple duration
- Average value of 1) within ripple (mean ripple amplitude)
- Maximum value of 1) within ripple (peak ripple amplitude)
- Spectrum of values of 1) within ripple is calculated using DFT with no pre-processing
- Peak ripple frequency is calculated as the frequency value where the spectrum is maximum

13) Values calculated in step 12) are averaged over all ripples for the current continuous channel.

Reference

Buzsaki lab GitHub repository function FindRipples.m.

Perievent Rasters for Continuous

This analysis shows the traces of the selected continuous variable relative to the timestamps of the reference variable.

Parameters

Parameter	Description
Reference	Specifies a reference neuron or event.
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Use last ref. event	An option to draw the trace only for the last reference event.
Round ref. timestamp	An option to align reference event timestamps to the timestamps of the continuous variable. This option is valid only if XMin <0 and XMax >0.
Draw Variation	An option to draw variation around the mean as a colored background.
Variation (standard deviations)	Number of standard deviations in the colored background around the mean.
Variation Background Color	The color to be used for drawing of the colored background around the mean.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.

continues on next page

Table 75 – continued from previous page

Parameter	Description
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
Reference	Reference variable name.
NumRefEvents	The number of reference events used in calculation
YMin	Minimum of the average trace shown at the bottom.
YMax	Minimum of the average trace shown at the bottom.

Algorithm

Let `ref[i]` be the array of timestamps of the reference event.

For each timestamp `ref[k]`, NeuroExplorer draws the values of continuous variable with timestamps from `ref[k] + XMin` to `ref[k] + XMax`. The timestamps of the continuous variables are shown relative to the `ref[k]`: if the timestamp of continuous variable data point is `cont_ts[i]`, the timestamp shown in Numerical Results is `relative_ts[i] = cont_ts[i] - ref[k]`.

If **Round ref. timestamp** option is used, NeuroExplorer shifts the relative timestamps of the

continuous variable. The program finds the last timestamp `cont_ts[j]` of the continuous variable before `ref[k]`. Then, the program adjusts the relative timestamps of the continuous variable so that

```
relative_ts_adjusted[j] = ref[k]
```

If `diff = ref[k] - cont_ts[j]` then `relative_ts_adjusted[i] = relative_ts[i] + diff`.

Mean values of the perievent raster shown at the bottom graph are calculated using linear interpolation.

Phase Analysis via Hilbert Transform

This analysis graphs amplitude and phase of a continuous signal in a single trial.

Parameters

Parameter	Description
Reference	Specifies a reference neuron or event.
XMin	Time axis minimum in seconds.
XMax	Time axis maximum in seconds
Ref. Timestamp Number	Reference timestamp number.
Bandpass Filter From (Hz)	Minimum frequency for the bandpass filter.
Bandpass Filter To (Hz)	Maximum frequency for the bandpass filter.
Bandpass Filter Type	The type of the bandpass filter.
FIR Filter Order	The order of the bandpass FIR filter. The order should be 4 or higher. If FIR filter order is odd, order+1 is used.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name
YMin	Graph minimum
YMax	Graph maximum

Algorithm

The signal for the specified trial is bandpass filtered and the analytic signal of the result is calculated via Hilbert transform.

The original signal is drawn in light gray, the filtered signal is drawn in dark gray, the amplitude of the filtered signal is drawn in green and the phase is drawn in red.

Power Spectral Densities for Continuous Variables

This analysis captures the frequency content of continuous variables.

Parameters

Parameter	Description
Number of Fr. Values	Number of values in the spectrum.
Window Overlap (%)	Window (segment) overlap in percent (from 0 to 90).
Window Preprocessing	Preprocessing to be done for each window before calculating the spectrum of the window.
Windowing Function	Windowing Function to be applied to each window before calculating the spectrum of the window.
Discard Incomplete Windows	Option to discard the last window if it contains less than the expected number of data points (less than `NumberOfFrValues*2` points).

continues on next page

Table 79 – continued from previous page

Parameter	Description
Concatenate Selected Data Points	If this option is selected, data points that are selected for analysis (that is, points inside time interval filter) are concatenated and then the PSD of the concatenated signal is calculated. This means that the signal from two adjacent time intervals can be in the same data segment used to calculate FFT. If this option is not selected and interval filter is specified, data segments used to calculate FFT (see Algorithm below) are guaranteed to contain data points from one interval of the interval filter. For each time interval, the first data segment in the interval is aligned with the interval start.
Use Multitaper Algorithm	Option to use multi-taper algorithm.
Time-Bandwidth Product	Time-Bandwidth Product when using multi-taper algorithm.
Number of Tapers	Number of Tapers when using multi-taper algorithm.
Show Frequency From	Minimum frequency to be shown in the spectrum graph.
Show Frequency To	Maximum frequency to be shown in the spectrum graph.
Normalization	Spectrum normalization. See *Algorithm* below.
Frequency Bands	Percent of spectrum values and the sum of spectrum values will be calculated for each of the specified frequency bands. If *Log of PSD* Normalization is specified, band percents will be calculated BEFORE applying log() transform.
Log Frequency Scale	An option show Log10 frequency scale.
Smooth	Option to smooth the spectrum after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Add Freq. Values to Results	An option to add an additional vector of frequency values to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options* .
Matrix Name	Specifies the name of the results matrix in Matlab workspace.

continues on next page

Table 79 – continued from previous page

Parameter	Description
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options* .
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Spectrum minimum.
YMax	Spectrum maximum.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Number of FFT windows	Number of FFT windows used in analysis.
Frequency of Minimum	The position of the minimum of the displayed spectrum (in Hz). If there are multiple bins in the spectrum where the spectrum value is equal to the spectrum minimum, this value represents the frequency of the first such bin. If you need to calculate spectrum peaks, go to Post-processing tab, press Post-processing Script Options button and select PostAnalysis_Peaks.py
Frequency of Maximum	The position of the maximum of the displayed spectrum (in Hz). If there are multiple bins in the spectrum where the spectrum value is equal to the spectrum maximum, this value represents the frequency of the first such bin. If you need to calculate spectrum peaks, go to Post-processing tab, press Post-processing Script Options button and select PostAnalysis_Peaks.py

Algorithm

For each selected continuous variable, a standard or multi-taper power spectrum is calculated.

In each case, a Welsh periodogram method is used:

The original data array is split up into data segments (or windows) of length `2*Nf` (where `Nf` is the Number of Frequency Values), overlapping by `D` points. For example, if overlap is 50%, then `D` is `(2*Nf)/2=Nf`.

For each segment, the signal is preprocessed according to Window Preprocessing parameter. For example, if Subtract Mean is selected, `ProcessedSignal[i] = Signal[i] - meanOfSignalInSegment`.

The overlapping segments are then windowed: after the data is split up into overlapping segments, the individual data segments have a window applied to them (that is, `ProcessedWindowedSignal[i] = ProcessedSignal[i]*WindowValue[i]`; the window is specified by the Windowing Function).

Most window functions afford more influence to the data at the center of the segment than to data at the edges, which represents a loss of information. To mitigate that loss, the individual data segments are commonly overlapped in time (as in the above step).

After doing the above, the periodogram is calculated by computing the discrete Fourier transform, and then computing the squared magnitude of the result. The individual periodograms are then time-averaged, which reduces the variance of the individual power measurements. The end result is an array of power measurements vs. frequency "bin".

For multi-taper spectral estimate, several periodograms (tapers) are calculated for each segment. Each taper is calculated by applying a specially designed windowing function (Slepian function, see https://en.wikipedia.org/wiki/Multitaper). All the tapers for a given segment are then averaged to form the periodogram of the segment. The individual segment periodograms are then time-averaged.

See http://www.spectraworks.com/Help/mtmtheory.html for a discussion on selecting Multi-Taper parameters.

Normalization

If **Normalization** is Raw PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values is equal to the mean squared value of the signal. The formulas (13.4.10) of Numerical Recipes in C are used (squared fft values are divided by `N^2` where `N` is the number of values in data window). (Numerical Recipes in C, Press, Flannery, et al. (Cambridge University Press, 1992)). The units are signal_units^2 (mV^2 in NeuroExplorer). The units in the frequency band sums are mV^2.

If **Normalization** is % of Total PSD (Numerical Recipes), the power spectrum is normalized so that the sum of all the spectrum values equals 100. The units are %. The units in the frequency band sums are mV^2.

If **Normalization** is Log of PSD (Numerical Recipes), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum[i])
```

where raw_spectrum is calculated as described above in Raw PSD (Numerical Recipes). The units are dB. The units in the frequency band sums are mV^2.

If **Normalization** is Raw PSD (Matlab), the power spectrum is normalized as described in https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html (squared fft values are divided by `Fs*N`, where `Fs` is sampling rate, `N` is the number of values in window, see Matlab code below). The units are signal_units^2/Hz (mV^2/Hz in NeuroExplorer). The units in the frequency band sums are mV^2/Hz.

If **Normalization** is Log of PSD (Matlab), the power spectrum is calculated using the formula:

```
power_spectrum[i] = 10.*log10(raw_spectrum_matlab[i])
```

where raw_spectrum_matlab is calculated as described above in Raw PSD (Matlab). The units are dB/Hz. The units in frequency band sums are mV^2/Hz.

In Matlab:

```
Fs = 1000;
t = 0:1/Fs:1-1/Fs;
x = cos(2*pi*100*t) + randn(size(t));
xdft = fft(x);
N = length(x);
xdft = xdft(1:N/2+1);

% psdx below corresponds to Raw PSD (Matlab) normalization
% the units are signal_units^2/Hz (mV^2/Hz in NeuroExplorer)
% note that we divide squared fft by (Fs*N)

psdx = (1/(Fs*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);

% psdDb below corresponds to Log of Raw PSD (Matlab) normalization

psdDb = 10*log10(psdx)

freq = 0:Fs/length(x):Fs/2;
```

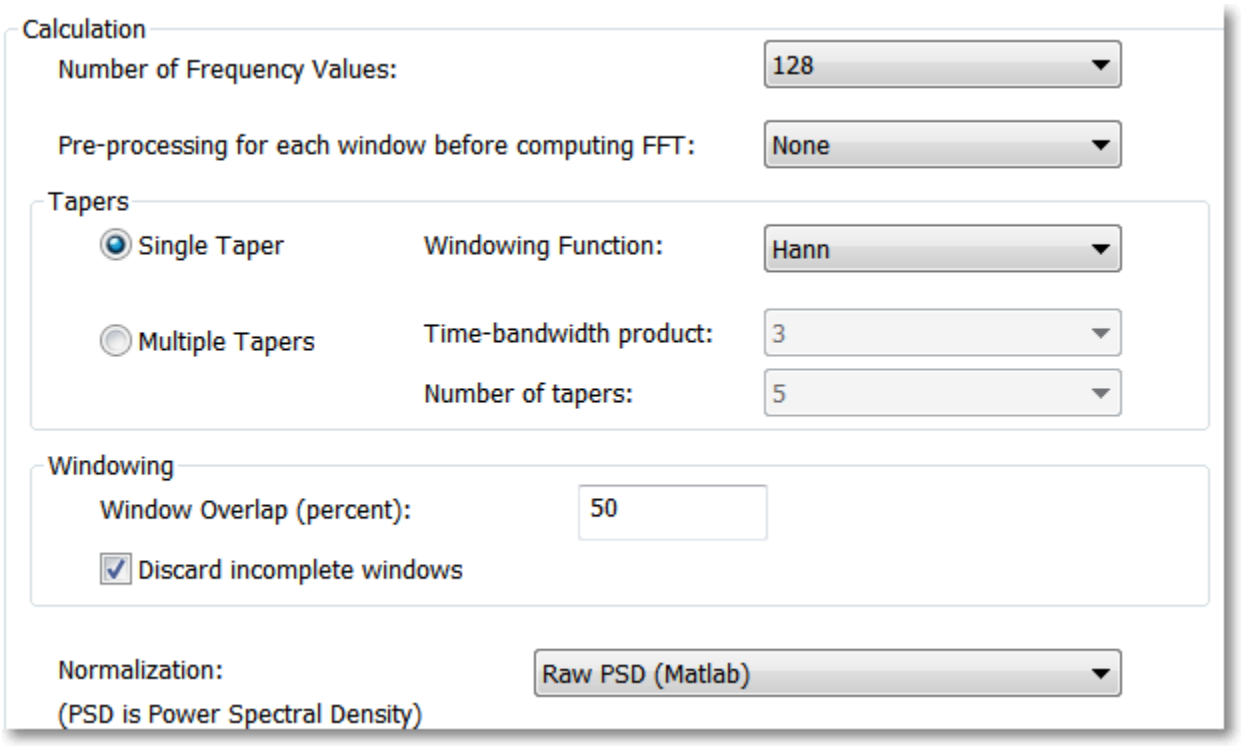
(continues on next page)

(continued from previous page)

```
plot(freq,psDb)
title('Periodogram Using FFT')
xlabel('Frequency (Hz)')
ylabel('Power/Frequency (dB/Hz)')
```

Here is how you can verify that calculations in NeuroExplorer produce the same values as spectrum calculations in Matlab:

- Open the file TestDataFile5.nex
- Deselect all neuronal variables
- Select ContChannel01 variable
- Select **Matlab | Send Selected Variables** to Matlab menu command
- Select **Power Spectra for Continuous** analysis
- Specify the following parameters:



- Make sure that **Add columns(s) with frequency values** option is disabled in Post-Processing tab of Analysis Properties dialog
- Run the analysis
- Select **Matlab | Send Numerical Results to Matlab** menu command

- In Matlab, execute:

```
p=pwelch(ContChannel01(:,1), hann(256), 128, 256, 10000);

maxDiff = max(abs(p-nex(:,1)))
```

- You will see the following result:

```
1.3500e-13
```

This means that at least the first 11 decimal places of the PSD values calculated in NeuroExplorer and in Matlab are identical.

Single Trial Spectrum Analysis

This analysis captures the frequency content of continuous variables for a single trial.

Parameters

Parameter	Description
Reference	Specifies a reference neuron or event.
XMin	Time window minimum in seconds.
XMax	Time window maximum in seconds
Ref. Timestamp Number	Reference timestamp number.
Number of Fr. Values	Number of values in the spectrum.
Window Overlap (%)	Window overlap in percent (from 0 to 90).
Window Preprocessing	Preprocessing to be done for each window before calculating the spectrum of the window.
Windowing Function	Windowing Function to be applied to each window before calculating the spectrum of the window.
Discard Incomplete Windows	Option to discard the last window if it contains less than the expected number of data points (less than `NumberOfFrValues*2` points).
Use Multitaper Algorithm	Option to use multi-taper algorithm.
Time-Bandwidth Product	Time-Bandwidth Product when using multi-taper algorithm.
Number of Tapers	Number of Tapers when using multi-taper algorithm.
Show Frequency From	Minimum frequency to be shown in the spectrum graph.
Show Frequency To	Maximum frequency to be shown in the spectrum graph.
Normalization	Spectrum normalization. See *Algorithm* below.

continues on next page

Table 81 – continued from previous page

Parameter	Description
Log Frequency Scale	An option show Log10 frequency scale.
Smooth	Option to smooth the spectrum after the calculation. See *Post-Processing Options* for details.
Smooth Filter Width	The width of the smooth filter. See *Post-Processing Options* for details.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter that will be used to preselect data before analysis. See also *Data Selection Options*.
Add Freq. Values to Results	An option to add an additional vector of frequency values to the matrix of numerical results.
Send to Matlab	An option to send the matrix of numerical results to Matlab. See also *Matlab Options*.
Matrix Name	Specifies the name of the results matrix in Matlab workspace.
Matlab command	Specifies a Matlab command that is executed after the numerical results are sent to Matlab.
Send to Excel	An option to send numerical results or summary of numerical results to Excel. See also *Excel Options*.
Sheet Name	The name of the worksheet in Excel where to copy the numerical results.
TopLeft	Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Spectrum minimum.
YMax	Spectrum maximum.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).
Number of FFT windows	Number of FFT windows used in analysis.

continues on next page

Table 82 – continued from previous page

Column	Description
Frequency of Minimum	The position of the minimum of the displayed spectrum (in Hz). If there are multiple bins in the spectrum where the spectrum value is equal to the spectrum minimum, this value represents the frequency of the first such bin.
Frequency of Maximum	The position of the maximum of the displayed spectrum (in Hz). If there are multiple bins in the spectrum where the spectrum value is equal to the spectrum maximum, this value represents the frequency of the first such bin.

Algorithm

For each selected continuous variable, a standard or multi-taper power spectrum is calculated. See *Power Spectral Densities for Continuous Variables* for details.

Spike Detection

This analysis optionally subtracts reference channel, band-pass filters continuous variables and detects spikes using threshold crossing algorithm. Detected spikes are added to current data file as new waveform variables.

Parameters

Parameter	Description
Subtract Reference	An option to subtract reference channel before filtering and spike detection.
Reference Channel	Reference Channel to subtract before filtering and spike detection.
Filter Before Detection	An option to band-pass filter continuous variables (using Butterworth filter) before spike detection.
Filter Order	The Butterworth filter order.
Filter Frequency From	Minimum frequency (Hz) of the band-pass filter.
Filter Frequency To	Maximum frequency (Hz) of the band-pass filter.
Threshold type	An option specifying how the threshold is calculated (see Algorithm below).
Custom Threshold	Threshold value or number of standard deviations (can be negative, see Algorithm below).

continues on next page

Table 83 – continued from previous page

Parameter	Description
Threshold Crossing Type	Type of threshold crossing to use. **Auto** (if threshold is non-negative, detect a spike when the signal crosses the threshold when signal is going up, if threshold is negative, detect a spike when the signal crosses the threshold when signal is going down); **Up** (detect a spike when the signal crosses the threshold when signal is going up); **Down** (detect a spike when the signal crosses the threshold when signal is going down).
Segment Before Threshold (mS)	Signal segment before threshold to include into spike waveform.
Segment After Threshold (mS)	Signal segment after threshold to include into spike waveform.
Minimum Time Between Spikes	Minimum time between spikes.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options* .
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y axis minimum.
YMax	Y axis maximum.
Threshold	The threshold value used to detect spikes (in mV)
Spikes Detected	The number of spikes detected.

Algorithm

The standard deviation of the signal (`SD`) and the median of the signal absolute values (`MA`) are calculated. 'Median Sigma' (`MS`) is equal to `MA/0.6745` (see eq. 3.1 in Reference).

- If the threshold type is **Auto**, the threshold is equal to `-4.0*MS` (see eq. 3.1 in Reference).
- If the threshold type is **Number of Median Sigma**, the threshold is equal to `Custom_Threshold*MS`.
- If the threshold type is **Number of Standard Deviations**, the threshold is equal to `Custom_Threshold*SD`.
- If the threshold type is **Absolute**, the threshold is equal to `Custom_Threshold`.

The spike is detected if the signal crosses the threshold when the signal is going up or down (depending on the value of the **Threshold Crossing Type** parameter).

Reference

Neural Comput. 2004 Aug;16(8):1661-87. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Quiroga R, Nadasdy Z, Ben-Shaul Y.

Analyses of Waveforms

Sort Spikes

This analysis groups spikes into clusters based on the similarity of their shapes.

Clustering algorithm in this analysis is similar to the one used in the clustering step of the SpyKING CIRCUS spike sorting toolbox.

Parameters

Parameter	Description
Percent of Waveforms in Neighborhood	Percent of waveforms to use when calculating the density of data points around each waveform in the PCA projection space. See **Algorithm** below.
Maximum Initial Number of Clusters	Maximum initial number of clusters. Similar clusters will be merged later.

continues on next page

Table 85 – continued from previous page

Parameter	Description
Cluster Merge Threshold	Threshold to be used when merging clusters. See **Algorithm** below.
Confidence Level for Outliers	Confidence level (in percent) for detecting outliers. See **Algorithm** below.
Create Neurons	An option to create neuron variables for each cluster (waveform variables for each cluster are always created).

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
XMin	X Axis minimum in the PCA projections space.
XMax	X Axis maximum in the PCA projections space.
YMin	Y Axis minimum in the PCA projections space.
YMax	Y Axis maximum in the PCA projections space.

Algorithm

The program selects waveforms in the specified time range and the interval filter.

Principal components are calculated using selected `N` waveforms of the given waveform variable.

First, the matrix of covariances between waveform points (`c[t, s]`) is calculated:

`c[t, s]` = covariance between vectors `waveform_value[t, *]` and `waveform_value[s,*]`, `s, t = 1, ...,number_of_points_in_each_waveform`.

Then, the eigenvalues and eigenvectors are calculated for the matrix `c[t, s]`. The eigenvectors (principal components) are sorted according to their eigenvalues. The first principal component has the largest eigenvalue.

Analysis graph shows the scatter plot where x and y are projections of the selected waveforms to the first two principal components (projection is a sum of products `waveform_value[t]*principal_component_value[t]`).

The points in the PCA projections space are then used for cluster analysis.

For each point, the mean distance `R` to the nearest `S` points is calculated, where

```
S = Number_of_waveforms * Percent_of_Waveforms_in_Neighborhood/100
```

Then, the distance `D` to the nearest point with a lower `R` (or higher density) is calculated for each data point.

The intuition of the algorithm is that the cluster centroids should be the points with a high density (i.e. low `R`) and far apart from other points with higher density (high `D`).

The `M` points (where `M = Maximum_Initial_Number_of_Clusters`) with the highest ratios `D/R` are considered as initial cluster centroids. Each point is then assigned to the same cluster as the closest point with a higher density (lower `R`).

Normalized distances `Gamma` between clusters are calculated according to equation (2) of the publication describing the details of the SpyKING CIRCUS algorithm. The pairs of clusters with `Gamma` less than `Cluster_Merge_Threshold` are merged.

For each cluster, the `Confidence_Level_for_Outliers` percentile `P` for the `R` values of all the points in the cluster is calculated using bootstrap. Data points with `R` values exceeding `P` are marked as outliers and the waveforms corresponding to outliers are assigned as `unsorted`.

Reference

Pierre Yger et al. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. Elife 2018 Mar 20;7:e34518

Waveform Comparison

This analysis can display:

- Means and standard deviations of the waveform variables in specified time intervals
- Waveforms in the Principal Component space
- Trough and peak values and other features of the waveforms

Parameters

Parameter	Description
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.

continues on next page

Table 87 – continued from previous page

Parameter	Description
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.
What to Draw	Specifies graphics output (means and standard deviations of waveforms, projections in the Principal Component space or trough and peak values).
Draw Variation	An option to draw waveform variation as a colored background (means and standard deviations display).
Variation (standard deviations)	Number of standard deviations in the colored background around waveform mean (means and standard deviations display).
Variation Background Color	The color to be used for drawing of the colored background around waveform mean (means and standard deviations display).
PC X Axis	Which principal component to use for X axis for the Principal Component projections display.
PC Y Axis	Which principal component to use for Y axis for the Principal Component projections display.
Trough/peak X Axis	Which waveform feature to use for X axis for the trough and peak display.
Trough/peak Y Axis	Which waveform feature to use for Y axis for the trough and peak display.
Peak calculation	Use the first local maximum after trough or use maximum with the highest value after trough.
Width at half height baseline	Use the last local maximum before trough as the baseline or use zero as the baseline.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

Column	Description
Variable	Variable name.
YMin	Y Axis minimum.
YMax	Y Axis maximum.
Filter Length	The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds)
Mean Waveform Min	Minimum of the mean waveform.
Mean Waveform Max	Maximum of the mean waveform.

continues on next page

Table 88 – continued from previous page

Column	Description
ProjPc1	Principal component display only. Projection to principal component 1.
ProjPc2	Principal component display only. Projection to principal component 2.
ProjPc3	Principal component display only. Projection to principal component 3.
ProjPc4	Principal component display only. Projection to principal component 4.
TroughTime	Trough/peak display only. Trough time in milliseconds.
TroughValue	Trough/peak display only. Trough value in milliVolts.
PeakTime	Trough/peak display only. Peak time in milliseconds.
PeakValue	Trough/peak display only. Peak value in milliVolts.
TroughToPeakDuration	Trough/peak display only. Distance in time from trough to peak (milliseconds).
WidthAtHalfHeight	Trough/peak display only. Width of the trough at half height (milliseconds).
TroughToPeakSlope	Trough/peak display only. Angle of the line from trough to peak (degrees).

Algorithm

The program selects waveforms in the specified time range and the interval filter(s) and calculates the mean and the standard deviation of the selected waveforms.

Principal components are calculated from all the waveforms of the given waveform variable.

First, the matrix of covariances between waveform points (`c[t, s]`) is calculated:

`c[t, s]` = covariance between vectors `waveform_value[t, *]` and `waveform_value[s,*]`, `s, t = 1, ...,number_of_points_in_each_waveform`.

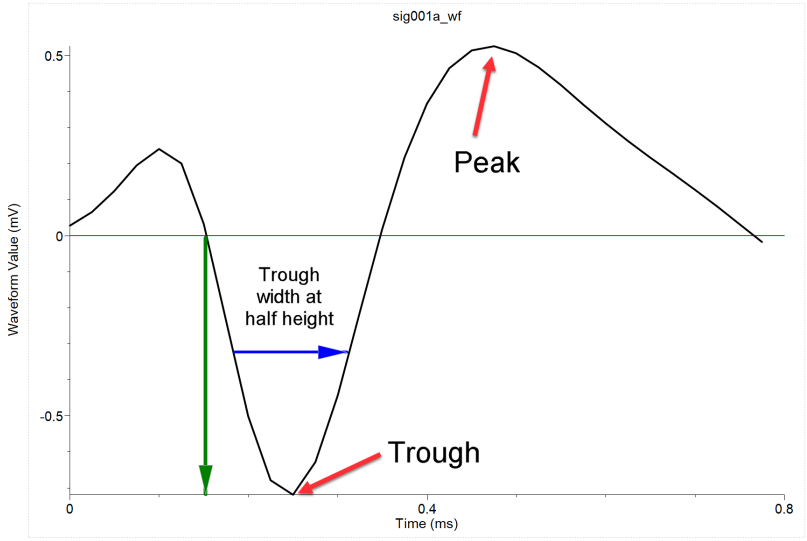
Then, the eigenvalues and eigenvectors are calculated for the matrix `c[t, s]`. The eigenvectors (principal components) are sorted according to their eigenvalues. The first principal component has the largest eigenvalue.

PCA projection display shows scatter plots where x and y are projections of a waveform to the specified principal components (projection is a sum of products `waveform_value[t]*principal_component_value[t]`).

Trough and Peak display

The algorithm assumes that the threshold crossing down was used to detect waveforms.

Therefore, the algorithm expects that there is a trough in the waveform followed by a peak:



The algorithm for calculating trough and peak is the following:

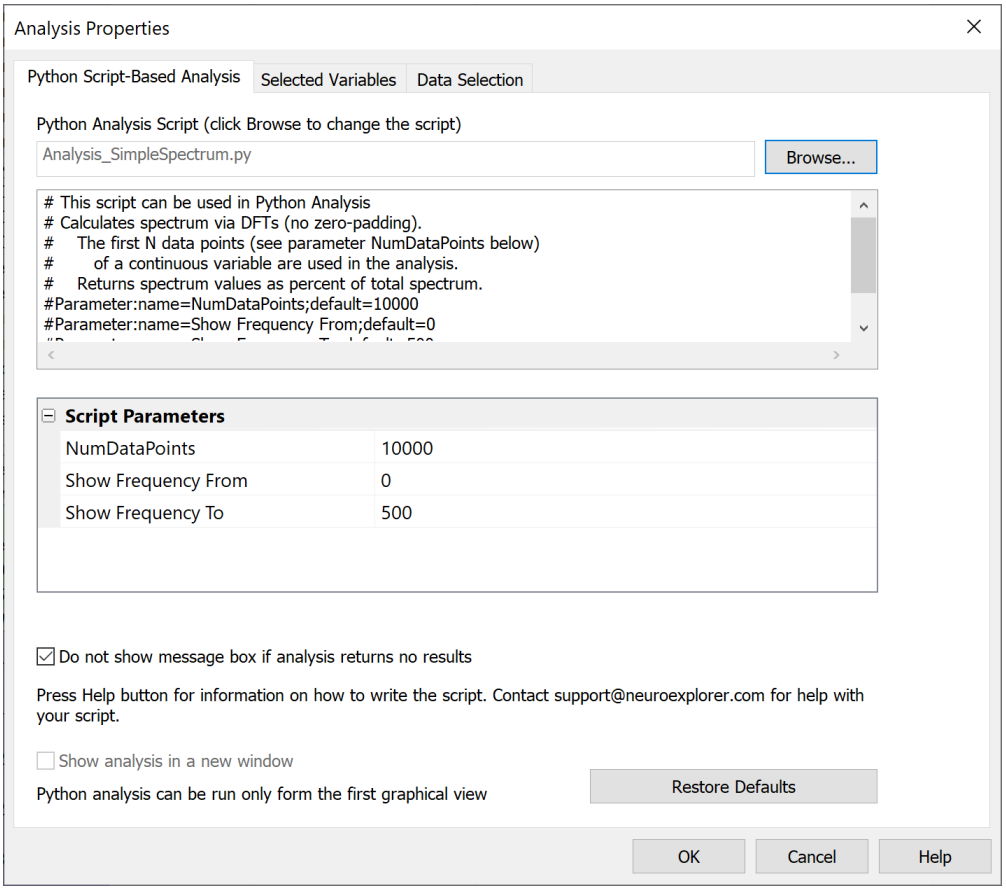
1. Each wave is resampled at 1 microsecond resolution using cubic splines.
2. The local minima of the resampled waveform are found. The local minimum is the waveform point `w[i]` such that `w[i]<w[i-1]` and `w[i]<w[i+1]`.
3. The minimum with the smallest voltage value is the trough.
4. Starting with the trough waveform point, the local maxima of the resampled waveform are found.
5. The maximum with the highest voltage value is the peak.
6. The width at half height is a width of the trough negative going dip when the waveform voltage values are equal to `voltage_at_trough/2` (blue arrow).

Custom Analyses

Python-based Analysis

This analysis uses a Python script to calculate analysis results.

Specify the script using Python analysis dialog:



Parameters

These parameters are shown in the Properties panel, Analysis Properties group.

Parameter	Description
Script Path	Python script path.
Script Parameters as JSON	Script Parameters as JSON string.
Select Data	If Select Data is *From Time Range*, only the data from the specified (by *Select Data From* and *Select Data To* parameters) time range will be used in analysis. See also *Data Selection Options*.
Select Data From	Start of the time range in seconds.
Select Data To	End of the time range in seconds.
Interval filter	Specifies the interval filter(s) that will be used to preselect data before analysis. See also *Data Selection Options*.

Algorithm

For each selected data variable, NeuroExplorer will execute the specified Python script. If you need to execute the script only once for all selected variables, add this line to the top of the script:

```
#Option:callOnce
```

The script can specify script parameters. The specified parameters with be shown in Python Analysis properties dialog.

Script parameter specifications should placed as comments at the top of the script.

Each script parameter specification should be on its own line.

The specification should start with `#Parameter:name=` followed by the name of the parameter, semicolon and the default parameter value, for example:

```
#Parameter:name=NumDataPoints;default=10000
```

If the parameter is used to select a continuous variable, add `;type=ContChoice`:

```
#Parameter:name=ReferenceContChannel;default=LFP01;type=ContChoice
```

The following code retrieves the `NumDataPoints` parameter value that was specified by the user in Python Analysis properties dialog:

```
inputJson = doc.GetPythonAnalysisInput()
inputPars = json.loads(inputJson)
numValues = int(inputPars['ScriptParameters']['NumDataPoints'])
```

Use `doc.GetPythonAnalysisInput()` to get information about the current data variable and to get the values of analysis parameters.

Use `doc.SetPythonAnalysisOutput()` to specify numerical results of the analysis as JSON string.

The results object should be a dictionary with `'XValues'` and `'YValues'` properties containing Python lists of numeric values (see lines with `result['XValues']` and `result['YValues']` in the script below).

Here is a script that calculates a DFT-based spectrum for a specified Continuous variable:

```
# This script can be used in Python Analysis
# Calculates spectrum via DFTs (no zero-padding).
```

(continues on next page)

(continued from previous page)

```
#    The first N data points (see parameter NumDataPoints below)
#       of a continuous variable are used in the analysis.
#    Returns spectrum values as percent of total spectrum.
#Parameter:name=NumDataPoints;default=10000
#Parameter:name=Show Frequency From;default=0
#Parameter:name=Show Frequency To;default=500

import nex
import json
import math

def ReturnEmptyResult(doc):
    result = {}
    result['XAxisLabel'] = 'Frequency (Hz)'
    result['YAxisLabel'] = 'Spectrum (%)'
    result['XValues'] = [0, 1]
    result['YValues'] = [0, 0]
    doc.SetPythonAnalysisOutput(json.dumps(result))

# main calculate function of the script
# called in the last line of the script below
# in Python Analysis, this function is called for each selected variable
def Calculate():
    doc = nex.GetActiveDocument()
    # get the variable info and values of script parameters
    inputJson = doc.GetPythonAnalysisInput()
    if not inputJson:
        raise ValueError('this script should be run from Python Analysis only
↪')

    inputPars = json.loads(inputJson)

    variable = inputPars['Variable']

    if variable['Type'] != 'Continuous':
        ReturnEmptyResult(doc)
        return

    # get parameter values in numeric form
    numValues = int(inputPars['ScriptParameters']['NumDataPoints'])
    freqFrom = float(inputPars['ScriptParameters']['Show Frequency From'])
    freqTo = float(inputPars['ScriptParameters']['Show Frequency To'])
```

(continues on next page)

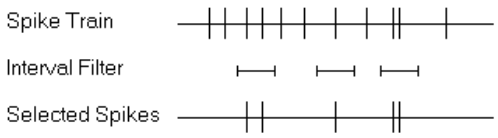
(continued from previous page)

```
    # get continuous values
    v = doc[variable['Name']].ContinuousValues()
    numberOfContValues = len(v)
    if numberOfContValues == 0:
        ReturnEmptyResult(doc)
        return
    if numberOfContValues > numValues:
        v = v[0:numValues]
        numberOfContValues = len(v)

    samplingRate = float(variable['SamplingRate'])
    spectrumStep = samplingRate/numberOfContValues

    # this is the main spectrum calculation
    spectrum = nex.Spectrum(v)

    s = sum(spectrum)

    result = {}
    result['XAxisLabel'] = 'Frequency (Hz)'
    result['YAxisLabel'] = 'Spectrum (%)'
    result['XValues'] = []
    result['YValues'] = []
    for i in range(len(spectrum)):
        freq = i*spectrumStep
        if freq >= freqFrom and freq <= freqTo:
            result['XValues'].append(freq)
            if s == 0:
                result['YValues'].append(0.0)
            else:
                result['YValues'].append(100.0*spectrum[i]/s)

    if len(result['XValues']) == 0:
        ReturnEmptyResult(doc)
    else:
        doc.SetPythonAnalysisOutput(json.dumps(result))


Calculate()
```

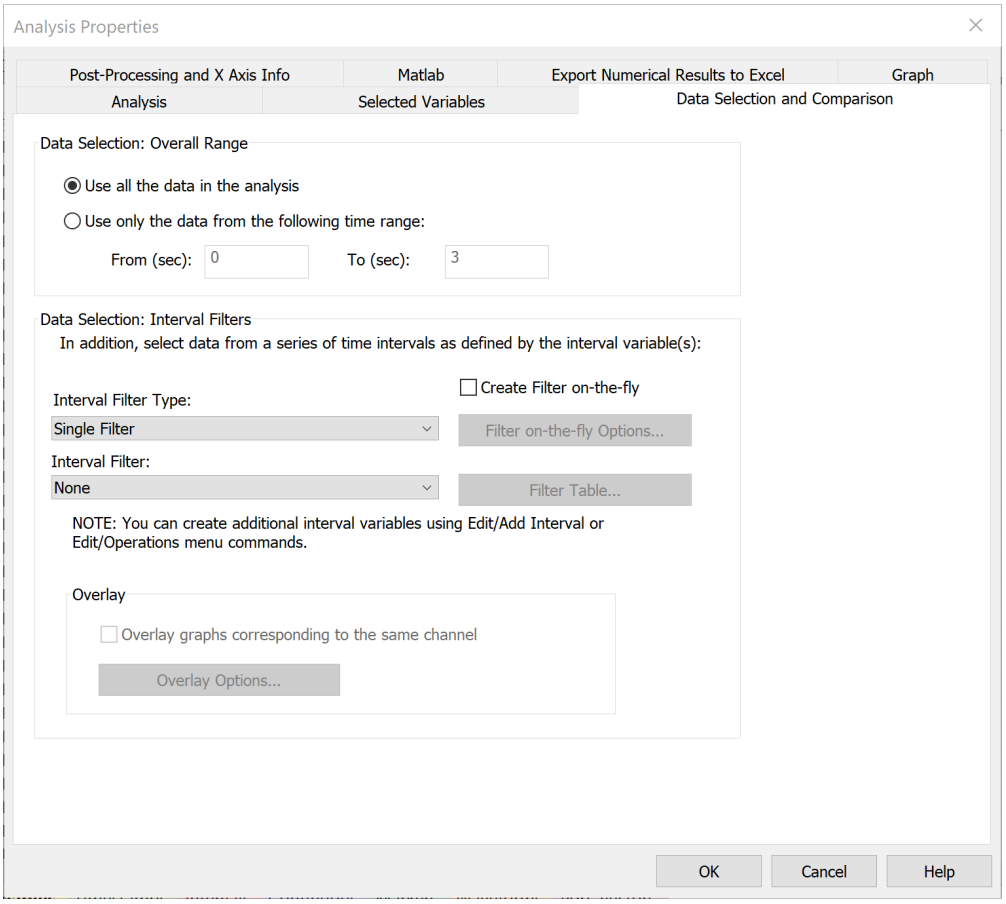
3.1.3 Common Analysis Options

Data Selection Options

Before performing the analysis, NeuroExplorer can select the timestamped events that are inside the specified time intervals:



For example, you may want to analyze only the first 10 minutes of the recording session. To do this, choose the Data Selection tab in the Analysis Parameters dialog, click **Use only the data from the following time range** and specify **From** and **To** parameters in seconds:



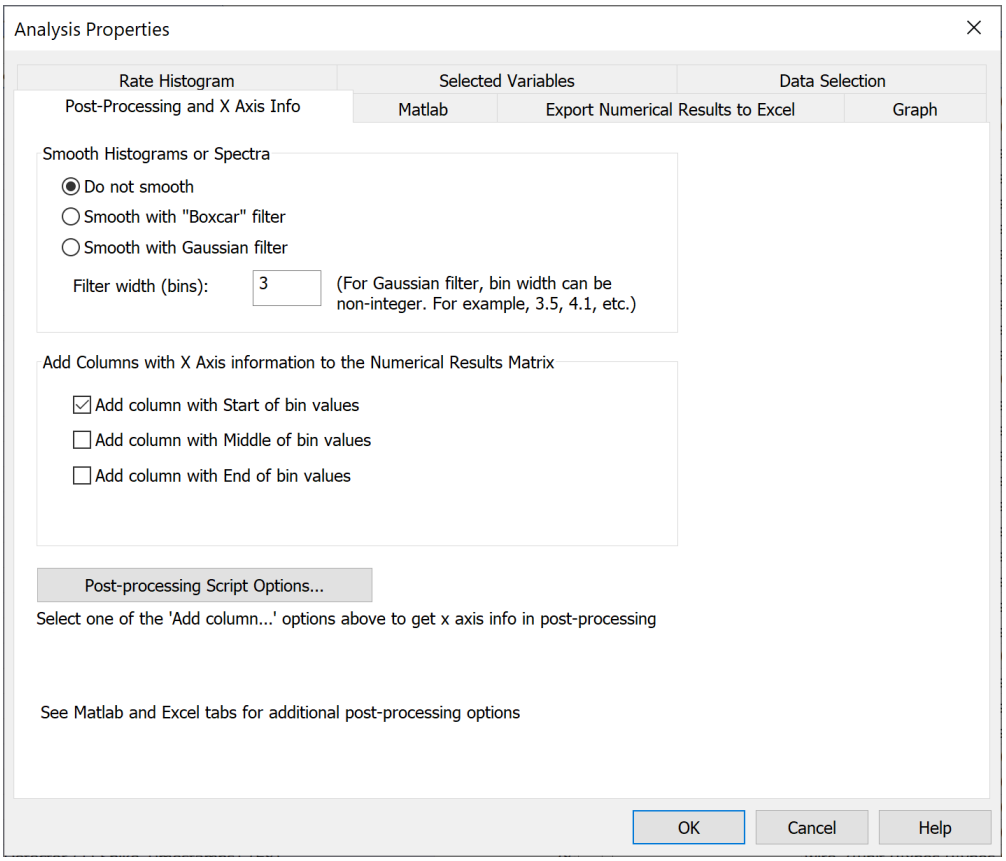
You can also choose an interval variable (at the bottom of the Data Selection page) as an additional data filter. NeuroExplorer then will select data from one or more time intervals as specified by the interval variable.

Some analyses (Perievent Histograms, Perievent Rasters and Crosscorrelograms) allow you to specify several interval filters, so that a different filter will be used for a column or for a

row of graphs.

Post-Processing Options

For the histogram-style analyses, NeuroExplorer has an optional Post-processing analysis step. You can specify post-processing options using **Post-Processing and X Axis Info** tab in the Analysis Parameters dialog.



You can smooth the resulting histogram with Gaussian or Boxcar filters.

The boxcar filter is a filter with equal coefficients. If `f[i]` is i-th filter coefficient and `w` is the filter width, then

```
f[i] = 1.0/w for all i
```

Thus for the filter of width 3, the boxcar filter is

```
f[-1] = 0.33333, f[0] = 0.33333, f[1] = 0.33333.
```

and the smoothed histogram `sh[i]` is calculated as

```
sh[i] = f[-1]*h[i-1] + f[0]*h[i] + f[1]*h[i+1]
```

where `h[i]` is the original histogram.

The boxcar filter needs to include the same number of histogram values before and after the current value. This means that the number of coefficients of the filter cannot be even. If the filter width `w` is specified as an even number, the actual filter width will be `w+1`.

The Gaussian filter is calculated using the following formula:

```
f[i] = exp(-i*i/sigma)/norm, i from -2*d to 2*d
```

where

```
d = ( (int)w + 1 )/2
sigma = -w * w * 0.25 / log(0.5)
norm = sum of exp( -i * i / sigma), i from -2*d to 2*d
```

The parameters of the filter are such that the width of the Gaussian curve at half the height equals to the specified filter width. Please note that for the Gaussian filter, width of the filter (w) can be a non-integer. For example, w can be 3.5.

See *Matlab Options* and *Excel Options* for more information on NeuroExplorer post-processing capabilities.

Matlab Options

NeuroExplorer can interact with Matlab via COM interface using Matlab as a powerful post-processing engine. Immediately after calculating histograms, NeuroExplorer can send the resulting matrix of histograms to Matlab and then ask Matlab to execute any series of Matlab commands.

Use the **Matlab** tab in the Analysis parameters dialog to specify the matrix name and the Matlab command string.

Excel Options

NeuroExplorer can send numerical results directly to Excel. There are two ways to send the results to Excel:

- Use Send to Excel button on the toolbar or menu command **File | Save Numerical Results | Send Results to Excel**
- Use **Excel** tab in the Analysis Parameters dialog to specify what to transfer to Excel and the location of the top-left cell for the data from NeuroExplorer.

Confidence Limits for Perievent Histograms

If the duration of experimental session is `T` (seconds) and we have `N` spikes in the session, then the neuron firing rate `F` (spikes per second) is:

```
F = N/T
```

Several options how to calculate neuron firing rate `F` are available. See **Options** below.

If the spike train is a Poisson train, the probability of the neuron to fire in the small bin of the size `b` (seconds) is

```
P = F*b
```

The **expected** bin count for the perievent histogram is then:

`C = P*NRef`, where `NRef` is the number of the reference events.

The value `C` is used for drawing the Mean Frequency in the Perievent Histograms and Cross- and Autocorrelograms.

The confidence limits for bin counts are calculated using the assumption that the bin count has a Poisson distribution with mean `C`.

Assume that a random variable `S` has a Poisson distribution with parameter (and mean) `C`. Then, the 99% confidence limits are calculated as follows:

```
Low Conf. = x such that Prob(S < x) = 0.005
```

```
High Conf. = y such that Prob(S > y) = 0.005
```

If `C < 30`, NeuroExplorer uses the actual Poisson distribution

`Prob(S = K) = exp(-C) \* (C^K) / K!`, where `C^K` is `C` to the power of `K`,

to calculate the confidence limits.

If `C>= 30.`, the Gaussian approximation of the Poisson distribution is used. The approximation of Poisson distribution with mean `C` is a Gaussian distribution with both the mean and variance equal to `C`.

For example, for 99% confidence limits:

```
Low Conf. = C - 2.58*sqrt(C)
```

```
High Conf.= C + 2.58*sqrt(C)
```

Options

The following options to calculate the neuron firing rate `F` are available:

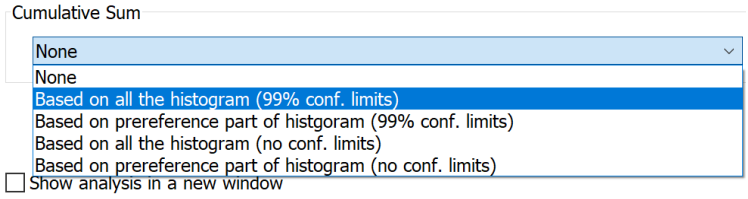
- **Use target and reference timestamps from selected time range and interval filter.** `T` is the length of all the time intervals used in analysis, `N` is the number of spikes within these intervals.
- **Use all target and reference timestamps from the file.** Here `T` is the total length of experimental session, `N` is the total number of spikes for a given neuron.
- **Use target timestamps from the time intervals corresponding to bins before zero.** This option only works for a stimulation-type data. For example, if you stimulate every second and calculate Perievent Histogram with `XMin=-0.2, XMax=0.2`, NeuroExplorer can easily distinguish the spikes before and after the stimulus. However, if you stimulate every 200 ms, the spikes before the second stimulus are also the spikes after the first stimulus, so you cannot distinguish the spikes that should be used to calculate the mean firing rate. The algorithm:
  - For each reference event timestamp `r`, a time interval `(r+XMin, r)` is created (where `XMin` is Perievent Histogram or Crosscorrelogram time axis minimum parameter; `XMin` should be negative).
  - `T` is calculated as the length of all `(r+XMin, r)` intervals that do not overlap.
  - `N` is calculated as the number of spikes in these intervals.
  - If more than 5% of intervals overlap, `F` is set to zero.
- **Use target timestamps from interval filter.** Use this option when you want to calculate the neuron firing rate using spikes from an interval filter that is different from the interval filter specified in the Data Selection page. `T` is the length of all the time intervals of the specified interval filter, `N` is the number of spikes within these intervals.

Reference

Abeles M. Quantification, smoothing, and confidence limits for single-units histograms. Journal of Neuroscience Methods. 5(4):317-25, 1982

Cumulative Sum Graphs

Use Cumulative Sum option in Analysis Parameters dialog to display Cumulative Sum above histogram-type analyses:



Here is the algorithm that is used to draw optional cumulative sum graphs above the histograms.

Suppose we have a histogram with bin counts `bc[i], i=1,...,N`.

Cumulative Sum Graph displays the following values cs[i]:

for bin 1: `cs[1] = bc[1] - A`

for bin 2: `cs[2] = bc[1]+bc[2] - A*2`

for bin 3: `cs[3] = bc[1]+bc[2]+bc[3] - A*3`, etc.

The value of `A` depends on the selected Cumulative Sum option:

`A` is equal to average of all `bc[i]` if you select "Based on all the histogram" option

`A` is equal to average of all `bc[i]` for bins that are before zero on time scale if you select "Based on prereference part of histogram" option.

If you use "Based on all the histogram" option, the value of the cumulative sum for the last bin is always zero:

`cs[N] = bc[1]+bc[2]+...bc[N] - A*N, where A = (bc[1]+bc[2]+...bc[N])/N`.

Shift-Predictor for Crosscorrelograms

Shift-predictor is defined for a series of trials - you take the spikes of one neuron in trial 1 and correlate them with the spikes of another neuron in trial 2, etc.

These “trials” are represented in NeuroExplorer as time intervals, i.e. pairs of numbers:

start of interval 1, end of interval 1

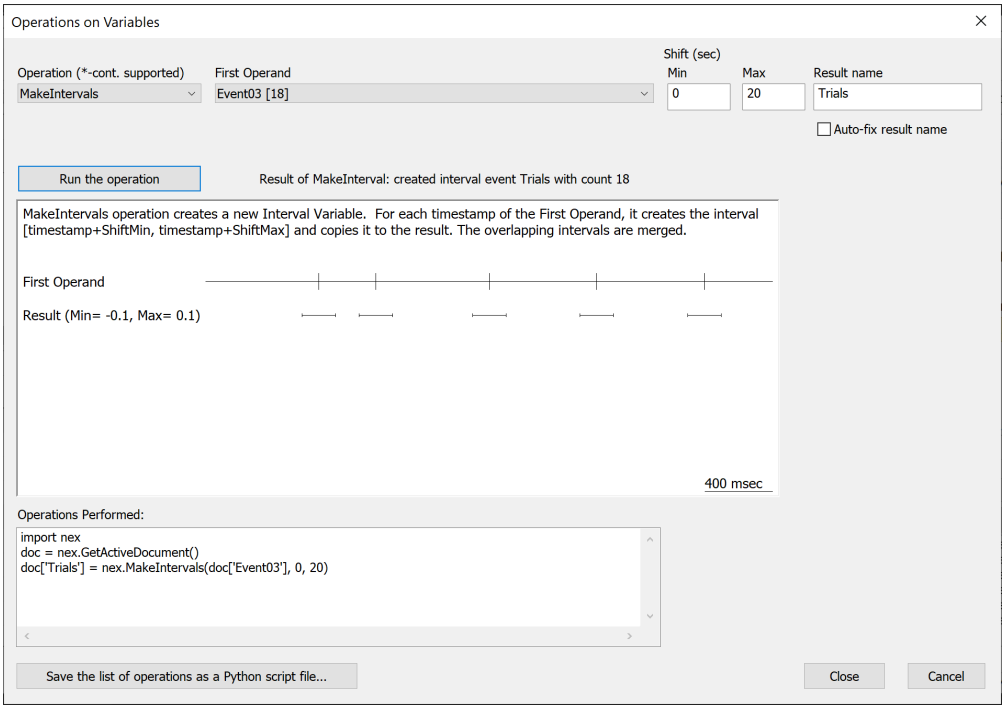
start of interval 2, end of interval 2, etc.

To use the shift-predictor, you need to create those “trials” first.

You need an external event that is fired at the beginning of each trial. Suppose *Event03* is the event that happens at the beginning of each trial and each trial lasts 20 seconds.

To create the trial intervals:

- select **Edit | Operations on Variables** menu command
- in the operations list, select *MakeIntervals*
- in the First Operand, select the external event *Event03*
- set *Shift Min* to 0. and *Shift Max* to 20.0
- in the *New Var Name*, type: *Trials*
- press *Run the Operation* button
- close the dialog.



Now, select *Crosscorrelograms* analysis, go to *Data Selection and Shift-Predictor* tab, select *Trials* as an interval filter and specify other shift-predictor parameters.

Classic Shift-Predictor Algorithm

Suppose we have n trial intervals:

`[trial_1_start, trial_1_end], [trial_2_start, trial_2_end], ..., [trial_n_start, trial_n_end]`

and we have 2 spike trains:

reference spike train with N timestamps `r1, r2, ... , rN`;

and target spike train with M timestamps `s1, s2, ... , sM`.

Shift-predictor for shift = 1 is calculated like this:

Step 1: find all the timestamps `ri` that are inside the first trial interval `[trial_1_start, trial_1_end]`

Step 2: find all the timestamps `sj` that are inside the second trial interval `[trial_2_start, trial_2_end]`

Step 3: shift all the timestamps `ri` so that they align with the second trial:

calculate new timestamps `pi = ri + (trial_2_start - trial_1_start)`

Step 4: calculate a crosscorrelogram between `pi` and `sj`

Repeat Steps 1 through 4 with interval `[trial_2_start, trial_2_end]` in Step 1 and interval `[trial_3_start, trial_3_end]` in Step 2.

...

Repeat Steps 1 through 4 with interval `[trial_n_start, trial_n_end]` in Step 1 and interval `[trial_1_start, trial_1_end]` in Step 2. Note that we wrap around the trials: the last trial is correlated with the first one.

Then, calculate shift-predictors for shift = 2:

Repeat Steps 1 through 4 with interval `[trial_1_start, trial_1_end]` in Step 1 and interval `[trial_3_start, trial_3_end]` in Step 2. That is, shift trials by 2.

...

And, finally, calculate shift-predictors for `shift = Number_of_shifts` specified in the shift-predictor options.

Shift-Predictor With Random Shuffle

When using random shuffle, for each shift, we shuffle the trial numbers. That is, we take trial numbers array `tn = [1, 2, 3, ..., n]` and randomly shuffle this array. If randomly shuffled array is `tn_shuffled`, we correlate trial `i` with the trial `tn_shuffled[i]`.

More on Time Intervals

These time intervals can be used in other analyses in NeuroExplorer to analyze spikes only from those intervals.

For example, you may have a pre-drug period in the experiment (say, from 0 to 600 seconds) and want to analyze the spikes from this pre-drug time period.

To do this, you create a new interval variable (let's call it `PreDrug`) that contains only one interval [0., 600.]. To create a new interval variable, select **Edit | Add Interval Variable...** menu command.

Then you can calculate, for example, the autocorrelograms for the spikes in the pre-drug period by specifying `PreDrug` as an interval filter in the Data Selection page of the Autocorrelogram parameters dialog.

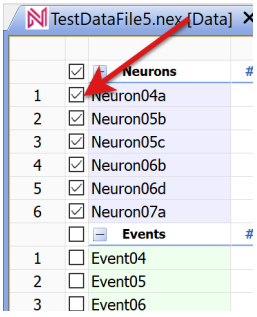
Selecting Variables for Analysis

NeuroExplorer can analyze at once any group of variables in the file. There are several ways to select which variables are analyzed.

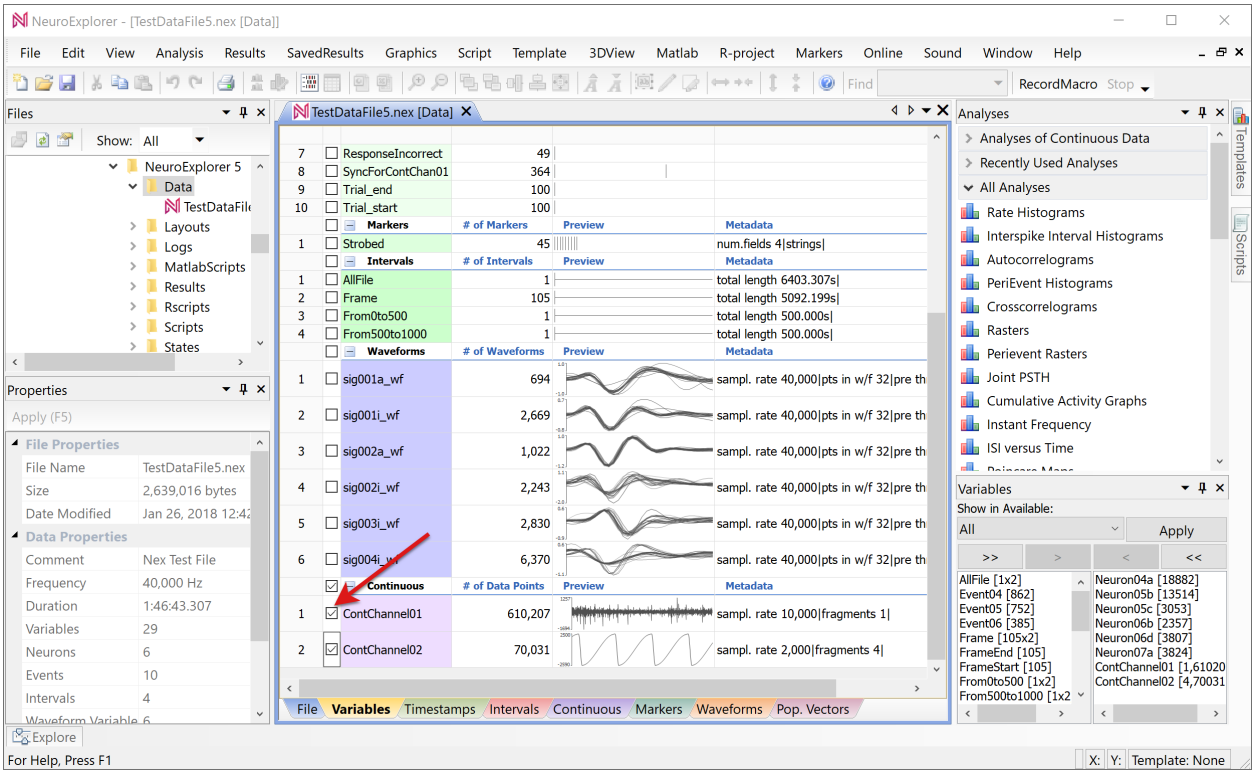
1. Select variables in the Variables window.

The checkbox to the left of a variable name indicates if the variable is selected to be viewed or to be analyzed.

When NeuroExplorer loads a data file, if the Neuron variables are present, NeuroExplorer selects all the Neuron variables:



To select continuous variables (for example, variables ContChannel01 and ContChannel02 in TestDataFile5.nex), click to the left of the channel names:



Note: Variable selection in **Variables Window** is applied only when a **new** Graph Window or 1D View is created.

To change the list of selected variables in existing Graph Window or 1D Viewer window, see options listed below.

2. To change the list of selected variables in existing **1D Viewer** window, use **Analysis | Select Variables** menu command.
3. To change the list of selected variables in existing **Graph** window, use **Selected Variables** tab in the **Analysis Properties** dialog. See *First Continuous Variables Analysis tutorial* for details.

3.2 Supported File Formats

NeuroExplorer can load native data files created by many popular data acquisition systems (see table below).

Note: If your data file format is not listed in the table below, contact NeuroExplorer technical support (support@neuroexplorer.com).

Data acquisition system	Files with these extensions that can be opened in NeuroExplorer
3Brain	.brw, .bxr
Alpha Omega Engineering	.isi, .lsm, .map, .mat, .mpx, .nda
Amplipex	.dat (use File | Import Data | Binary File… menu command)
Axion Biosystems	spk, .raw
Axon Instruments	.abf
Binary file with continuous data	(no specified extension, use File | Import Data | Binary File… menu command)
Blackrock Neurotech	.nev, .ns1, .ns2, .ns3, .ns4, .ns5, .ns6, .ns7, .ns8, .ns9 (saved using file specification 2.X)
CED Spike-2	.smr, .smrx
Cortex	(no specified extension, use File | Import Data | Cortex Data File… menu command)
DataWave Technologies	.act, .cut, .dat, .edt, .uff
European Data Format	.edf, .bdf
g.Tec Medical Engineering g.Recorder	.hdf5
Intan Technologies	.rhd, .rhs (time.dat, amplifier.dat and other .dat files are loaded automatically when .rhd or .rhs file is open
KlustaSuite spike sorting software	.kwik
MED64 (Panasonic, Alpha MED Scientific)	.csv, .modat
Multichannel Systems	.mcd, .msrd
Neuralynx	.dat, .nev, .ncs, .nse, .nst, .nts, .ntt, .nvt, .t
NeuroExplorer	.nex, .nex5
Open ePhys	.continuous, .events, .spikes
Plexon	.ddt, .pl2, .plx
RC Electronics	.prm
Ripple Neuro	.nev, .nf1, .nf2, .nf3, .nf4, .nf5, .nf6, .nf7, .nf8, .nf9
Thomas Recording WAV file	.wav
Tucker-Davis Technologies	.sev, .tbk, .tdx, .tev, .tin, .tnt, .tsq
Text file with timestamps	.txt
Text file with continuous data	.txt

The table below lists supported file extensions in alphabetical order.

File extension	Default file format
.abf	Axon Instruments .abf File
.act	DataWave File
.bdf	Biosemi .bdf File
.brw	3Brain .brw File
.bxr	3Brain .bxr File
.continuous	Open ePhys .continuous File
.csv	MED64 .csv File
.cut	DataWave File
.dat	Text File (Cont. Data with Timestamps)
.ddt	Plexon .ddt File
.edf	European Data Format .edf File
.edt	DataWave File
.events	Open ePhys .events File
.hdf5	g.Recorder .hdf5 File
.isi	Alpha Omega .isi File
.kwik	KlustaSuite .kwik File
.lsm	Alpha Omega .lsm File
.map	Alpha Omega File
.mat	Alpha Omega Matlab .nda File
.mcd	Multichannel Systems .mcd File
.modat	MED64 .modat File
.mpx	Alpha Omega File
.msrd	Multichannel Systems .msrd File
.ncs	Neuralynx Cont. Recording File (.ncs)
.nda	Alpha Omega .nda File
.nev	Blackrock Neurotech .nev File
.nex	NeuroExplorer .nex File
.nex5	NeuroExplorer .nex5 File
.nf1	Ripple .nfX File
.nf2	Ripple .nfX File
.nf3	Ripple .nfX File
.nf4	Ripple .nfX File
.nf5	Ripple .nfX File
.nf6	Ripple .nfX File
.nf7	Ripple .nfX File
.nf8	Ripple .nfX File
.nf9	Ripple .nfX File
.ns1	Blackrock .nsX File
.ns2	Blackrock .nsX File
.ns3	Blackrock .nsX File
.ns4	Blackrock .nsX File
.ns5	Blackrock .nsX File

continues on next page

Table 91 – continued from previous page

File extension	Default file format
.ns6	Blackrock .nsX File
.ns7	Blackrock .nsX File
.ns8	Blackrock .nsX File
.ns9	Blackrock .nsX File
.nse	Neuralynx Single Electrode File (.nse)
.nst	Neuralynx Stereotrode File (.nst)
.nts	Neuralynx Timestamp File (.nts)
.ntt	Neuralynx Tetrode File (.ntt)
.nvt	Neuralynx Video Tracker File (.nvt)
.pl2	Plexon .pl2 File
.plx	Plexon .plx File
.prm	RCE 12-bit File
.raw	Axion .raw File
.rhd	Intan Technologies .rhd File
.rhs	Intan Technologies .rhs File
.sev	TDT data File
.smr	CED .smr File
.smrx	CED .smrx File
.spikes	Open ePhys .spikes File
.spk	Axion .spk File
.str	Stranger .str File
.t	Neuralynx .t File
.tbk	TDT data File
.tdx	TDT data File
.tev	TDT data File
.tin	TDT data File
.tnt	TDT data File
.tsq	TDT data File
.txt	Text File (matrix of timestamps)
.uff	DataWave File
.wav	WAV File

Note: NeuroExplorer tries to identify the data file type from the file extension.

Some file extensions are used by several file formats and NeuroExplorer may be unable to identify what is the file format of the file that is being loaded.

For example, a .nev file can be either a Neuralynx data file or Ripple Neuro data file.

You can specify what file format corresponds to a specific file extension. See *Dealing with Unknown File Format Error*.

3.3 Scripting Reference

3.3.1 Introduction to Scripting in NeuroExplorer

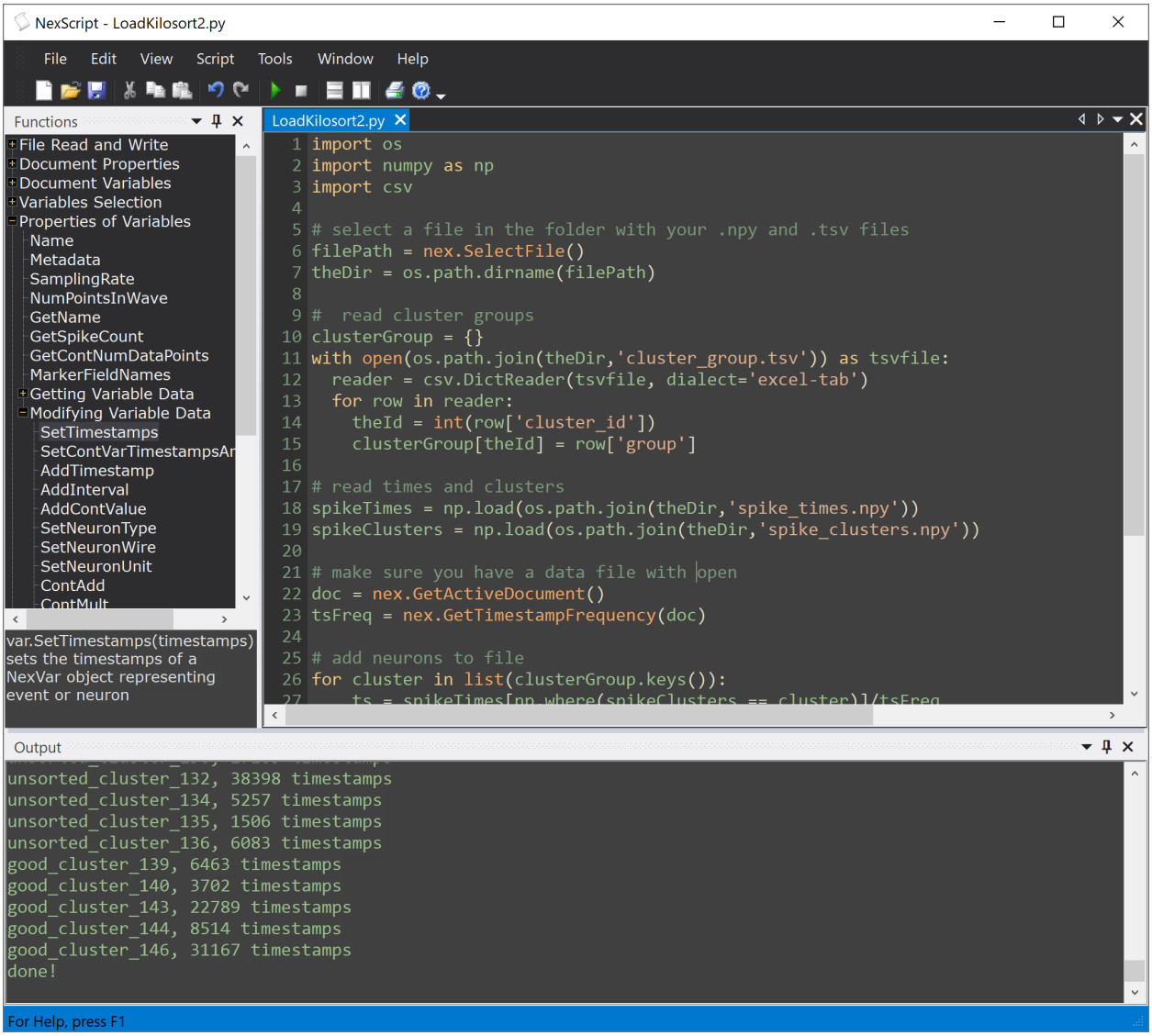
NeuroExplorer scripts can be written using a powerful and well-documented Python language.

NeuroExplorer uses Python 2.7.17 and Python 3.7.8 and there is no need to install Python – all the Python files needed for scripting are installed by NeuroExplorer setup program.

A legacy scripting language NexScript is also supported. However, many new new scripting capabilities in NeuroExplorer can only be used from Python. It is recommended that you use Python for all the new scripts.

NexScript Editor

To create or edit script, use **Script | New Script** or **Script | Open Script** NeuroExplorer menu commands. NeuroExplorer will open NexScript editor:



There are three main panels in NexScript: the main *Editor* panel (upper right), the *Functions* panel (upper left) and *Output* panel (bottom).

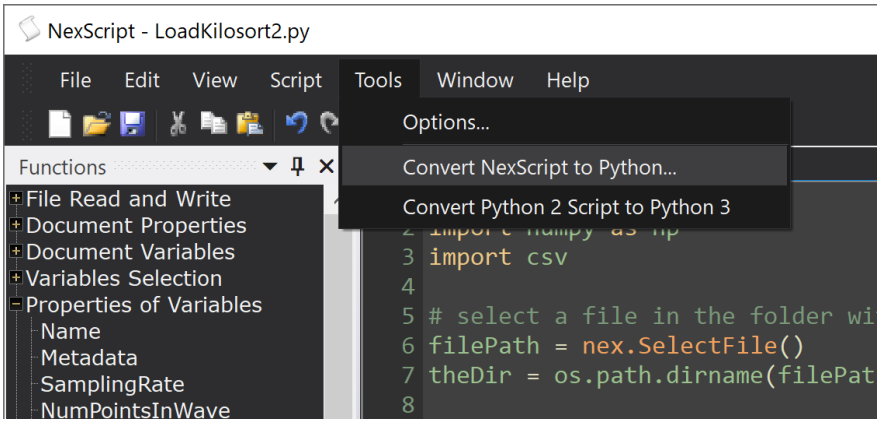
The *Functions* panel lists NeuroExplorer custom functions that allow you to load and modify data, run analyses and save results.

The output of `print()` Python statements and `Trace()` NexScript statements is shown in the *Output* panel.

To create and run a new script:

- Select **File | New** menu command
- Add code to the new script
- Save the new script using **File | Save As** menu command
- Run the script by selecting **Script | Start** menu command

You can convert your existing NexScript script to Python using **Tools | Convert NexScript to Python** menu command:



Using External Python Development Environment

You can run NeuroExplorer Python scripts using a Python development environment of your choice. For example, Visual Studio Code and Spyder are excellent free Python development environments with debugging and auto-completion.

To use an external Python environment:

- In NeuroExplorer, select **Script | Enable Running Python Scripts in External Editor** menu command
- Install `nex` Python package. Run this command in Windows Command Prompt

```
python.exe -m pip install -U nex
```

- If you get an error message `pip: command not found`, run this command first to install pip:

```
python.exe -m ensurepip --upgrade
```

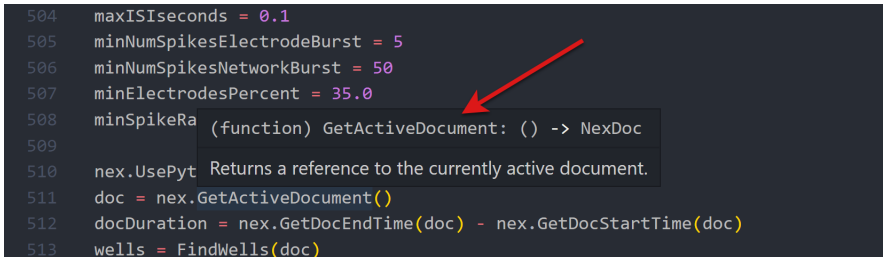
- If you are using Anaconda:
  - Type `Anaconda Prompt` in Windows Search box
  - Select a version of Anaconda Prompt in Best Match panel above Windows Search box
  - Run this command in Anaconda Prompt

```
pip install -U nex
```

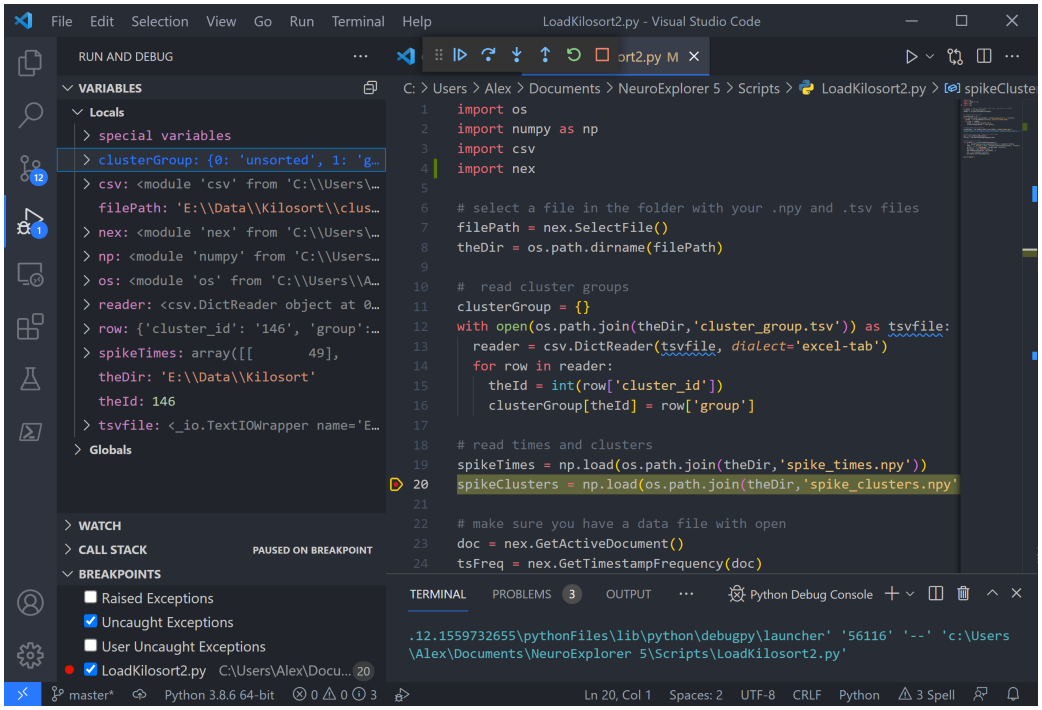
No changes in your script are required.

If you are using Visual Studio Code:

- Open Python script in Visual Studio Code or, in NexScript, select menu command **File | Open in VS Code**
- In Visual Studio Code, you will see full IntelliSense support (code completion, parameter info, etc.) for nex package:



- Press F5 to debug your script in Visual Studio Code:



Using Anaconda Python Distribution

NeuroExplorer comes with a standard Python 3 (version 3.7) installation that includes thousands of functions and packages. In addition to standard packages, NeuroExplorer distribution also includes numpy and h5py.

If you need advanced data analysis methods (advanced statistics, clustering, etc.), you can use a free scientific Python distribution Anaconda with NeuroExplorer.

To use Anaconda with NeuroExplorer:

- Install Anaconda for Python 3

- In NeuroExplorer, select **Script | Enable Running Python Scripts in External Editor** menu command
- Type `Anaconda Prompt` in Windows Search box
- Select a version of Anaconda Prompt in Best Match panel above Windows Search box
- Run this command in Anaconda Prompt

```
pip install -U nex
```

You can now open Anaconda Navigator and select Visual Studio Code or Spyder to run your scripts.

Python Language

Python is a very popular programming language. You can easily find Python tutorials on Internet.

For example, here is a very short Python tutorial that we can recommend:

https://cs231n.github.io/python-numpy-tutorial/#python

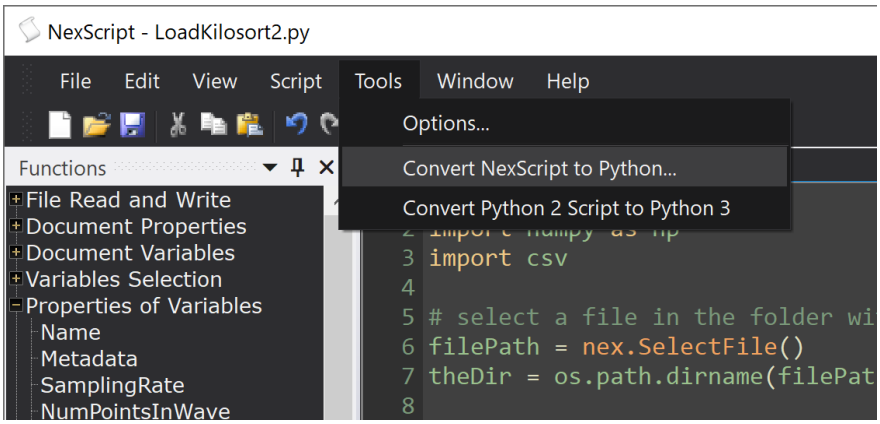
In your Python script, you can call more than 200 functions to access data and analysis functionality of NeuroExplorer. See *Scripting Reference* for details.

NeuroExplorer uses Python 2.7.17 and Python 3.7.8 and there is no need to install Python – all the Python files needed for scripting are installed by NeuroExplorer setup program.

We recommend that you use Python 3 for your scripts since Python 2 is being deprecated.

To specify Python version for your scripts, select **Script | Python Options** menu command and select Python version.

You can convert your existing Python 2 script to Python 3 using **Tools | Convert Python 2 Script to Python 3** menu command in *NexScript Editor*:



Windows File Paths in Python

Windows file paths may contain backslashes, for example *"C:\Data\file1.nex5"*.

To use a path with the basklashes in **Python** (for example, to open the file *"C:\Data\file1.nex5"*), you can either prepend the file path with r

```
doc = nex.OpenDocument(r"C:\Data\file1.nex5")
```

or replace single backslashes with double backslashes

```
doc = nex.OpenDocument("C:\\Data\\file1.nex5")
```

or replace single backslashes with forward slashes

```
doc = nex.OpenDocument("C:/Data/file1.nex5")
```

NexScript Language

We recommend to use Python when developing new scripts.

NeuroExplorer also has a less powerful (than Python) internal scripting language NexScript. NexScript syntax resembles the Matlab language.

Lines and Comments in NexScript

Each line of the script may contain only one statement, for example:

```
x = 0
```

If the statement is very long, you can use the line continuation symbol (backslash \) to indicate that the next line contains the continuation of the current line. For example, instead of:

```
Dialog(doc, "D:\Experiment101\data\mydata\*.nex5", "Filter", "string")
```

you can write:

```
Dialog(doc, "D:\Experiment101\data\mydata\*.plx",\
"Filter"", "string")
```

The percent symbol (%) marks the beginning of a comment in NexScript, for example:

```
% this is a comment line
x = 0 % this is also a comment
```

Variable Names

The variable name in NexScript should begin with a letter and contain only letters, digits and the underscore sign.

The following names are valid:

```
Neuron01   Bar_press  Nr_1a    SPK02b
```

These variable names cannot be used in NexScript:

```
2Neuron   Bar-press
```

The variables from the opened data file can be accessed using the document reference with the variable name specified in the square brackets. For example, the spike train `SPK01a` from the active document can be addressed as:

```
doc["SPK01a"]
```

where `doc` is a reference to the document. You can get this reference by calling `GetActiveDocument()` or calling `OpenDocument()`.

An alternative way to access file variable is by getting a reference to the variable:

```
SPK01a = GetVarByName(doc, "SPK01a")
```

Variable Types

NexScript supports numeric variables:

```
xmean = (xmax - xmin)/2.
```

and strings:

```
name = "SPK" + "01"
```

A variable can also be a reference to the existing variable in the file:

```
neuron1 = GetVar(doc, 1, "neuron")
```

or a reference to the opened document:

```
doc = GetActiveDocument()
```

A variable type can be changed if the right-hand side of the assignment has a different type. For example:

```
x = 0.005 % x now is a numeric variable
x = "SPK" % after this statement, x is a string
```

Global Variables

A variable can be declared global so that it can be accessed from several scripts:

```
Global name
```

`Global` statements should be placed at the beginning of the script.

Assignments Involving File Variables

If the left-hand side of the assignment is not a file variable:

```
v = doc["Neuron01"]
```

the script variable `v` will contain a reference (a shortcut) to the file variable `Neuron01` and `Neuron01` spike times will not be copied to `v`.

If the left-hand side of the assignment (`doc["ContVarCopy"]` below) is a file variable:

```
doc["ContVarCopy"] = doc["ContChannel01"]
```

all the data of the file variable `ContChannel01` will be copied to the new file variable `ContVarCopy`.

For example, to create a new continuous variable containing rectified signal of continuous variable `FP01`, we can write the following code:

```
doc["FP01_Rectified"] = NewContVarWithFloats(doc, 10000)
for i=1 to GetContNumDataPoints(doc["FP01"])
    t = doc["FP01"][i,1]
    v = abs(doc["FP01"][i,2])
    AddContValue(doc["FP01_Rectified"], t, v)
end
```

We can achieve the same result using variable references and copying data to a new variable (and the code below will run more than 7 times faster than the code above):

```
doc["FP01_Rectified"] = doc["FP01"]
rectified = doc["FP01_Rectified"]
ContVarStoreValuesAsFloats( rectified )
```

(continues on next page)

(continued from previous page)

```
n = GetContNumDataPoints( rectified )
for i=1 to n
   rectified[i,2] = abs( rectified[i,2] )
end
```

Access to the Data in File Variables

Timestamped Variables

You can access the timestamp value by specifying the timestamp index (1-based, i.e. the first timestamp has index 1) in square brackets:

```
doc = GetActiveDocument()
% first option: use doc["VarName"] notation
timestamp = doc["SPK01a"][3]
% second option: get variable by name and then use the variable directly
spikeTrain = doc["SPK01a"]
timestamp = spikeTrain[3]
```

You can assign a new value for any timestamp in the current file. For example, to assign the value 0.5 (sec) to the third timestamp of the variable `SPK01a`, you can use this script:

```
doc = GetActiveDocument()
doc["SPK01a"][3] = 0.5
```

You can also add timestamps to a variable using NexScript functions. See *Modifying Variable Data Functions* for details.

Interval Variables

`IntVar[i,1]` gives you read/write access to the start of the i-th interval, `IntVar[i,2]` gives you read/write access to the end of the i-th interval.

For example, to assign the value 27.5 seconds to the end of the first interval of interval variable `Frame`, you would use this script:

```
doc = GetActiveDocument()
doc["Frame"][1,2] = 27.5
```

You can also add intervals to an interval variable using NexScript functions. See *Modifying Variable Data Functions* for details.

Continuous Variables

`ContVar[i,1]` gives you read-only access to the timestamp of the i-th data point.

`ContVar[i,2]` gives you read-write access to the value of the i-th data point.

For example, the following script prints the timestamp and the value of the fifth data point in variable `ContChannel01`:

```
Trace("ts = ", doc["ContChannel01"][5,1], "value =" ,doc["ContChannel01"][5,
↪2])
```

The following script line assigns the value of 100 to the fifth data point:

```
doc["ContChannel01"][5,2] = 100.0
```

Note that when you assign a value to a continuous variable that was imported from a file created by a data acquisition system, the assigned value might be clipped.

Data acquisition systems usually store continuous values as 2-byte integers. NeuroExplorer also stores imported analog data as 2-byte integers with some scaling factors.

For example, if analog to digital converter has input range from -1000mV to +1000mV, then maximum 2-byte value 32767 corresponds to 1000mV and the scaling factor is 1000/32768. If we try to assign the value that is outside the original input range, the value has to be clipped so that it could be stored as a 2-byte integer. For example, if we try to assign 2000mV, the stored value will be 32767 or 1000mV.

To avoid data clipping, use `ContVarStoreValuesAsFloats()` function to convert 2-byte integer storage to 4-byte float storage that does not have clipping problems.

Marker Variables

`MarkerVar[i,1]` gives you read-only access to the timestamp of the i-th marker.

`MarkerVar[i,2]` gives you read-only access to the value of the first field of the i-th marker. Note that marker field values are stored as strings, so you will need to use `StrToNum()` function to convert these values to numbers.

For example, the following script prints the timestamp and the first field value of the fifth marker in variable `Strobed`:

```
Trace("ts = ", doc["Strobed"][5,1], "marker value =" ,doc["Strobed"]5,2])
```

Expressions

Standard algebraic expressions are supported:

```
xmean = (xmax - xmin)/2.
```

Addition operation can also be applied to the strings:

```
name = "SPK" + "01"
```

Logical expressions may be used in `if` and `while` statements:

```
x = 2

if x >= 2
    Trace("x is greater or equal to 2")
end

if x > 2
    Trace("x > 2")
end

if x <= 2
    Trace("x <= 2")
end

if x == 2
   Trace("x equals 2")
end

if x <> 1
   Trace("x is not equal to 1")
end
```

Logical expressions may be combined using logical **AND** (`&`) or logical **OR** (`|`) operators:

```
if (x >= 2) & (y <4)
    Trace("x <=2 and y <4")
end

if (x >= 2) | (y <4)
    Trace("x <=2 or y <4")
end
```

Flow Control in NexScript

Loops

NexScript supports two types of loops: `for` loops and `while` loops.

`for` loop has the following syntax:

```
for variable = expression to expression
   statements ...
end
```

Example:

```
for i = 1 to 10
    SelectVar(doc, i, "neuron")
end
```

`while` loop has the following syntax:

```
while logical_expression
   statements ...
end
```

Example:

```
i = 1
while i < 10
    SelectVar(doc, i, "neuron")
    i = i + 1
end
```

break

`break` statement causes an immediate exit from the loop.

The following loop

```
for i = 1 to 10
   Trace(i)
   if i == 5
      break
   end
end
```

will produce output: 1 2 3 4 5

continue

`continue` statement returns to the loop's beginning skipping the statements that follow it.

The following loop

```
for i = 1 to 5
   if i == 3
      continue
   end
   Trace(i)
end
```

will produce output: 1 2 4 5

return

`return` statement causes an immediate exit from the script.

Conditional operators

Operator `if` has the following syntax:

```
if logical_expression
   statements ...
end
```

or

```
if logical_expression
   statements ...
else
   statements ...
end
```

Example

```
% select a variable if it has at lest one spike
% otherwise, deselect the variable
```

(continues on next page)

(continued from previous page)

```
if GetVarCount(doc, i, "neuron") > 0
    SelectVar(doc, i, "neuron")
else
    DeselectVar(doc, i, "neuron")
end
```

3.3.2 File Read and Write

GetActiveDocument

Returns a reference to the currently active document.

Syntax

```
GetActiveDocument()
```

Return

Returns a reference to the opened document (`nex.NexDoc` object in Python, integer in NexScript). The function returns zero (invalid document reference) if there are no data files open.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
```

NexScript

```
doc = GetActiveDocument()
```

OpenDocument

Opens a data file with the specified path. Returns a reference to the opened document.

Syntax

```
OpenDocument(filePath)
```

Parameters

Parameter	Type	Description
filePath	string	Full path of the file

Return

Returns a reference to the opened document (`nex.NexDoc` object in Python, integer in NexScript). The function returns invalid document reference if OpenDocument operation failed.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
# This function can be used to open all supported data file formats
doc1 = nex.OpenDocument(r"C:\Data\File1.nex5")
```

(continues on next page)

(continued from previous page)

```
doc2 = nex.OpenDocument(r"C:\Data\File1.nex5")
doc3 = nex.OpenDocument(r"C:\Data\File3.msrd")
```

NexScript

```
% This function can be used to open all supported data file formats
doc1 = OpenDocument("C:\Data\File1.nex5")
doc2 = OpenDocument("C:\Data\File1.nex5")
doc3 = OpenDocument("C:\Data\File3.msrd")
```

Note: Before opening a text file, specify text file import options using **View/Options** menu command.

NewDocument

Creates a new document (data file) with the specified timestamp frequency.

Syntax

```
NewDocument(timestampFrequency)
```

Parameters

Parameter	Type	Description
timestampFrequency	number	Timestamp frequency

Return

Returns a reference to the opened document (`nex.NexDoc` object in Python, integer in NexScript).

Examples

Python

```
import nex
doc = nex.NewDocument(100000.)
```

NexScript

```
doc = NewDocument(25000.)
```

CloseDocument

Closes the specified document.

Syntax

```
CloseDocument(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Examples

Python

```
import nex
# this script prints the number of variables in the file
doc = nex.GetActiveDocument()
print("document contains {} variables".format(nex.GetVarCount(doc,"all")))
nex.CloseDocument(doc)
```

NexScript

```
% this script prints the number of variables in the file
doc = GetActiveDocument()
Trace("document contains", GetVarCount(doc, "all"), "variables")
CloseDocument(doc)
```

SaveDocument

Saves the specified document.

Syntax

```
SaveDocument(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Note: By default, the document is saved as a .nex file. The document **cannot** be saved as .nex file if:

- the file size is more than 1.5 GB
- the document contains continuous channels with values stored as floats
- some of the timestamps (in time ticks) cannot be represented as 32-bit integers (the timestamp values are greater than 2.1 billion)

If the document cannot be saved as .nex file, the document is saved as .nex5 file.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SaveDocument(doc)
```

NexScript

```
doc = GetActiveDocument()
SaveDocument(doc)
```

SaveDocumentAs

Saves the data (in .nex, .nex5 or .edf format) in a file with the specified file path.

Syntax

```
SaveDocumentAs(doc, filePath)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
filePath	string	File path

Return

None.

Note: If filePath has extension '.nex', the document is saved using .nex format. If filePath has extension '.nex5', the document is saved using .nex5 format.

If filePath has extension '.edf', continuous variables of the document that have no gaps in recording (the number of fragments is 1) as well as the marker variables are saved in the .edf (European Data Format) file.

NeuroExplorer uses **edflib** library (https://www.teuniz.net/edflib/) to load and save data in EDF files. The library license can be found in the file

```
C:\Program Files\Nex Technologies\NeuroExplorer 5\EDFLIB_LICENSE.txt
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# save file in .nex5 format
nex.SaveDocumentAs(doc, r"C:\Data\file1.nex5")
```

NexScript

```
doc = GetActiveDocument()
SaveDocumentAs(doc, "C:\Data\file1.nex")
```

SaveAsTextFile

Saves the document data in the text file with the specified file name.

Syntax

```
SaveAsTextFile(doc, filePath)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
filePath	string	Text file path.

Return

None.

Note: This function uses options that were specified the last time the menu command **File | Export Data | As Text File** was executed.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SaveAsTextFile(doc, "C:\\Data\\dataFromNex.txt")
```

NexScript

```
doc = GetActiveDocument()
SaveAsTextFile(doc, "C:\Data\dataFromNex.txt")
```

MergeFiles

Opens and merges the specified files, returns the reference to the merged file.

Syntax

```
MergeFiles(filename_list, gap)
```

Parameters

Parameter	Type	Description
filename_list	string	The list of files to be merged. File names should be separated by commas
gap	number	Gap between the files (in seconds). See Note below.

Return

Returns the reference to the merged file.

Note: In the merge process, the data (for the same variable) from the second file is appended to the data from the first file. Before appending, the timestamps of the variables in the second file are shifted by the value equal to *duration_of_the_first_file + gap*.

Since the function uses commas as separators between file names, it is assumed that each file name does not contain commas.

Examples

Python

```
import nex
import json
dataDir = "C:\\Data\\"
files = []
files.append(dataDir + "file1.nex5")
files.append(dataDir + "file2.nex5")
doc = nex.MergeFiles(json.dumps(files), 1.)
```

NexScript

```
dataDir = "C:\Data\"
file1 = dataDir + "file1.nex5"
file2 = dataDir + "file2.nex5"
mergeList = file1 + "," + file2
doc = MergeFiles(mergeList, 1.)
```

3.3.3 File Selection

GetFileCount

Returns the number of files that match the file filter.

Syntax

```
GetFileCount(fileFilter)
```

Parameters

Parameter	Type	Description
fileFilter	string	File filter specification. Can contain wildcards (*). For example, to get all .nex files in folder *C:\Data*, use filter *C:\Data*.nex*

Return

Returns the number of files that match file filter.

Note: In Python, use `os.listdir(directory)` function. See example code below.

Examples

The following code will repeat analysis for all .nex files in the folder C:\Data.

Python

```
import nex
import os
directory = r"C:\Data"
for name in os.listdir(directory):
    if name.endswith(".nex"):
        fullPath = os.path.join(directory, name)
        doc = nex.OpenDocument(fullPath)
        if doc:
            nex.SelectAllEvents(doc)
            # run the analysis, save numerical results and close the file
            nex.ApplyTemplate(doc, "Autocorrelograms")
            nex.SaveNumResults(doc, fullPath + '.txt')
            nex.CloseDocument(doc)
```

NexScript

```
% repeat analysis for all .nex files in the folder
filefilter = "C:\Data1\*.nex"
n = GetFileCount(filefilter)
for i=1 to n
    name = GetFileName(i)
    doc = OpenDocument(name)
    if doc > 0
        % run the analysis, save numerical results and close the file
        ApplyTemplate(doc, "Autocorrelograms")
        SaveNumResults(doc, name + ".txt")
        CloseDocument(doc)
    end
end
```

GetFileName

Returns the file name for the specified index after `GetFileCount()` was called.

Syntax

```
GetFileName(index)
```

Parameters

Parameter	Type	Description
index	number	Index of the file in the file list created by the last call to `GetFileCount()`

Return

Returns the file name (the full path of the file) for the specified index after `GetFileCount()` was called.

Note: In Python, use `os.listdir(directory)` function. See example code below.

Examples

Python

```
import nex
import os
directory = r"C:\Data"
for name in os.listdir(directory):
    if name.endswith(".nex"):
        fullPath = os.path.join(directory, name)
        doc = nex.OpenDocument(fullPath)
        if doc:
            nex.SelectAllEvents(doc)
            # run the analysis, save numerical results and close the file
            nex.ApplyTemplate(doc, "Autocorrelograms")
            nex.SaveNumResults(doc, fullPath + '.txt')
            nex.CloseDocument(doc)
```

NexScript

```
% repeat analysis for all .nex files in the folder
filefilter = "C:\Data1\*.nex"
n = GetFileCount(filefilter)
for i=1 to n
    name = GetFileName(i)
    doc = OpenDocument(name)
    if doc > 0
        % run the analysis, save numerical results and close the file
        ApplyTemplate(doc, "Autocorrelograms")
        SaveNumResults(doc, name + ".txt")
        CloseDocument(doc)
```

(continues on next page)

(continued from previous page)

```
    end
end
```

SelectFile

Opens File Open dialog and returns the path of the file selected in the dialog.

Syntax

```
SelectFile()
```

Return

Returns the path of the file selected in File Open dialog.

Examples

Python

```
import nex
path = nex.SelectFile()
# if the user pressed Cancel in the file dialog, the returned path is empty
if len(path) > 0:
    # open file for reading
    file = open(path, "r")
    # read the first line of the file
    line = file.readline()
    print(line)
    file.close()
```

NexScript

```
path = SelectFile()
% if the user pressed Cancel in the file dialog, the returned path is empty
if StrLength(path) > 0
  % open file for reading
  file = OpenFile(path, "r")
  line = ""
  % read the first line of the file
  ReadLine(file, line)
  Trace(line)
  CloseFile(file)
end
```

SelectFiles

Opens File Open dialog with multiple selection option and returns the list of file paths selected in the dialog. To select multiple files in the dialog, use Ctrl+Mouse_Click or Shift+Mouse_Click

Syntax

```
SelectFiles(initialDirectory, extension)
```

Parameters

Parameter	Type	Description
initialDirectory	string	Optional parameter that specifies initial directory for File Open dialog
extension	string	Optional parameter that specifies file extension for File Open dialog.

Return

Returns the list of file paths selected in the File Open dialog. If the user pressed Cancel in File Open dialog, the returned list is empty.

Examples

Python

```
import nex
# use default initial directory and show all files in dialog
files = nex.SelectFiles()
# use default initial directory and show only .nex5 files in dialog
files = nex.SelectFiles(extension='nex5')
# open files in directory 'C:/MyData' and show only .nex5 files in dialog
files = nex.SelectFiles(initialDirectory='C:/MyData', extension='nex5')

# run the same analysis (Detect Spikes) on all selected files
if files:
   for filePath in files:
      doc = nex.OpenDocument(filePath)
      nex.DeselectAll(doc)
      # select all continuous variables
      contNames = doc.ContinuousNames()
      for name in contNames:
         nex.Select(doc, doc[name])
      # run DetectSpikes analysis
      nex.ApplyTemplate(doc, 'Default\\DetectSpikes')
      # save results in new .nex5 file
      newName = os.path.splitext(filePath)[0] + "_spikes.nex5"
      nex.SaveDocumentAs(doc, newName)
      nex.CloseDocument(doc)
```

3.3.4 Reading and Writing Result Files

SaveNumResults

Saves the numerical results to a text file with the specified name.

Syntax

```
SaveNumResults(doc, fileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
fileName	string	File path for saved results

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveNumResults(doc, "C:\\Data\\res1.txt")
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
ApplyTemplate(doc, "Autocorrelograms")
SaveNumResults(doc, "C:\Data\res1.txt")
```

SaveNumSummary

Saves the summary of numerical results to a text file with the specified name.

Syntax

```
SaveNumSummary(doc, filename)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
filename	string	File path for saved results

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
```

(continues on next page)

(continued from previous page)

```
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveNumSummary(doc, "C:\\Data\\res1summary.txt")
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
ApplyTemplate(doc, "Autocorrelograms")
SaveNumSummary(doc, "C:\Data\res1summary.txt")
```

OpenSavedResult

Opens a saved result file with the specified path.

Syntax

```
OpenSavedResult(string filePath)
```

Parameters

Parameter	Type	Description
filePath	string	A full path of a saved result (.nexresult) file or analysis template (.ntp) file. Use `SaveResults|Restore Analysis in NeuroExplorer` menu command to restore analysis saved in the result file. You can also use the script commands saved in history.py file.

Return

None.

Note: Analysis template files can be used in two functions: `ApplyTemplate` and `OpenSavedResult`. Here are the differences:

- The template file used in `OpenSavedResult` needs to have the data file path and the list of selected variables specified within the template file.
- `ApplyTemplate` command ignores the data file path and the list of selected variables specified within the template file.

Examples

Python

```
import nex
nex.OpenSavedResult(r"C:\Data\File1_nex Autocorrelograms 001.nexresult")
```

NexScript

```
OpenSavedResult("C:\Data\File1_nex Autocorrelograms 001.nexresult")
```

3.3.5 Reading and Writing Binary and Text Files

OpenFile

Opens text file using the specified mode, returns file ID.

Syntax

```
OpenFile(filePath, mode)
```

Parameters

Parameter	Type	Description
filePath	string	Full path of the file.
mode	string	File open mode. Can be either 'r', 'rt' (read), 'w' or 'wt' (write) or 'a' or 'at' (append)

Return

Returns file ID.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
# open a file in read mode
f = open("C:\\Data\\parameters.txt", "r")
# read all the lines in the file and print them
if f:
    lines = f.readlines()
    f.close()
    for line in lines:
        print(line.rstrip()) # rstrip() removes end-of-line character from␣
↪line
```

NexScript

```
% open a file in read mode
file = OpenFile("C:\Data\parameters.txt", "r")
% read all the lines in the file and print them
if file > 0
    line = "" % make line a string variable
    while ReadLine(file, line) > 0
        Trace(line)
    end
    CloseFile(file)
end
```

CloseFile

Closes the specified file.

Syntax

```
CloseFile(fileID)
```

Parameters

Parameter	Type	Description
fileID	number	File ID received from `OpenFile` function

Return

None.

Note: In Python, use Python native file `close()` function. See Python code below.

Examples

Python

```
# open a file in read mode
f = open("C:\\Data\\parameters.txt", "r")
# read all the lines in the file and print them
if f:
    lines = f.readlines()
    f.close()
    for line in lines:
        print(line.rstrip()) # rstrip() removes end-of-line character from␣
↪line
```

NexScript

```
% open a file in read mode
file = OpenFile("C:\Data\parameters.txt", "r")
% read all the lines in the file and print them
if file > 0
    line = " " % make line a string variable
    while ReadLine(file, line) > 0
        Trace(line)
    end
    CloseFile(file)
end
```

ReadLine

Reads a line from the text file.

Syntax

```
ReadLine(fileID, lineString)
```

Parameters

Parameter	Type	Description
fileID	number	File ID received from `OpenFile` function.
lineString	string	String that receives the text from the file

Return

Returns 1 if more text to read is available in the file, otherwise, returns 0.

Note: In Python, use Python native file `readlines()` method. See Python code below.

Examples

Python

```
# open a file in read mode
f = open("C:\\Data\\parameters.txt", "r")
# read all the lines in the file and print them
if f:
    lines = f.readlines()
    f.close()
    for line in lines:
        print(line.rstrip()) # rstrip() removes end-of-line character from␣
↪line
```

NexScript

```
% open a file in read mode
file = OpenFile("C:\Data\parameters.txt", "r")
% read all the lines in the file and print them
if file > 0
    line = " " % make line a string variable
    while ReadLine(file, line) > 0
        Trace(line)
    end
```

(continues on next page)

(continued from previous page)

```
    CloseFile(file)
end
```

WriteLine

Writes a line of text to a text file.

Syntax

```
WriteLine(fileID, lineString)
```

Parameters

Parameter	Type	Description
fileID	number	File ID received from `OpenFile` function
lineString	string	The string to be written to the file.

Return

None.

Note: In Python, use file object write() method. See Python code below.

Examples

Python

```
# open a file in write mode
file = open("C:\\Data\\results.txt", "w")
file.write("first line\n")
file.write("second line\n")
file.close()
```

NexScript

```
% open a file in write mode
fileID = OpenFile("C:\Data\results.txt", "w")
WriteLine(fileID, "first line")
CloseFile(fileID)
```

ReadBinary

Reads a binary value of a specified type from a file.

Syntax

```
ReadBinary(fileID, valueType)
```

Parameters

Parameter	Type	Description
fileID	number	File ID received from `OpenFile` function.
valueType	string	Binary type. Should be `char`, `uchar`, `short`, `ushort`, `int`, `uint`, `int64`, `uint64`, `float` or `number`

Return

The value read from the file.

Examples

Python

```
import nex
# open binary file in read mode
file = nex.OpenFile(r"C:\Data\binaryfile.dat", "r")
# read short (2-byte signed) value
```

(continues on next page)

(continued from previous page)

```
shortValue = nex.ReadBinary(file, "short")
nex.CloseFile(file)
```

NexScript

```
% open binary file in read mode
file = OpenFile("C:\Data\binaryfile.dat", "r")
% read short (2-byte signed) value
shortValue = ReadBinary(file, "short")
CloseFile(file)
```

FileSeek

Repositions file pointer by the specified offset.

Syntax

```
FileSeek(fileID, offset, type)
```

Parameters

Parameter	Type	Description
fileID	number	File ID received from `OpenFile` function.
offset	number	Number of bytes to move the file pointer.
type	string	Pointer movement mode. Should be 'begin', 'current' or 'end'

Return

Returns the new byte offset from the beginning of the file.

Note: FileSeek(file, 0, "end") returns file size in bytes. FileSeek(file, 0, "current") returns the current file position.

In Python, you may want to use Python native file.seek() https://www.tutorialspoint.com/python/file_seek.htm .

Examples

Python

```
import nex
# open binary file in read mode
file = nex.OpenFile("C:\\Data\\binaryfile.dat", "r")
# get file length
fileLength = nex.FileSeek(file, 0, "end")
# move pointer 4 bytes from the beginning of the file
newPosition = nex.FileSeek(file, 4, "begin")
# read short (2-byte signed) value
shortValue = nex.ReadBinary(file, "short")
nex.CloseFile(file)
```

NexScript

```
% open binary file in read mode
file = OpenFile("C:\Data\binaryfile.dat", "r")
% get file length
fileLength = FileSeek(file, 0, "end")
% move pointer 4 bytes from the beginning of the file
newPosition = FileSeek(file, 4, "begin")
% read short (2-byte signed) value
shortValue = ReadBinary(file, "short")
CloseFile(file)
```

3.3.6 Document Properties

GetDocPath

Returns the full path of the data file.

Syntax

```
GetDocPath(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the full path of the data file.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
path = nex.GetDocPath(doc)
```

NexScript

```
doc = GetActiveDocument()
path = GetDocPath(doc)
```

GetDocTitle

Returns the data file name.

Syntax

```
GetDocTitle(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the data file name. For example, if the document has the path "C:\Data\data1.nex", this function will return "data1.nex"

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
fileName = nex.GetDocTitle(doc)
```

NexScript

```
doc = GetActiveDocument()
fileName = GetDocTitle(doc)
```

GetDocComment

Returns the document comment string.

Syntax

```
GetDocComment(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the document comment string.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
comment = nex.GetDocComment(doc)
print(comment)
```

NexScript

```
doc = GetActiveDocument()
comment = GetDocComment(doc)
Trace(comment)
```

GetDocStartTime

Returns the minimum timestamp value (in seconds) for all the document variables.

Syntax

```
GetDocStartTime(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the minimum timestamp value (in seconds) for all the document variables.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
startTime = nex.GetDocStartTime(doc)
```

NexScript

```
doc = GetActiveDocument()
startTime = GetDocStartTime(doc)
```

SetDocStartTime

Sets the start of experimental session (in seconds) for the document.

Syntax

```
SetDocStartTime(doc, start_time)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
start_time	number	Start time in seconds

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
start_time = 0.1
nex.SetDocStartTime(doc, start_time)
```

NexScript

```
doc = GetActiveDocument()
start_time = 0.1
SetDocStartTime(doc, start_time)
```

GetDocEndTime

Returns the maximum timestamp value (in seconds) for all the document variables.

Syntax

```
GetDocEndTime(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the maximum timestamp value (in seconds) for all the document variables.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
endTime = nex.GetDocEndTime(doc)
```

NexScript

```
doc = GetActiveDocument()
endTime = GetDocEndTime(doc)
```

SetDocEndTime

Sets the length of experimental session for the document.

Syntax

```
SetDocEndTime(doc, end_time)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
end_time	number	Document end time in seconds

Return

None.

Note: NeuroExplorer determines document end time when the document is loaded. Then, the duration of experimental session (end_time - start_time) may be used in the calculations of the confidence limits. If your script modifies or adds the variables, you may need to set the document end time directly to maintain the correctness of the confidence calculations.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SetDocEndTime(doc, 2200.3)
```

NexScript

```
doc = GetActiveDocument()
SetDocEndTime(doc, 2200.3)
```

GetTimestampFrequency

Returns the frequency used in the internal representation of the timestamps.

Syntax

```
GetTimestampFrequency(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the frequency (in Hertz) used in the specified file to store the timestamps.

Note: Internally, the timestamps are stored as 64-bit integers representing the number of time ticks from the start of the experiment. The time tick is equal to `1./Timestamp_Frequency`.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
tsFreq = nex.GetTimestampFrequency(doc)
```

NexScript

```
doc = GetActiveDocument()
tsFreq = GetTimestampFrequency(doc)
```

SetDocTimestampFrequency

Returns the frequency (in Hertz) used in the specified file to store the timestamps.

Syntax

```
SetDocTimestampFrequency(doc, freq)
```

Parameters

Parameter	Type	Description
doc	document reference	Reference to the document.
freq	number	Timestamp frequency in Hz.

Note: Internally, the timestamps are stored as 64-bit integers representing the number of time ticks from the start of the experiment. The time tick is equal to `1./Timestamp_Frequency`.

Return

None.

Examples

Python

```
import nex
doc = nex.NewDocument(20000)
nex.SetDocTimestampFrequency(doc, 33000.33)
```

NexScript

```
doc = GetActiveDocument()
SetDocTimestampFrequency(doc, 33000.33)
```

GetRecordingStartTimeString

If recording start time for the document is available, this function returns the string representing recording start time in ISO 8601 format. Python only.

Syntax

```
GetRecordingStartTimeString(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the string representing recording start time. Use `datetime.strptime(sts, '%Y-%m-%dT%H:%M:%S.%f')` to convert to Python datetime object.

Examples

Python

```
import nex
from datetime import datetime

doc = nex.GetActiveDocument()
startTimeString = nex.GetRecordingStartTimeString(doc)
dt = datetime.strptime(startTimeString, "%Y-%m-%dT%H:%M:%S.%f")
print(dt)
```

SetRecordingStartTime

Sets the recording start time. Python only.

Syntax

```
SetRecordingStartTime(doc, dateTimeString)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
dateTimeString	string	String representing recording start time in ISO 8601 format

Notes

Use `myDateTime.isoformat()` to convert Python datetime object to ISO 8601 string.

Examples

Python

```
import nex
from datetime import datetime

doc = nex.GetActiveDocument()
# set recording start time to current time
now = datetime.now()
nex.SetRecordingStartTime(doc, now.isoformat())
```

3.3.7 Document Variables

GetVarCount

Returns the number of variables of the specified type in the file.

Syntax

```
GetVarCount(doc, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
varType	string	Variable type. Should be 'neuron', 'neuronorevent' , 'event', 'interval' 'wave', 'popvector', 'continuous', 'marker' or 'all'

Return

Returns the number of variables of the specified type in the file.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the number of continuous variables
numContVars = nex.GetVarCount(doc, "continuous")
```

NexScript

```
doc = GetActiveDocument()
% get the number of continuous variables
numContVars = GetVarCount(doc, "continuous")
```

GetVar

Returns the reference to the specified variable.

Syntax

```
GetVar(doc, varNumber, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

continues on next page

Table 123 – continued from previous page

Parameter	Type	Description
varNumber	number	1-based variable number within the group specified by varType.
varType	string	Variable type. Should be 'neuron', 'neuronorevent', 'event', 'interval' 'wave', 'popvector', 'continuous', 'marker' or 'all'

Return

Returns the reference to the specified variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the second event variable
event = nex.GetVar(doc, 2, "event")
```

NexScript

```
doc = GetActiveDocument()
% get the second event variable
event = GetVar(doc, 2, "event")
```

GetVarName

Returns the name of the specified variable.

Syntax

```
GetVarName(doc, varNumber, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
varNumber	number	1-based variable number within the group specified by varType.
varType	string	Variable type. Should be 'neuron', 'neuronorevent' , 'event', 'interval' 'wave', 'popvector', 'continuous', 'marker' or 'all'

Return

Returns the name of the specified variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the name of the first event variable
name = nex.GetVarName(doc, 1, "event")
```

NexScript

```
doc = GetActiveDocument()
% get the name of the first event variable
name = GetVarName(doc, 1, "event")
```

GetVarByName

Returns the reference to the variable which has the specified name.

Syntax

```
GetVarByName(doc, name)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
name	string	Variable name

Return

Returns the reference to the variable which has the specified name.

Note: You can also use `doc[name]` notation to get the variable by name. For example, instead of

```
start = nex.GetVarByName(doc, "TrialStart")
```

you can use

```
start = doc["TrialStart"]
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the variable with the name TrialStart
start = nex.GetVarByName(doc, "TrialStart")
```

NexScript

```
doc = GetActiveDocument()
% get the variable with the name TrialStart
start = GetVarByName(doc, "TrialStart")
```

SetNeuronType

Changes the type of the specified timestamped variable.

Syntax

```
SetNeuronType(doc, var, type)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
var	variableReference	Reference to the variable
type	number	If type is positive, the variable type is set to 'neuron', if type is zero or negative, the variable type is set to 'event'

Return

None.

Note: Neuron and event types are almost identical. The main difference is that when a data file is opened by NeuroExplorer, all the neuron variables in this file are selected for analysis.

You may need to use this function when creating new neuron variables using `NewEvent()` function. `NewEvent` creates an event variable and the variable type can later be changed to neuron using `SetNeuronType()` function.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["neuron1"] = nex.NewEvent(doc, 0)
nex.SetNeuronType(doc, doc["neuron1"], 1)
```

NexScript

```
doc = GetActiveDocument()
doc["neuron1"] = NewEvent(doc, 0)
SetNeuronType(doc, doc["neuron1"], 1)
```

NeuronNames

Returns the list of neuron names in the document.

Syntax

```
doc.NeuronNames()
```

Return

Returns the list of neuron names in the document.

Note: You can also use global function `nex.NeuronNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
namesOfNeurons = nex.NeuronNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
namesOfNeurons = doc.NeuronNames()
```

EventNames

Returns the list of event variable names in the document.

Syntax

```
doc.EventNames()
```

Return

Returns the list of event variable names in the document.

Note: You can also use global function `nex.EventNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
namesOfEvents = nex.EventNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
namesOfEvents = doc.EventNames()
```

IntervalNames

Returns the list of interval variable names in the document.

Syntax

```
doc.IntervalNames()
```

Return

Returns the list of interval variable names in the document.

Note: You can also use global function `nex.IntervalNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
namesOfIntervalVars = nex.IntervalNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
namesOfIntervalVars = doc.IntervalNames()
```

WaveNames

Returns the list of waveform variable names in the document.

Syntax

```
doc.WaveNames()
```

Return

Returns the list of waveform variable names in the document.

Note: You can also use global function `nex.WaveNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
namesOfWaveformVars = nex.WaveNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
namesOfWaveformVars = doc.WaveNames()
```

ContinuousNames

Returns the list of continuous variable names in the document.

Syntax

```
doc.ContinuousNames()
```

Return

Returns the list of continuous variable names in the document.

Note: You can also use global function `nex.ContinuousNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
namesOfContinuousVars = nex.ContinuousNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
namesOfContinuousVars = doc.ContinuousNames()
```

MarkerNames

Returns the list of marker variable names in the document.

Syntax

```
doc.MarkerNames()
```

Return

Returns the list of marker variable names in the document.

Note: You can also use global function `nex.MarkerNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
namesOfMarkerVars = nex.MarkerNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
namesOfMarkerVars = doc.MarkerNames()
```

NeuronVars

Returns the list of neuron variables in the document.

Syntax

```
doc.NeuronVars()
```

Return

Returns the list of neuron variables in the document.

Note: You can also use global function `nex.NeuronVars(doc)`:

```
import nex
doc = nex.GetActiveDocument()
neurons = nex.NeuronVars(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
neurons = doc.NeuronVars()
```

EventVars

Returns the list of event variables in the document.

Syntax

```
doc.EventVars()
```

Return

Returns the list of event variables in the document.

Note: You can also use global function `nex.EventVars(doc)`:

```
import nex
doc = nex.GetActiveDocument()
events = nex.EventVars(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
events = doc.EventVars()
```

IntervalVars

Returns the list of interval variables in the document.

Syntax

```
doc.IntervalVars()
```

Return

Returns the list of interval variables in the document.

Note: You can also use global function `nex.IntervalVars(doc)`:

```
import nex
doc = nex.GetActiveDocument()
intervals = nex.IntervalVars(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
intervals = doc.IntervalVars()
```

WaveVars

Returns the list of waveform variables in the document.

Syntax

```
doc.WaveVars()
```

Return

Returns the list of waveform variables in the document.

Note: You can also use global function `nex.WaveVars(doc)`:

```
import nex
doc = nex.GetActiveDocument()
waveformVariables = nex.WaveVars(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
waveformVariables = doc.WaveVars()
```

ContinuousVars

Returns the list of continuous variables in the document.

Syntax

```
doc.ContinuousVars()
```

Return

Returns the list of continuous variables in the document.

Note: You can also use global function `nex.ContinuousVars(doc)`:

```
import nex
doc = nex.GetActiveDocument()
contVars = nex.ContinuousVars(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
contVars = doc.ContinuousVars()
```

MarkerVars

Returns the list of marker variables in the document.

Syntax

```
doc.MarkerVars()
```

Return

Returns the list of marker variables in the document.

Note: You can also use global function `nex.MarkerVars(doc)`:

```
import nex
doc = nex.GetActiveDocument()
markers = nex.MarkerVars(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
markers = doc.MarkerVars()
```

3.3.8 Creating New Variables

NewEvent

Creates a new timestamped variable.

Syntax

```
NewEvent(doc, count)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
count	number	Initial number of the timestamps in the variable

Return

Returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
temp = nex.NewEvent(doc, 10) # creates a script-only variable
doc["NewVar"] = nex.NewEvent(doc, 0) # creates a new variable in the file
```

NexScript

```
doc = GetActiveDocument()
temp = NewEvent(doc, 10) % creates a script-only variable
doc["NewVar"] = NewEvent(doc, 0) % creates a new variable in the file
```

NewIntEvent

Creates a new interval variable.

Syntax

```
NewIntEvent(doc, count)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
count	number	Initial number of intervals in the variable. This parameter is optional

Return

Returns a reference to the new variable.

Note: If initial number of intervals is positive, intervals are initialized with values [0, 0.5], [1, 1.5], etc.

Examples

The following scripts create a new interval variable that has two intervals: from 0 to 100 seconds and from 200 to 300 seconds.

Python

```
import nex
doc = nex.GetActiveDocument()
doc["MyInterval"] = nex.NewIntEvent(doc)
nex.AddInterval(doc["MyInterval"], 0., 100.)
nex.AddInterval(doc["MyInterval"], 200., 300.)
```

NexScript

```
doc = GetActiveDocument()
doc["MyInterval"] = NewIntEvent(doc)
AddInterval(doc["MyInterval"], 0., 100.)
AddInterval(doc["MyInterval"], 200., 300.)
```

NewContVar

Creates a new continuous variable.

Syntax

```
NewContVar(doc, frequency, mVmin, mVmax)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
frequency	number	Specifies the sampling frequency of the new variable (in Hz)

continues on next page

Table 129 – continued from previous page

Parameter	Type	Description
mVmin	number	Minimum of the values of the new variable (in milliVolts)
mVmax	number	Maximum of the values of the new variable (in milliVolts)

Return

Returns a reference to the new variable.

Note: When you use this function, NeuroExplorer will store the values of the new continuous variable as scaled 2-byte integers. Specifying minimum and maximum of the variable values helps to determine the correct scaling factor for the new variable.

It is recommended that you use `nex.NewContVarWithFloats` function instead of this function.

Examples

Python

```
import nex
import math
doc = nex.GetActiveDocument()
freq = 1000.
# create new variable in the file
doc["SinValues"] = nex.NewContVar(doc, freq,  - 500., 500.)
# add the values to the new variable
for i in range(1000):
    # timestamp
    ts = i / freq
    # value
    value = 500. * math.sin(ts)
    nex.AddContValue(doc["SinValues"], ts, value)
```

NexScript

```
doc = GetActiveDocument()
freq = 1000.
% create new variable in the file
doc["SinValues"] = NewContVar(doc, freq, -500.,500.)
% add the values to the new variable
for i = 1 to 10000
    % timestamp
    ts = i/freq
    % value
    value = 500.*sin(ts)
    AddContValue(doc["SinValues"], ts, value)
end
```

NewContVarWithFloats

Creates a new continuous variable, returns a reference to the new variable. Frequency specifies the sampling frequency of the new variable (in Hz).

Syntax

```
NewContVarWithFloats(doc, frequency)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
frequency	number	Sampling frequency of the new variable (in Hz)

Return

Creates a new continuous variable, returns a reference to the new variable.

Examples

Python

```
import nex
import math
doc = nex.GetActiveDocument()
freq = 1000.
# create new variable in the file
doc["SinValues"] = nex.NewContVarWithFloats(doc, freq)
# add the values to the new variable
for i in range(1000):
    # timestamp
    ts = i / freq
    # value
    value = 500. * math.sin(ts)
    nex.AddContValue(doc["SinValues"], ts, value)
```

NexScript

```
doc = GetActiveDocument()
freq = 1000.
% create new variable in the file
doc["SinValues"] = NewContVarWithFloats(doc, freq)
% add the values to the new variable
for i = 1 to 10000
    % timestamp
    ts = i/freq
    % value
    value = 500.*sin(ts)
    AddContValue(doc["SinValues"], ts, value)
end
```

CreateWaveformVariable

Creates a new waveform variable with specified values.

Syntax

```
doc.CreateWaveformVariable(name, waveformSamplingRate, timestamps,␣
↪waveformValues)
```

Parameters

Parameter	Type	Description
name	string	The name of the variable
waveformSamplingRate	number	Waveform sampling rate in Hz (sampling rate of the continuous channel that was used to extract waveforms)
timestamps	list	List of timestamps in seconds
waveformValues	list of lists	List of waveform values in milliVolts. Each sub-list represents a single waveform (values should be in milliVolts). See sample code below.

Return

None.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# create waveform variable with 2 waveforms:
# waveform 1: timestamp = 10s; waveform values are 2, 3, 4 and 1 (mV)
```

(continues on next page)

(continued from previous page)

```
# waveform 2: timestamp = 20s; waveform values are 5, 6, 7 and 2 (mV)
doc.CreateWaveformVariable('ScriptGenerated', 40000, [10,20], [ [2,3,4,1], [5,
↪6,7,2] ] )
```

NewPopVector

Creates a new population vector.

Syntax

```
NewPopVector(doc, type)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
type	number	If **type** = 0, all weights of the new vector are equal to zero. If **type** = 1, NeuroExplorer creates a vector representing the average of all neurons in the file. If **type** = 2, NeuroExplorer creates a vector representing the sum of all neurons in the file. If **type** = 3, NeuroExplorer creates a vector representing the average of all selected neurons and events in the file. If **type** = 4, NeuroExplorer creates a vector representing the sum of all selected neurons and events in the file.

Return

Returns a reference to the new vector.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["neuronAverage"] = nex.NewPopVector(doc, 1)
```

NexScript

```
doc = GetActiveDocument()
doc["neuronAverage"] = NewPopVector(doc, 1)
```

3.3.9 Deleting Variables

DeleteVar

Deletes the specified variable from the file.

Syntax

```
DeleteVar(doc, number, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
number	number	1-based variable number within the group specified by varType
varType	string	Variable type. Should be 'neuron', 'neuronorevent' , 'event', 'interval' 'wave', 'popvector', 'continuous', 'marker' or 'all'

Return

None.

Note: All the analysis windows are closed before executing this function. Please note that this function produces an immediate result - after the execution of this function, the number of variables of the specified type is reduced by 1. Therefore, you cannot use a simple **for** loop to delete all the variables of a certain type. You can use **while** loop as shown in the example script below.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# delete the first continuous variable
nex.DeleteVar(doc, 1, "continuous")

# delete all waveform variables
# note that we always delete the first variable
while nex.GetVarCount(doc, "wave") > 0:
    nex.DeleteVar(doc, 1, "wave")
```

NexScript

```
doc = GetActiveDocument()
% delete the first continuous variable
DeleteVar(doc, 1, "continuous")

% delete all waveform variables
% note that we always delete the first variable
while GetVarCount(doc, "wave") > 0
   DeleteVar(doc, 1, "wave")
end
```

Delete

Deletes the specified variable from the file.

Syntax

```
Delete(doc, var)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
var	variableReference	Reference to the variable

Return

None.

Note: All the analysis windows are closed before executing this function.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
var = doc["Event04"]
nex.Delete(doc, var)
```

NexScript

```
doc = GetActiveDocument()
var = doc["Event04"]
Delete(doc, var)
```

3.3.10 Copying Variables

CopySelectedVarsToAnotherFile

Copies selected variables from one file to another.

Syntax

```
CopySelectedVarsToAnotherFile(fromDoc, toDoc)
```

Parameters

Parameter	Type	Description
fromDoc	documentReference	Reference to the document. The variables are copied **from** this document
toDoc	documentReference	Reference to the document. The variables are copied **to** this document

Return

None.

Examples

Python

```
import nex
fromDoc = nex.OpenDocument("C:\\Data\\File1.nex")
toDoc = nex.OpenDocument("C:\\Data\\File2.nex")
```

(continues on next page)

(continued from previous page)

```
# copy all events from fromDoc to toDoc
# first, select all events
nex.SelectAllEvents(fromDoc)
# call CopySelectedVarsToAnotherFile
nex.CopySelectedVarsToAnotherFile(fromDoc, toDoc)
```

NexScript

```
fromDoc = OpenDocument("C:\Data\File1.nex")
toDoc = OpenDocument("C:\Data\File2.nex")
% copy all events from fromDoc to toDoc
% first, select all events
SelectAllEvents(fromDoc)
% call CopySelectedVarsToAnotherFile
CopySelectedVarsToAnotherFile(fromDoc, toDoc)
```

3.3.11 Selecting Variables

SelectAll

Selects all the variables for analysis.

Syntax

```
SelectAll(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SelectAll(doc)
```

NexScript

```
doc = GetActiveDocument()
SelectAll(doc)
```

DeselectAll

Deselects all variables.

Syntax

```
DeselectAll(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
```

SelectAllNeurons

Selects all neuron type variables for analysis.

Syntax

```
SelectAllNeurons(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SelectAllNeurons(doc)
```

NexScript

```
doc = GetActiveDocument()
SelectAllNeurons(doc)
```

SelectAllEvents

Selects all event type variables for analysis.

Syntax

```
SelectAllEvents(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SelectAllEvents(doc)
```

NexScript

```
doc = GetActiveDocument()
SelectAllEvents(doc)
```

Select

Selects the specified variable for analysis.

Syntax

```
Select(doc, var)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
var	variableReference	Reference to the variable

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.Select(doc, doc["Neuron04a"])
```

NexScript

```
doc = GetActiveDocument()
Select(doc, doc["Neuron04a"])
```

Deselect

Deselects the specified variable.

Syntax

```
Deselect(doc, var)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
var	variableReference	Reference to the variable

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
```

(continues on next page)

(continued from previous page)

```
# deselect variable Neuro04a from analysis
nex.Deselect(doc, doc["Neuron04a"])
```

NexScript

```
doc = GetActiveDocument()
% deselect variable Neuro04a from analysis
Deselect(doc, doc["Neuron04a"])
```

SelectVar

Selects the specified variable for analysis.

Syntax

```
SelectVar(doc, number, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
number	number	Variable number within the group specified by varType.
varType	string	Variable type. Should be 'neuron', 'neuronorevent' , 'event', 'interval' 'wave', 'popvector', 'continuous', 'marker' or 'all'

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# select the second neuron for analysis
nex.SelectVar(doc, 2, "neuron")
```

NexScript

```
doc = GetActiveDocument()
% select the second neuron for analysis
SelectVar(doc, 2, "neuron")
```

DeselectVar

Deselects the specified variable.

Syntax

```
DeselectVar(doc, number, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
number	number	Variable number within the group specified by varType.
varType	string	Variable type. Should be 'neuron', 'neuronorevent' , 'event', 'interval' 'wave', 'popvector', 'continuous', 'marker' or 'all'

Return

None.

Note: Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# deselect the second neuron
nex.DeselectVar(doc, 2, "neuron")
```

NexScript

```
doc = GetActiveDocument()
% deselect the second neuron
DeselectVar(doc, 2, "neuron")
```

SelectVariables

Selects variables for analysis using provided comma separated list of variable names.

Syntax

```
SelectVariables(doc, varNamesInCommaSeparatedString)
```

Parameters

Parameter	Type	Description
doc	document reference	Reference to the document
varNamesInCommaSeparatedString	string	List of variable names

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.SelectVariables(doc, 'Neuron05b, Neuron07a')
```

IsSelected

Returns 1 if the variable var is selected, 0 otherwise.

Syntax

```
IsSelected(var)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable

Return

Returns 1 if the variable var is selected, 0 otherwise.

Note: Only selected variables are used in analysis.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
isSel = nex.IsSelected(doc["Neuron04a"])
```

NexScript

```
doc = GetActiveDocument()
isSel = IsSelected(doc["Neuron04a"])
```

EnableRecalcOnSelChange

Enables recalculation of analyses when the list of selected variables changes.

Syntax

```
EnableRecalcOnSelChange()
```

Parameters

None.

Return

None.

Note: If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or `ApplyTemplate()` was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This automatic recalculation is disabled by default. This function allows you to enable automatic recalculation on selection change.

Examples

Python

```
import nex
nex.EnableRecalcOnSelChange()
```

NexScript

```
EnableRecalcOnSelChange ()
```

DisableRecalcOnSelChange

Disables recalculation of analyses when the list of selected variables changes.

Syntax

```
DisableRecalcOnSelChange()
```

Parameters

None.

Return

None.

Note: If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or `ApplyTemplate()` was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This function allows you to disable automatic recalculation on selection change.

Examples

Python

```
import nex
nex.DisableRecalcOnSelChange()
```

NexScript

```
DisableRecalcOnSelChange()
```

GetSelVarNames

Returns the list of selected variables in the document.

Syntax

```
doc.GetSelVarNames()
```

Return

Returns the list of selected variables in the document.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
selNames = doc.GetSelVarNames()
```

3.3.12 Properties of Variables

Name

Returns the name of the `NexVar` object.

Syntax

```
var.Name()
```

Parameters

Name

Return

Returns the name of the `NexVar` object.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
var = nex.GetVar(doc, 1, 'neuron')
print(var.Name())
```

GetName

Returns the name of the variable.

Syntax

```
GetName(var)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable

Return

Returns the name of the variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the first neuron variable
var = nex.GetVar(doc, 1, "neuron")
# get the variable name
name = nex.GetName(var)
```

NexScript

```
doc = GetActiveDocument()
% get the first neuron variable
var = GetVar(doc, 1, "neuron")
% get the variable name
name = GetName(var)
```

Metadata

Returns metadata of the `NexVar` object as Python dictionary.

Syntax

```
var.Metadata()
```

Parameters

Name

Return

Returns metadata of the `NexVar` object as Python dictionary (for example, wire and unit number for a neuron variable).

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
var = nex.GetVar(doc, 1, 'neuron')
meta = var.Metadata()
print(meta)
```

SamplingRate

Returns the sampling rate of the `NexVar` object representing continuous channel.

Syntax

```
var.SamplingRate()
```

Parameters

None.

Return

Returns the sampling rate (in Hertz) of the `NexVar` object representing continuous channel.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
var = nex.GetVar(doc, 1, 'continuous')
print(var.SamplingRate())
```

NumPointsInWave

Returns the number of data points in each waveform of the `NexVar` object representing waveform variable.

Syntax

```
var.NumPointsInWave()
```

Parameters

None.

Return

Returns the number of data points in each waveform of the `NexVar` object representing waveform variable.

Note: Python only

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
var = nex.GetVar(doc, 1, 'wave')
print(var.NumPointsInWave())
```

PreThresholdTime

Returns pre-threshold time in seconds of the `NexVar` object representing waveform variable.

Syntax

```
var.PreThresholdTime()
```

Parameters

None.

Return

Returns pre-threshold time in seconds of the `NexVar` object representing waveform variable.

Note: Python only

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
var = doc["sig001a_wf"]
print(var.PreThresholdTime())
```

MarkerFieldNames

Returns the list of marker field names of the `NexVar` object representing marker variable.

Syntax

```
var.MarkerFieldNames()
```

Parameters

None.

Return

Returns the list of marker field names of the `NexVar` object representing marker variable.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
fieldNames = doc['Strobed'].MarkerFieldNames()
#print the name of the first marker field
print(fieldNames[0])
```

GetVarSpikeCount

Returns the number of timestamps in the specified variable.

Syntax

```
GetVarSpikeCount(doc, varNumber, varType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
varNumber	number	1-based variable number within the group specified by varType.
varType	string	Variable type. Should be 'neuron', 'neuronorevent', 'event', 'interval', 'wave', 'continuous', 'marker' or 'all'

Return

Returns the number of timestamps in the variable. For a continuous variable, returns the number of fragments.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
numSpikes = nex.GetVarSpikeCount(doc, 3, "neuron")
```

NexScript

```
doc = GetActiveDocument()
numSpikes = GetVarSpikeCount(doc, 3, "neuron")
```

GetSpikeCount

Returns the number of timestamps in the variable.

Syntax

```
GetSpikeCount(var)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable.

Returns

Returns the number of timestamps in the variable. For a continuous variable, returns the number of fragments.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
count = nex.GetSpikeCount(doc["Neuron04a"])
```

NexScript

```
doc = GetActiveDocument()
count = GetSpikeCount(doc["Neuron04a"])
```

GetContNumDataPoints

Returns the number of data points in the continuous variable.

Syntax

```
GetContNumDataPoints(var)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to a continuous variable

Return

Returns the number of data points in the continuous variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the number of data points in the continuous variable named "FP01"
numPoints = nex.GetContNumDataPoints(doc["FP01"])
```

NexScript

```
doc = GetActiveDocument()
% get the number of data points in the continuous variable named "FP01"
numPoints = GetContNumDataPoints(doc["FP01"])
```

See also *Getting Variable Data*.

3.3.13 Getting Variable Data

Timestamps

Returns the list of all the timestamps of the `NexVar` object.

Syntax

```
var.Timestamps()
```

Return

Returns the list of all the timestamps of the `NexVar` object. The timestamp values are in seconds.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
ts = doc['ResponseCorrect'].Timestamps()
# print the first timestamp
print(ts[0])
```

Intervals

Returns the list of all the intervals of the `NexVar` object representing interval variable.

Syntax

```
var.Intervals()
```

Return

Returns the list of all the intervals of the `NexVar` object representing interval variable. The values are in seconds.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
intervals = doc['CorrectTrials'].Intervals()
print(intervals)
```

WaveformValues

Returns the list of waveform values of the `NexVar` object representing waveform variable.

Syntax

```
var.WaveformValues()
```

Return

Returns the list of waveform values of the `NexVar` object representing waveform variable.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
waveforms = doc['sig001a_wf'].WaveformValues()
#print the values of the first waveform
print(waveforms[0])
```

Markers

Returns the list of marker values of the `NexVar` object representing marker variable.

Syntax

```
var.Markers()
```

Return

Returns the list of marker values of the `NexVar` object representing marker variable.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
markerValues = doc['Strobed'].Markers()
#print the second marker value of the first marker field
print(markerValues[0][1])
```

MarkerFieldNames

Returns the list of marker field names of the `NexVar` object representing marker variable.

Syntax

```
var.MarkerFieldNames()
```

Return

Returns the list of marker field names of the `NexVar` object representing marker variable.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
fieldNames = doc['Strobed'].MarkerFieldNames()
#print the name of the first marker field
print(fieldNames[0])
```

ContinuousValues

Returns the list of all continuous values of the `NexVar` object representing continuous variable.

Syntax

```
var.ContinuousValues()
```

Return

Returns the list of all continuous values of the `NexVar` object representing continuous variable. The values are in milliVolts.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
values = doc['FP01'].ContinuousValues()
# print the first 10 values
print(values[0:10])
```

ContinuousValuesAsNumPyArray

Returns the `numpy` array object containing all continuous values of the `NexVar` object representing continuous variable. This function is much faster than `ContinuousValues` function when the number of values is very large (more than 200 million).

Syntax

```
var.ContinuousValuesAsNumPyArray()
```

Return

Returns the `numpy` array object containing all continuous values of the `NexVar` object representing continuous variable. The values are in milliVolts.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
values = doc['FP01'].ContinuousValuesAsNumPyArray()
# print the first 10 values
print(values[0:10])
```

ContinuousValuesAsInt16

Returns the list of all continuous values of the NexVar object representing continuous variable. The values are returned as 16-bit integers converted to double.

Syntax

```
var.ContinuousValuesAsInt16()
```

Return

Returns the list of all continuous values of the NexVar object representing continuous variable. The values are returned as 16-bit integers (so called 'raw' values as recorded by a data acquisition system) converted to double.

Note: Python only.

Examples

Python

```
import nex
import numpy as np

doc = nex.GetActiveDocument()
rawValues = np.array(doc['FP01'].ContinuousValuesAsInt16(), dtype=np.int16)
```

(continues on next page)

(continued from previous page)

```
# print first 10 values
print(rawValues[0:10])
```

FragmentTimestamps

Returns the list of fragment timestamps of the `NexVar` object representing continuous variable.

Syntax

```
var.FragmentTimestamps()
```

Return

Returns the list of fragment timestamps of the `NexVar` object representing continuous variable. The timestamps are in seconds.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
fts = doc['FP01'].FragmentTimestamps()
#print the first fragment timestamp
print(fts[0])
```

FragmentCounts

Returns the list of fragment counts (numbers of data points in each fragment) of the `NexVar` object representing continuous variable.

Syntax

```
var.FragmentCounts()
```

Return

Returns the list of fragment counts (numbers of data points in each fragment) of the `NexVar` object representing continuous variable.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
counts = doc['FP01'].FragmentCounts()
#print number of points in the first fragment
print(counts[0])
```

ContMin

Returns minimum of continuous values of the `NexVar` object representing continuous variable.

Syntax

```
var.ContMin()
```

Return

Returns minimum of continuous values of the `NexVar` object representing continuous variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
minFP01 = doc['FP01'].ContMin()
```

ContMax

Returns maximum of continuous values of the `NexVar` object representing continuous variable.

Syntax

```
var.ContMax()
```

Parameters

None.

Return

Returns maximum of continuous values of the `NexVar` object representing continuous variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
maxFP01 = doc['FP01'].ContMax()
```

ContMean

Returns the mean of continuous values of the `NexVar` object representing continuous variable.

Syntax

```
var.ContMean()
```

Return

Returns the mean of continuous values of the `NexVar` object representing continuous variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
meanFP01 = doc['FP01'].ContMean()
```

3.3.14 Modifying Variable Data and Metadata

SetTimestamps

Sets the timestamps of a `NexVar` object representing event or neuron.

Syntax

```
var.SetTimestamps(timestamps)
```

Parameters

Parameter	Type	Description
timestamps	list	list containing timestamps in seconds

Return

None.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["ScriptGeneratedEvent"] = nex.NewEvent(doc, 0)
doc["ScriptGeneratedEvent"].SetTimestamps([ 1.0025,  2.5, 34.5])
```

AddTimestamp

Adds a new timestamp to the specified event or neuron variable.

Syntax

```
AddTimestamp(var, timestamp)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the event or neuron variable
timestamp	number	New timestamp (in seconds)

Return

None.

Note: The new timestamp should be greater than the last timestamp of the specified variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
eventVar = doc["LightOff"]
# add timestamp at 30.5 seconds (assuming that all the existing timestamps of␣
↪eventVar are less than 30.5
nex.AddTimestamp(eventVar, 30.5)
```

NexScript

```
doc = GetActiveDocument()
eventVar = doc["LightOff"]
% add timestamp at 30.5 seconds (assuming that all the existing timestamps of␣
↪eventVar are less than 30.5
AddTimestamp(eventVar, 30.5)
```

AddInterval

Adds a new interval to the specified interval variable.

Syntax

```
AddInterval(var, interval_start, interval_end)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the interval variable
interval_start	number	Start of new interval (in seconds)
interval_end	number	End of new interval (in seconds)

Return

None.

Note: The new interval should not overlap with any of the existing intervals of the specified interval variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
intervalVar = doc["CorrectTrials"]
# add interval [100s,120s]
nex.AddInterval(intervalVar, 100, 120)
```

NexScript

```
doc = GetActiveDocument()
intervalVar = doc["CorrectTrials"]
% add interval [100s,120s]
AddInterval(intervalVar, 100, 120)
```

SetContVarTimestampsAndValues

Sets the timestamps and continuous values of a `NexVar` object representing continuous variable.

Syntax

```
var.SetContVarTimestampsAndValues(timestamps, values)
```

Parameters

Parameter	Type	Description
timestamps	list	list containing timestamps in seconds
values	list	list containing values in milliVolts

Return

None.

Note: Python only.

Examples

Python

```
import nex
import numpy as np

doc = nex.GetActiveDocument()
doc["ScriptGenerated"] = nex.NewContVarWithFloats(doc, 1000)
doc["ScriptGenerated"].SetContVarTimestampsAndValues([0,0.001,0.002],[1,2,22])

start_time = 0
end_time = 100
sample_rate = 1000
time = np.arange(start_time, end_time, 1.0/sample_rate)
frequency = 10
sinewave = np.sin(2 * np.pi * frequency * time)
doc["sine wave"] = nex.NewContVarWithFloats(doc, sample_rate)
doc["sine wave"].SetContVarTimestampsAndValues(time, sinewave)
```

SetContVarStartTimeAndValues

Sets start time and continuous values of a continuous variable with a single fragment.

Syntax

```
var.SetContVarStartTimeAndValues(startTime, values)
```

Parameters

Parameter	Type	Description
startTime	number	start time in seconds
values	list	list containing values in milliVolts

Return

None.

Note: Python only.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["ScriptGenerated"] = nex.NewContVarWithFloats(doc, 1000)
doc["ScriptGenerated"].SetContVarStartTimeAndValues(0.5, [1, 2, 22.3])
```

SetContVarStartTimeAndValues16bit

Sets start time and continuous values of a continuous variable with a single fragment.

Syntax

```
var.SetContVarStartTimeAndValues16bit(startTime, values)
```

Parameters

Parameter	Type	Description
startTime	number	start time in seconds
values	list	list containing raw recorded values

Return

None.

Note: Python only.

Examples

Python

```
import nex
import numpy as np
doc = nex.GetActiveDocument()
# assume that we know the coefficient c to convert raw recorded 16-bit values␣
↪to milliVolts
c = .12345e-5
# we need ymin and ymax values for nex.NewContVar()
# calculate ymin and ymax values so that NeuroExplorer will calculate the␣
↪same coefficient
# to convert raw recorded 16-bit values to milliVolts:
ymax = c*32768.
ymin = -ymax
doc['ScriptGeneratedContVar'] = nex.NewContVar(doc, 1000, ymin, ymax)
# create a test numpy array with 16-bit values
testValues = np.array([7, 8, 9, 11], dtype=np.int16)
```

(continues on next page)

(continued from previous page)

```
# the first data point is recorded at 0.5 seconds. add 16-bit values
doc['ScriptGeneratedContVar'].SetContVarStartTimeAndValues16bit(0.5,␣
↪testValues)

# verify that values in milliVolts are the same
valuesInMV = testValues*c
valuesFromNex = doc['ScriptGeneratedContVar'].ContinuousValues()
for i in range(len(testValues)):
   print(valuesFromNex[i] - valuesInMV[i])
```

AddContValue

Adds a new data point to the specified continuous variable

Syntax

```
AddContValue(var, timestamp, value)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the continuous variable
timestamp	number	Timestamp value in seconds.
value	number	Voltage value in milliVolts

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
contVar = doc["FP01"]
# add voltage value of 25.7 mV at the time 100.3 seconds
nex.AddContValue(contVar, 100.3, 25.7)
```

NexScript

```
doc = GetActiveDocument()
contVar = doc["FP01"]
% add voltage value of 25.7 mV at the time 100.3 seconds
AddContValue(contVar, 100.3, 25.7)
```

SetNeuronPosition

Sets the position (used in 3D displays) of a neuron variable.

Syntax

```
doc.SetNeuronPosition(neuronVar, x, y)
```

Parameters

Parameter	Type	Description
neuronVar	variable reference	Reference to the neuron variable.
x	number	x coordinate of position (within [0,100] range
y	number	y coordinate of position (within [0,100] range

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc.SetNeuronPosition(doc['SPK01a'], 8.0, 12.5)
```

SetNeuronWire

Sets the wire number of a neuron variable.

Syntax

```
doc.SetNeuronWire(neuronVar, wire)
```

Parameters

Parameter	Type	Description
neuronVar	variable reference	Reference to the neuron variable.
wire	number	Wire number

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc.SetNeuronWire(doc['SPK01a'], 16)
```

SetNeuronUnit

Sets the unit number (cluster ID) of a neuron variable.

Syntax

```
doc.SetNeuronUnit(neuronVar, unit)
```

Parameters

Parameter	Type	Description
neuronVar	variable reference	Reference to the neuron variable.
unit	number	Unit number

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc.SetNeuronUnit(doc['SPK01a'], 2)
```

SetWavePreThresholdTime

Sets pre-threshold time in seconds of the `NexVar` object representing waveform variable.

Syntax

```
var.SetPreThresholdTime(preThrTimeInSeconds)
```

Parameters

Parameter	Type	Description
preThrTimeInSeconds	number	pre-threshold time in seconds

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc['sig001a_wf'].SetPreThresholdTime(0.0002)
```

ContVarStoreValuesAsFloats

Converts storage of continuous values from 16-bit integers to 32-bit floats.

Syntax

```
ContVarStoreValuesAsFloats(contVar)
```

Parameters

Parameter	Type	Description
contVar	variable reference	Continuous variable

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ContVarStoreValuesAsFloats(doc['FP01'])
```

NexScript

```
doc = GetActiveDocument()
ContVarStoreValuesAsFloats(doc["FP01"])
```

3.3.15 Operations on Variables

Rename

Renames the specified variable.

Syntax

```
Rename(doc, var, newName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
var	variableReference	Reference to the variable
newName	string	The new name of the variable

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.Rename(doc, doc["Strobed_DIO_01000"], "Reward")
```

NexScript

```
doc = GetActiveDocument()
Rename(doc, doc["Strobed_DIO_01000"], "Reward")
```

Shift

Shifts all the timestamps of a variable.

Syntax

```
Shift(var, shiftBy)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable
shiftBy	number	Shift value (in seconds)

Return

Reference to the new variable.

Note: Returns a new variable with all the timestamps of variable var shifted in time by shiftBy seconds.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["Event04Shifted"] = nex.Shift(doc["Event04"], 10)
```

NexScript

```
doc = GetActiveDocument()
doc["Event04Shifted"] = Shift(doc["Event04"], 10)
```

Join

Creates the new event that contains the timestamps of the two specified variables.

Syntax

```
Join(var1, var2)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable
var2	variableReference	Reference to the variable

Return

Reference to the new variable.

Note: Creates the new event that contains both the timestamps of var1 and the timestamps of var2.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["Events4and5"] = nex.Join(doc["Event04"], doc["Event06"])
```

NexScript

```
doc = GetActiveDocument()
doc["Events4and5"] = Join(doc["Event04"], doc["Event06"])
```

Sync

Creates the new event containing all the timestamps of var1 that are in the intervals [var2+fromTime, var2+toTime].

Syntax

```
Sync(var1, var2, fromTime, toTime)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable
var2	variableReference	Reference to the variable
fromTime	number	Offset minimum (seconds)
toTime	number	Offset maximum (seconds)

Return

Reference to the new variable.

Note: Creates the new event containing all the timestamps of var1 that are in the intervals [var2+from, var2+to]

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["synced_1_and_2"] = nex.Sync(doc["Neuron04a"], doc["Neuron06b"], -0.01, 0.
↪01)
```

NexScript

```
doc = GetActiveDocument()
doc["synced_1_and_2"] = Sync(doc["Neuron04a"], doc["Neuron06b"], -0.01, 0.01)
```

NotSync

Creates the new event containing all the timestamps of var1 that are NOT in the intervals [var2+fromTime, var2+toTime].

Syntax

```
NotSync(var1, var2, fromTime, toTime)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable
var2	variableReference	Reference to the variable
fromTime	number	Offset minimum (seconds)
toTime	number	Offset maximum (seconds)

Return

Reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["notSync_4_and_5"] = nex.NotSync(doc["Neuron04a"], doc["Neuron05c"], - 0.
↪01, 0.01)
```

NexScript

```
doc = GetActiveDocument()
doc["notSync_4_and_5"] = NotSync(doc["Neuron04a"], doc["Neuron05c"], -0.01, 0.
↪01)
```

FirstAfter

Creates the new event containing the first timestamp of var1 in each of the intervals [var2+fromTime, var2+toTime].

Syntax

```
FirstAfter(var1, var2, fromTime, toTime)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable.
var2	variableReference	Reference to the variable.
fromTime	number	Offset of interval start around var2 timestamps
toTime	number	Offset of interval end around var2 timestamps

Return

Reference to the new variable.

Note: Creates the new event containing the first timestamp of var1 in each of the intervals [var2+from, var2+to].

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# find the first spike of Neuron05b after each stimulus (in the first second␣
↪after stimulus)
doc["FirstN1"] = nex.FirstAfter(doc["Neuron05b"], doc["Stimulus"], 0, 1)
```

NexScript

```
doc = GetActiveDocument()
% find the first spike of Neuron05b after each stimulus (in the first second␣
↪after stimulus)
doc["FirstN1"] = FirstAfter(doc["Neuron05b"], doc["Stimulus"], 0, 1)
```

FirstNAfter

Creates the new event containing the first N timestamps of one variable after each of the timestamps of the second variable.

Syntax

```
FirstNAfter(var1, var2, count)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable.
var2	variableReference	Reference to the variable.
count	number	How many timestamps of var1 (that are after each timestamp of var2) to copy to the result

Return

The reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["first5SpikesAfterStimulus"] = nex.FirstNAfter(doc["Neuron06b"], doc[
↪"Stimulus"], 5)
```

NexScript

```
doc = GetActiveDocument()
doc["first5SpikesAfterStimulus"] = FirstNAfter(doc["Neuron06b"], doc["Stimulus
↪"], 5)
```

LastBefore

Creates the new event containing the last timestamp of var1 in each of the intervals [var2+fromTime, var2+toTime].

Syntax

```
LastBefore(var1, var2, fromTime, toTime)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable.
var2	variableReference	Reference to the variable.
fromTime	number	Interval start offset (seconds)
toTime	number	Interval end offset (seconds).

Return

Reference to the new variable.

Note: Creates the new event containing the last timestamp of var1 in each of the intervals [var2+from, var2+to].

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["lastBefore"] = nex.LastBefore(doc["Neuron07a"], doc["Event04"], 0, 5)
```

NexScript

```
doc = GetActiveDocument()
doc["lastBefore"] = LastBefore(doc["Neuron07a"], doc["Event04"], 0, 5)
```

IntervalFilter

Creates the new event containing all the timestamps of the specified event or neuron variable that are in the intervals of the specified interval variable.

Syntax

```
IntervalFilter(var, intervalVar)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable.
intervalVar	variableReference	Reference to the interval variable

Return

Reference to the new variable.

Note: Creates the new event containing all the timestamps of var that are in the intervals of the intervalVar

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["filtered"] = nex.IntervalFilter(doc["Neuron07a"], doc["CorrectTrials"])
```

NexScript

```
doc = GetActiveDocument()
doc["filtered"] = IntervalFilter(doc["Neuron07a"], doc["CorrectTrials"])
```

SelectTrials

Creates the new event containing the specified timestamps of a variable.

Syntax

```
SelectTrials(var, selectList)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable.
selectList	string	A list of comma-separated indexes or ranges of timestamps. for example: 1,3-5,10

Return

The reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["selectedEvents"] = nex.SelectTrials(doc["Event04"], "1,5-10")
```

NexScript

```
doc = GetActiveDocument()
doc["selectedEvents"] = SelectTrials(doc["Event04"], "1,5-10")
```

SelectRandom

Creates the new event containing randomly selected timestamps of the specified variable.

Syntax

```
SelectRandom(var, nSelect)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable.
nSelect	number	Number of timestamps to select

Return

The reference to the new event containing randomly selected timestamps of the specified variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["randomEvents04"] = nex.SelectRandom(doc["Event04"], 10)
```

NexScript

```
doc = GetActiveDocument()
doc["randomEvents04"] = SelectRandom(doc["Event04"], 10)
```

SelectOdd

Creates the new event containing the odd (1st, 3rd, etc.) timestamps of the specified variable.

Syntax

```
SelectOdd(var)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable

Return

The reference to the new event containing the odd (1st, 3rd, etc.) timestamps of the specified variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["oddEvents04"] = nex.SelectOdd(doc["Event04"])
```

NexScript

```
doc = GetActiveDocument()
doc["oddEvents04"] = SelectOdd(doc["Event04"])
```

SelectEven

Creates the new event containing even (2nd, 4th, etc.) timestamps of the specified variable.

Syntax

```
SelectEven(var)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable

Return

The reference to the new event containing even (2nd, 4th, etc.) timestamps of the specified variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["evenEvents04"] = nex.SelectEven(doc["Event04"])
```

NexScript

```
doc = GetActiveDocument()
doc["evenEvents04"] = SelectEven(doc["Event04"])
```

ISIFilter

Creates the new event containing the timestamps of the specified variable that have preceding interspike intervals larger than the specified value.

Syntax

```
ISIFilter(var, minISI)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable
minISI	number	Minimum interspike interval (in seconds).

Return

Reference to the new variable.

Note: Creates the new event containing timestamps of the variable var that have preceding interspike intervals larger than minISI.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["RemovedSmallISI"] = nex.ISIFilter(doc["Neuron07a"], 0.1)
```

NexScript

```
doc = GetActiveDocument()
doc["RemovedSmallISI"] = ISIFilter(doc["Neuron07a"], 0.1)
```

FirstInInterval

Creates the new event. For each interval of the specified interval variable, the first timestamp in this interval is copied to the result.

Syntax

```
FirstInInterval(var, intervalVar)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the neuron or event variable
intervalVar	variableReference	Reference to the interval variable.

Return

The reference to the new variable.

Note: Creates the new event. For each interval of intervalVar, the first var timestamp in this interval is copied to the result.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["firstInTrial"] = nex.FirstInInterval(doc["Neuron04a"], doc["CorrectTrials
↪"])
```

NexScript

```
doc = GetActiveDocument()
doc["firstInTrial"] = FirstInInterval(doc["Neuron04a"], doc["CorrectTrials"])
```

LastInInterval

Creates the new event. For each interval of the specified interval variable, the last timestamp in this interval is copied to the result.

Syntax

```
LastInInterval(var, intervalVar)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable.
intervalVar	variableReference	Reference to the interval variable

Return

Reference to the new variable.

Note: Creates the new event. For each interval of intervalVar, the last var timestamp in this interval is copied to the result.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["LastSpikeInTrial"] = nex.LastInInterval(doc["Neuron04a"], doc[
↪"CorrectTrials"])
```

NexScript

```
doc = GetActiveDocument()
doc["LastSpikeInTrial"] = LastInInterval(doc["Neuron04a"], doc["CorrectTrials
↪"])
```

StartOfInterval

Creates the new event. Copies the start of each interval of the specified interval variable to the result.

Syntax

```
StartOfInterval(intervalVar)
```

Parameters

Parameter	Type	Description
intervalVar	variableReference	Reference to the interval variable

Return

Reference to the new variable.

Note: Creates the new event. Copies the start of each interval of intervalVar to the result.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["CorrectTrialStarts"] = nex.StartOfInterval(doc["CorrectTrials"])
```

NexScript

```
doc = GetActiveDocument()
doc["CorrectTrialStarts"] = StartOfInterval(doc["CorrectTrials"])
```

EndOfInterval

Creates the new event based on the specified interval variable. Copies the end of each interval of the interval variable to the result.

Syntax

```
EndOfInterval(intervalVar)
```

Parameters

Parameter	Type	Description
intervalVar	variableReference	Reference to an interval variable

Return

Reference to the new variable.

Note: Creates the new event based on the specified interval variable. Copies the end of each interval of the interval variable to the result.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["CorrectTrialEnds"] = nex.EndOfInterval(doc["CorrectTrials"])
```

NexScript

```
doc = GetActiveDocument()
doc["CorrectTrialEnds"] = EndOfInterval(doc["CorrectTrials"])
```

MakeIntervals

Creates new interval variable with intervals `[varTimestamp+shiftMin, varTimestamp+shiftMax]`. Overlapping intervals are merged.

Syntax

```
MakeIntervals(var, shiftMin, shiftMax)
```

Parameters

Parameter	Type	Description
var	variableReferenceUseTimestamps	Reference to neuron, event, marker, interval or waveform variable. For neuron, event or marker variable, the variable timestamps are used. For interval variable, interval start times are used as timestamps. For waveform variable, wave timestamps are used.
shiftMin	number	Shift minimum in seconds.
shiftMax	number	Shift maximum in seconds.

Return

Reference to the new variable.

Note: Creates new interval variable with intervals `[varTimestamp+shiftMin, varTimestamp+shiftMax]`. Overlapping intervals are merged.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["IntAroundEvent04"] = nex.MakeIntervals(doc["Event04"], 0, 2)
```

NexScript

```
doc = GetActiveDocument()
doc["IntAroundEvent04"] = MakeIntervals(doc["Event04"], 0, 2)
```

MakeIntFromStart

Creates new interval variable. For each timestamp `tstart` of `intStartVar`, it looks for the first timestamp `tend` of the `intEndVar` after `tstart`. If `tend` is before the next timestamp of `intStartVar`, it adds the interval `[tstart+shift1, tend+shift2]` to the result.

Syntax

```
MakeIntFromStart(intStartVar, intEndVar, shift1, shift2)
```

Parameters

Parameter	Type	Description
intStartVar	variableReference	Reference to the variable
intEndVar	variableReference	Reference to the variable
shift1	number	Shift minimum (seconds).
shift2	number	Shift maximum (seconds).

Return

Reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["Int2"] = nex.MakeIntFromStart(doc["FrameStart"], doc["FrameEnd"], - 0.1,␣
↪0.1)
```

NexScript

```
doc = GetActiveDocument()
doc["Int2"] = MakeIntFromStart(doc["FrameStart"], doc["FrameEnd"], -0.1, 0.1)
```

MakeIntFromEnd

Creates new interval variable. For each timestamp `tend` of the `intEndVar`, it looks for the last timestamp (`tstart`) of the `intStartVar` before `tend`. If `tstart` is after the previous timestamp of `intEndVar`, it adds the interval `[tstart+shift1, tend+shift2]` to the result.

Syntax

```
MakeIntFromEnd(intStartVar, intEndVar, shift1, shift2)
```

Parameters

Parameter	Type	Description
intStartVar	variableReferenceUseTimestamps	Reference to neuron, event, marker, interval or waveform variable. For neuron, event or marker variable, the variable timestamps are used. For interval variable, interval start times are used as timestamps. For waveform variable, wave timestamps are used.
intEndVar	variableReferenceUseTimestamps	Reference to neuron, event, marker, interval or waveform variable. For neuron, event or marker variable, the variable timestamps are used. For interval variable, interval start times are used as timestamps. For waveform variable, wave timestamps are used.
shift1	number	Shift minimum (seconds).
shift2	number	Shift maximum (seconds).

Return

Reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["Int2"] = nex.MakeIntFromEnd(doc["FrameStart"], doc["FrameEnd"],  - 0.1,␣
↪0.1)
```

NexScript

```
doc = GetActiveDocument()
doc["Int2"] = MakeIntFromEnd(doc["FrameStart"], doc["FrameEnd"], -0.1, 0.1)
```

IntOpposite

Creates a new interval variable that contains intervals 'complementary' to the intervals of the specified interval variable.

Syntax

```
IntOpposite(doc, intervalVariable)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
intervalVariable	variableReference	Reference to the interval variable

Return

Reference to the new interval variable.

Note: Creates a new interval variable that contains intervals 'complementary' to the intervals of intervalVariable.

For example, if you have an interval variable that contains time intervals with artifacts, IntOpposite will create a new interval variable with 'clean' time intervals.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["CleanDataIntervals"] = nex.IntOpposite(doc, doc["Artifacts"])
```

NexScript

```
doc = GetActiveDocument()
doc["CleanDataIntervals"] = IntOpposite(doc, doc["Artifacts"])
```

IntOr

Creates a new Interval Variable that contains unions of the intervals of intervalVar1 and intervalVar2.

Syntax

```
IntOr(doc, intervalVar1, intervalVar2)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
intervalVar1	variableReference	Reference to the interval variable
intervalVar2	variableReference	Reference to the interval variable

Return

Reference to the new interval variable.

Note: Creates a new Interval Variable that contains unions of the intervals of intervalVar1 and intervalVar2

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["Trials1and2"] = nex.IntOr(doc, doc["Trials1"], doc["Trials2"])
```

NexScript

```
doc = GetActiveDocument()
doc["Trials1and2"] = IntOr(doc, doc["Trials1"], doc["Trials2"])
```

IntAnd

Creates a new Interval Variable that contains intersections of the intervals of intervalVar1 and intervalVar2.

Syntax

```
IntAnd(intervalVar1, intervalVar2)
```

Parameters

Parameter	Type	Description
intervalVar1	variableReference	Reference to the interval variable
intervalVar2	variableReference	Reference to the interval variable

Return

The reference to the new interval variable.

Note: Creates a new Interval Variable that contains intersections of the intervals of intervalVar1 and intervalVar2

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["Conditions_1_and_2"] = nex.IntAnd(doc["Condition1"], doc["Condition2"])
```

NexScript

```
doc = GetActiveDocument()
doc["Conditions_1_and_2"] = IntAnd(doc["Condition1"], doc["Condition2"])
```

IntSize

Creates a new Interval Variable that contains the intervals with the specified length range.

Syntax

```
IntSize(intervalVar, minInt, maxInt)
```

Parameters

Parameter	Type	Description
intervalVar	variableReference	Reference to the interval variable.
minInt	number	Minimum interval length (in seconds)
maxInt	number	Maximum interval length (in seconds)

Return

Reference to the new interval variable.

Note: Creates a new Interval Variable that contains all of the intervals of intervalVar that have the length which is more or equal to minInt and less than or equal to maxInt.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["TrialsLessThan10secDuration"] = nex.IntSize(doc["Trials1"], 0, 10)
```

NexScript

```
doc = GetActiveDocument()
doc["TrialsLessThan10secDuration"] = IntSize(doc["Trials1"], 0, 10)
```

IntFind

Finds all intervals that contain at least one timestamp of the specified event or neuron variable.

Syntax

```
IntFind(intervalVar, eventVar)
```

Parameters

Parameter	Type	Description
intervalVar	variableReference	Reference to the interval variable.
eventVar	variableReference	Reference to the event or neuron variable

Return

New interval variable.

Note: Creates a new Interval Variable. Each interval of intervalVar that contains one or more timestamps of eventVar is copied to the result.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["IntervalsWithEvent04"] = nex.IntFind(doc["Trials1"], doc["Event04"])
```

NexScript

```
doc = GetActiveDocument()
doc["IntervalsWithEvent04"] = IntFind(doc["Trials1"], doc["Event04"])
```

MarkerExtract

Creates a new event variable based on existing marker variable.

Syntax

```
MarkerExtract(doc, MarkerVariableName, ExtractString)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
MarkerVariableName	string	The name of the marker variable
ExtractString	string	Extract string. See comments.

Return

New event variable.

Note: **ExtractString** contains a list of items separated by commas. Items `AND` or `OR` should be placed between the conditions for the same field. Conditions for each marker field should end with the item `EOF`. The last item in **ExtractString** list should be `END`.

For example, assume that we have a marker variable `Strobed` with one field that contains integer values. To extract all the timestamps with the field value 3 you may use the following command:

```
doc["NewEvent"] = MarkerExtract(doc, "Strobed", "=3,EOF,END")
```

To extract all the timestamps with the field values 3, 4 and 5 you may use the command:

```
doc["NewEvent"] = MarkerExtract(doc, "Strobed", ">2,AND,<6,EOF,END")
```

To use string comparisons in timestamp extraction, add $ sign at the beginning of the string. For example, to extract timestamps with `Ev_Marker` field value `WL`, use:

```
doc["NewEvent1"] = MarkerExtract(doc, "Ev_Marker", "=$WL,EOF,END")
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["NewEvent"] = nex.MarkerExtract(doc, "Strobed", "=3,EOF,END")
```

NexScript

```
doc = GetActiveDocument()
doc["NewEvent"] = MarkerExtract(doc, "Strobed", "=3,EOF,END")
```

NthAfter

Creates the new variable with the N-th timestamp in var1 after each timestamp in var2.

Syntax

```
NthAfter(var1, var2, N)
```

Parameters

Parameter	Type	Description
var1	variableReference	Reference to the variable
var2	variableReference	Reference to the variable

continues on next page

Table 189 – continued from previous page

Parameter	Type	Description
N	number	Spike number.

Return

Reference to the new variable.

Note: Creates the new variable with the N-th timestamp in var1 after each timestamp in var2.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["SecondSpike"] = nex.NthAfter(doc["Neuron05c"], doc["Neuron06d"], 2)
```

NexScript

```
doc = GetActiveDocument()
doc["SecondSpike"] = NthAfter(doc["Neuron05c"], doc["Neuron06d"], 2)
```

PositionSpeed

Calculates the position speed from X and Y coordinate variables and creates a new continuous variable.

Syntax

```
PositionSpeed(varX, varY, deltaT, smoothRadius)
```

Parameters

Parameter	Type	Description
varX	variableReference	Reference to the variable.
varY	variableReference	Reference to the variable.
deltaT	number	Time step for speed calculation
smoothRadius	number	Smooth parameter. See Comments.

Return

Reference to the new variable

Note: PositionSpeed operation calculates the scalar speed of a pair of the position variables.

1. First, for each data point of a Position variable `PosX[T]`, where `T` is time, the raw scalar speed is calculated:

```
dX = PosX[ T + DeltaT ] - PosX[ T ]
dY = PosY[ T + DeltaT ] - PosY[ T ]
RawScalarSpeed[ T ] = sqrt( dX*dX + dY*dY ) / DeltaT
```

If there is no data point at time `T+DeltaT`, a linear interpolation is used to calculate `PosX[T + DeltaT]` and `PosY[T + DeltaT]`.

2. Second, `RawScalarSpeed` is smoothed with the Gaussian filter. The parameters of the filter are such that the width (in seconds) of the Gaussian curve at half the height is equal to the value of Smooth parameter. If Smooth = 0, Gaussian filter is not applied.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
doc["speed"] = nex.PositionSpeed(doc["LED1_X"], doc["LED1_Y"], 0.1, 0.5)
```

NexScript

```
doc = GetActiveDocument()
doc["speed"] = PositionSpeed(doc["LED1_X"], doc["LED1_Y"], 0.1, 0.5)
```

FilterContinuousVariable

Filters the specified continuous variable using the specified frequency filter.

Syntax

```
FilterContinuousVariable(doc, contVar, filteredVarName, filterType,␣
↪filterOrder, freq1, freq2)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
contVar	variableReference	Reference to the variable.
filteredVarName	string	The name of the filtered variable.
filterType	string	The type of the filter. Should be 'Lowpass', 'Highpass', 'Bandpass', 'Bandstop' or 'Notch'
filterOrder	number	The number specifying the filter order. Should be between 3 and 11 inclusive.
freq1	number	Filter frequency parameter (in Hz). See comments below.

continues on next page

Table 191 – continued from previous page

Parameter	Type	Description
freq2	number	Filter frequency parameter (in Hz). See comments below.

Return

None.

Note: If the filter type is `Lowpass` or `Highpass`, `freq1` is a cutoff frequency and `freq2` is not used. Butterworth filter is used.

If the filter type is `Bandpass` or `Bandstop`, `freq1` is the minimum of the frequency range and `freq2` is the maximum of the frequency range. Butterworth filter is used.

If the filter type is `Notch`, `freq1` is the center of the Notch filter and `freq2` is the width of the Notch filter. Standard Notch filter is used.

Examples

The following sample scripts apply band-pass filter to the variable `ContChannel01`. The result of filtering is then saved in a continuous variable `Cont1BandFiltered`. The filter order is 5 and the frequency band is from 1000 Hz to 2000 Hz:

Python

```
import nex
doc = nex.GetActiveDocument()
var = doc["ContChannel01"]
nex.FilterContinuousVariable(doc, var, "Cont1BandFiltered", "Bandpass", 5,␣
↪1000, 2000)
```

NexScript

```
doc = GetActiveDocument()
var = doc["ContChannel01"]
FilterContinuousVariable(doc, var, "Cont1BandFiltered", "Bandpass", 5, 1000,␣
↪2000)
```

FilterContinuousVariableEx

Filters a continuous variable using the specified frequency filter. Returns a reference to the new variable with filtered values.

Syntax

```
FilterContinuousVariableEx(contVar, filterType, filterImplementation,␣
↪filterOrder, freq1, freq2, ripple=1.0)
```

Parameters

Parameter	Type	Description
contVar	variableReference	Reference to the variable.
filterType	string	The type of the filter. Should be `'Lowpass'`, `'Highpass'`, `'Bandpass'`, `'Bandstop'` or `'Notch'`.
filterImplementation	string	Filter implementation. Should be `'IIR Butterworth'`, `'IIR Chebyshev'`, `'FIR Bartlett'`, `'FIR Blackman'`, `'FIR Hamming'` or `'FIR Hann'`.
filterOrder	number	The number specifying the filter order. Use 1 to 12 for IIR filters, 4 and higher for FIR filters. If FIR filter order is odd, order+1 is used.
freq1	number	Filter frequency parameter (in Hz). See comments below.
freq2	number	Filter frequency parameter (in Hz). See comments below.

continues on next page

Table 192 – continued from previous page

Parameter	Type	Description
ripple	number	Filter ripple. Used for `IIR Chebyshev` filter only.

Return

Returns reference to the new continuous variable with filtered values.

Note: Python only.

If the filter type is `Lowpass` or `Highpass`, `freq1` is a cutoff frequency and `freq2` is not used.

If the filter type is `Bandpass` or `Bandstop`, `freq1` is the minimum of the frequency range and `freq2` is the maximum of the frequency range.

If the filter type is `Notch`, `freq1` is the center of the Notch filter and `freq2` is the width of the Notch filter. Standard Notch filter is used.

Examples

The following sample script applies Hamming FIR band-pass filter to the variable `ContChannel01`. The result of filtering is then saved in a continuous variable `Cont1BandFiltered`. The filter order is 51 and the frequency band is from 1000 Hz to 2000 Hz.

Python

```
import nex
doc = nex.GetActiveDocument()
var = doc["ContChannel01"]
doc["Cont1BandFiltered"] = nex.FilterContinuousVariableEx(var, "Bandpass",
↪"FIR Hamming", 51, 1000, 2000)
```

LinearCombinationOfContVars

Calculates a linear combination of two continuous variables.

Syntax

```
LinearCombinationOfContVars(doc, resultName, contVar1, coeff1, contVar2,␣
↪coeff2)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
resultName	string	The name of the result.
contVar1	variableReference	Reference to the first continuous variable.
coeff1	number	Coefficient for the first continuous variable
contVar2	variableReference	Reference to the second continuous variable.
coeff2	number	Coefficient for the second continuous variable

Return

None.

Note: This function calculates a linear combination of two continuous variables. The values of the resulting variable are:

```
contVar1_value*coeff1 + contVar2_value*coeff2
```

Starting with version 5.104, you can get the same result by using math operations (multiplication and addition) on document continuous variables:

```
doc['contVar1_2_Average'] = doc['contVar1']*0.5 + doc['contVar2']*0.5
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# calculate average of contVar1 and contVar2
nex.LinearCombinationOfContVars(doc, "average", doc["FP01"], 0.5, doc["FP02"],
↪ 0.5)
```

NexScript

```
doc = GetActiveDocument()
% calculate average of contVar1 and contVar2
LinearCombinationOfContVars(doc, "average", doc["FP01"], 0.5, doc["FP02"], 0.
↪5)
```

AbsOfContVar

Calculates an absolute value of the signal of a continuous variable. Creates new continuous variable.

Syntax

```
AbsOfContVar(doc, resultName, contVar)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
resultName	string	The name of the result (name of continuous variable).
contVar	variableReference	Reference to continuous variable

Return

None.

Note: This function calculates an absolute value of the signal of a continuous variable. The values of the resulting variable are:

```
abs(contVar_value)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# calculate abs of FP01 and store result in document variable with the name
↪'absOfFP01'
nex.AbsOfContVar(doc, "absOfFP01", doc["FP01"])
```

NexScript

```
doc = GetActiveDocument()
% calculate abs of FP01 and store result in document variable with the name
↪'absOfFP01'
AbsOfContVar(doc, "absOfFP01", doc["FP01"])
```

DecimateContVar

Decimates a continuous variable.

Syntax

```
DecimateContVar(doc, resultName, contVar, decimationFactor)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
resultName	string	The name of the decimated continuous variable.
contVar	variableReference	Reference to the variable.
decimationFactor	number	If decimationFactor = 2, every second data point of contVar is copied to the result, if decimationFactor = 3, every third, etc

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DecimateContVar(doc, "decimated", doc["ContChannel01"], 10)
```

NexScript

```
doc = GetActiveDocument()
DecimateContVar(doc, "decimated", doc["ContChannel01"], 10)
```

ContAdd

Creates a new continuous variable with values contVar[i]+numberToAdd, returns a reference to the new variable.

Syntax

```
ContAdd(contVar, numberToAdd)
```

Parameters

Parameter	Type	Description
contVar	variableReference	Reference to the variable.
numberToAdd	number	The number to add to each value of the contVar

Return

Creates a new continuous variable with values contVar[i]+numberToAdd, returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
baseLine = 10.0
doc["subtractedBaseLine"] = nex.ContAdd(doc["FP01"], -baseLine)

# you can also use arithmetic operations on continuous variables:
doc["subtractedBaseLine"] = doc["FP01"] - baseLine

doc["average of FP01 and FP02"] = ( doc["FP01"] + doc["FP02"] ) / 2
```

NexScript

```
doc = GetActiveDocument()
baseLine = 10.0
doc["subtractedBaseLine"] = ContAdd(doc["FP01"], -baseLine)
```

ContMult

Creates a new continuous variable with values contVar[i]*numberToMultiply, returns a reference to the new variable.

Syntax

```
ContMult(contVar, numberToMultiply)
```

Parameters

Parameter	Type	Description
contVar	variableReference	Reference to the variable.
numberToMultiply	number	The result values will be contVar[i]*numberToMultiply

Return

Creates a new continuous variable with values contVar[i]*numberToMultiply, returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
coeff = -1.0
doc["invertedFP01"] = nex.ContMult(doc["FP01"], coeff)
```

(continues on next page)

(continued from previous page)

```
# you can also use arithmetic operations on continuous variables:
doc["average of FP01 and FP02"] = ( doc["FP01"] + doc["FP02"] ) / 2
```

NexScript

```
doc = GetActiveDocument()
coeff = -1.0
doc["invertedFP01"] = ContMult(doc["FP01"], coeff)
```

ContAddCont

Creates a new continuous variable with values contVar1[i]+contVar2[i], returns a reference to the new variable.

Syntax

```
ContAddCont(contVar1, contVar2)
```

Parameters

Parameter	Type	Description
contVar1	variableReference	Reference to the continuous variable.
contVar2	variableReference	Reference to the continuous variable.

Return

Creates a new continuous variable with values contVar1[i]+contVar2[i], returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
theSumOfFP1andFP2 = nex.ContAddCont(doc["FP01"], doc["FP02"])

# you can also use arithmetic operations on continuous variables:
doc["average of FP01 and FP02"] = ( doc["FP01"] + doc["FP01"] ) / 2
```

NexScript

```
doc = GetActiveDocument()
theSumOfFP1andFP2 = ContAddCont(doc["FP01"], doc["FP02"])
```

ContSubtractCont

Creates a new continuous variable with values contVar1[i] - contVar2[i], returns a reference to the new variable.

Syntax

```
ContSubtractCont(contVar1, contVar2)
```

Parameters

Parameter	Type	Description
contVar1	variableReference	Reference to the continuous variable.
contVar2	variableReference	Reference to the continuous variable.

Return

Creates a new continuous variable with values contVar1[i] - contVar2[i], returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
FP01_minus_FP02 = nex.ContSubtractCont(doc["FP01"], doc["FP02"])

# you can also use arithmetic operations on continuous variables:
doc["FP01_minus_FP02"] = doc["FP01"]-doc["FP01"]
```

NexScript

```
doc = GetActiveDocument()
FP01_minus_FP02 = ContSubtractCont(doc["FP01"], doc["FP02"])
```

ContMultCont

Creates a new continuous variable with values contVar1[i]*contVar2[i], returns a reference to the new variable.

Syntax

```
ContMultCont(contVar1, contVar2)
```

Parameters

Parameter	Type	Description
contVar1	variableReference	Reference to the continuous variable.
contVar2	variableReference	Reference to the continuous variable.

Return

Creates a new continuous variable with values contVar1[i]*contVar2[i], returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
FP01_mult_by_FP02 = nex.ContMultCont(doc["FP01"], doc["FP02"])

# you can also use arithmetic operations on continuous variables:
doc["FP01_mult_by_FP02"] = doc["FP01"]*doc["FP01"]
```

NexScript

```
doc = GetActiveDocument()
FP01_mult_by_FP02 = ContMultCont(doc["FP01"], doc["FP02"])
```

ContDivCont

Creates a new continuous variable with values contVar1[i]/contVar2[i], returns a reference to the new variable.

Syntax

```
ContDivCont(contVar1, contVar2)
```

Parameters

Parameter	Type	Description
contVar1	variableReference	Reference to the continuous variable.
contVar2	variableReference	Reference to the continuous variable.

Return

Creates a new continuous variable with values contVar1[i]/contVar2[i] (if contVar2[i] is zero, the result of division is contVar1[i]), returns a reference to the new variable.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
FP01_divided_by_FP02 = nex.ContDivCont(doc["FP01"], doc["FP02"])

# you can also use arithmetic operations on continuous variables:
doc["FP01_divided_by_FP02"] = doc["FP01"]/doc["FP01"]
```

NexScript

```
doc = GetActiveDocument()
FP01_divided_by_FP02 = ContDivCont(doc["FP01"], doc["FP02"])
```

3.3.16 Analysis Functions

How to Specify Template Names

By default, the templates are stored in the directory:

```
C:\Users\<USER>\Documents\NeuroExplorer 5\Templates
```

where <USER> is your Windows user name.

You can also specify custom templates directory using **Templates | Template Properties...** menu command.

The following discussion assumes that you are using the default templates directory.

When the template name is specified as "PSTH", for example:

```
ApplyTemplate(doc, "PSTH")
```

NeuroExplorer will use the following template file

```
C:\Users\<USER>\Documents\NeuroExplorer 5\Templates\PSTH.ntp
```

You can also specify the template name relative to the main templates folder.

For example, the template name "Grant\Auto.ntp" used in

```
ApplyTemplate(doc, "Grant\Auto.ntp")
```

means that the template file

```
C:\Users\<USER>\Documents\NeuroExplorer 5\Templates\Grant\Auto.ntp
```

will be used.

Finally, you can specify the absolute path of the template file.

In Python:

```
nex.ApplyTemplate(doc, "C:\\MyTemplates\\Auto.ntp")
```

In NexScript:

```
ApplyTemplate(doc, "C:\MyTemplates\Auto.ntp")
```

ApplyTemplate

Runs the analysis specified in the analysis template.

Syntax

```
ApplyTemplate(doc, templateName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
templateName	string	The name of the template. See *How To Specify Template Names* for details on how to specify the template name parameter

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# run PSTH analysis saved in the template PerieventHistograms
nex.ApplyTemplate(doc, "PerieventHistograms")
```

NexScript

```
doc = GetActiveDocument()
% run PSTH analysis saved in the template PerieventHistograms
ApplyTemplate(doc, "PerieventHistograms")
```

ApplyTemplateToWindow

Runs the analysis specified in the template and shows the result in the specified Graph window.

Syntax

```
ApplyTemplateToWindow(doc, templatename, windownumber)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
templatename	string	The name of the template. See *How To Specify Template Names* for details on how to specify template name parameter.
windownumber	number	1-based index of the graph window of the document. Graph windows are named Graphs1, Graphs2, etc. Thus, if you need to specify window Graphs2, windownumber should be equal to 2. If the Graph window with the specified number does not exist, a new Graph window is created

Return

None

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# Run PSTH analysis saved in the template PerieventHistograms and show the␣
 ↪results in Graph2 window
# We specify template name as "PerieventHistograms". This means that the␣
 ↪template file
# C:\Users\current_user\Documents\NeuroExplorer 5\Templates\
 ↪PerieventHistograms.ntp
# will be used (where current_user is your Windows user name)
nex.ApplyTemplateToWindow(doc, "PerieventHistograms", 2)
```

NexScript

```
doc = GetActiveDocument()
% Run PSTH analysis saved in the template PerieventHistograms and show the␣
 ↪results in Graph2 window
% We specify template name as "PerieventHistograms". This means that the␣
 ↪template file
% C:\Users\current_user\Documents\NeuroExplorer 5\Templates\
 ↪PerieventHistograms.ntp
% will be used (where current_user is your Windows user name)

ApplyTemplateToWindow(doc, "PerieventHistograms", 2)
```

PrintGraphics

Prints the contents of the first graphical window of the document.

Syntax

```
PrintGraphics(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.PrintGraphics(doc)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
PrintGraphics(doc)
```

SaveGraphics

Saves the graphics of the first graphics window of the document to a WMF, PNG or SVG (Scalable Vector Graphics) file. Use SVG file type if you plan to edit graphics in PowerPoint of Adobe Illustrator.

Syntax

```
SaveGraphics(doc, filePath, graphicsFileType)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
filePath	string	Graphics file path
graphicsFileType	number	Graphics file type (0: WMF, 1: PNG, 2: SVG, 3: EMF, 4: TIFF).

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
nex.SaveGraphics(doc, "C:\\Data\\NexGraphics.wmf", 0)
nex.SaveGraphics(doc, "C:\\Data\\NexGraphics.png", 1)
nex.SaveGraphics(doc, "C:\\Data\\NexGraphics.svg", 2)
nex.SaveGraphics(doc, "C:\\Data\\NexGraphics.emf", 3)
nex.SaveGraphics(doc, "C:\\Data\\NexGraphics.tiff", 4)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
SaveGraphics(doc, "C:\Data\NexGraphics.wmf", 0)
SaveGraphics(doc, "C:\Data\NexGraphics.png", 1)
SaveGraphics(doc, "C:\Data\NexGraphics.svg", 2)
SaveGraphics(doc, "C:\Data\NexGraphics.emf", 3)
SaveGraphics(doc, "C:\Data\NexGraphics.tiff", 4)
```

SendGraphicsToPowerPoint

Sends the contents of the first graphical window of the document to the specified PowerPoint presentation.

Syntax

```
SendGraphicsToPowerPoint(doc, presentationPath, slideTitle, comment,␣
↪addParameters, useBitmap)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
presentationPath	string	Path of the presentation file. File extension should be .ppt (NeuroExplorer cannot save .pptx files). If file extension is not .ppt, .ppt extension is added to the file path.
slideTitle	string	Slide title.
comment	string	Slide comment. Will be shown below graphics.
addParameters	number	If 1, add a text box with analysis parameter values.
useBitmap	number	If 1, transfer graphics as a bitmap, otherwise, transfer graphics as a metafile.

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SendGraphicsToPowerPoint(doc, "C:\\Data\\NexResults.ppt", "Slide 1",
↪"Sample slide", 1, 0)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
SendGraphicsToPowerPoint(doc, "C:\Data\NexResults.ppt", "Slide 1", "Sample␣
↪slide", 1, 0)
```

ModifyTemplate

Modifies one of the template parameters.

Syntax

```
ModifyTemplate(doc, templateName, paramName, newParValue)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
templateName	string	The name of the template. See *How To Specify Template Names* for details on how to specify template name parameter
paramName	string	The name of the parameter to be modified.
newParValue	string	The new value of the parameter (as a string).

Return

None.

Examples

To set the new bin value in the *PerieventHistograms* template, you need to write:

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "PerieventHistograms", "Bin (sec)", "0.5")
```

NexScript

```
doc = GetActiveDocument()
ModifyTemplate(doc, "PerieventHistograms", "Bin (sec)", "0.5")
```

Note that parameter name should be specified exactly as it is shown in the left column of the *Properties Panel* (e.g. not `"Bin"`, but `"Bin (sec)"` as in the example above). You can select the parameter name in the left column of the *Properties Panel* and press Ctrl+C to copy the parameter name to the clipboard. Then, paste the name of the parameter into your script.

If you need to use a numeric value as newParValue, you need to convert it to string using `str` function in Python and `NumToStr` function in NexScript. For example, if you need to set `Select Data From = doc["Start"][1]`:

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "Select Data From (sec)", str(doc["Start
↪"][1]))
```

ModifyTemplate can be used to specify multiple references in *Perievent Histograms*, *Crosscorrelograms* and *Perievent Rasters* by using +:

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "Ref. type", "Table (row)")
nex.ModifyTemplate(doc, "Peri", "Reference", "Event04+Event05+Event06")
nex.ApplyTemplate(doc, "Peri")
```

You can also use + to specify multiple interval filters in *Perievent Histograms*, *Crosscorrelograms* and *Perievent Rasters*.

In Python, you can also use JSON array notation `'["Event04", "Event05", "Event06"]'`:

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "Reference", '["Event04", "Event05", "Event06
↪"]')
```

In Python, you can also use JSON array notation to specify multiple interval filters in `Interval Filter` parameter when `Int. Filter Type` is `Table (row)` or `Table (col)`.

For example, to specify two interval filters `IntFilter1` and `IntFilter2`:

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "Int. Filter Type", "Table (col)")
nex.ModifyTemplate(doc, "Peri", "Interval Filter", '["IntFilter1", "IntFilter2
↪"]')
```

To specify `Frequency Bands` parameter in templates that run `Power Spectra for Continuous` analysis, use a comma separated list `band1_name_in_double_quotes, band1_start_frequency, band1_end_frequency, band2_name_in_double_quotes, band2_start_frequency, band2_end_frequency`, etc.:

```
import nex
doc = nex.GetActiveDocument()
```

(continues on next page)

(continued from previous page)

```
nex.ModifyTemplate(doc, 'Spectrum2', 'Frequency Bands', '"Delta",1,3.5,"Theta
↪",3.5,7,"Alpha",7,12,"Beta",12,30,"Gamma",30,80,"Epsilon",80,250')
```

ModifyTemplate can be used to specify graphics parameters. To change the Graph parameter, you need to add `Graph|` before the parameter name:

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "Graph|Graph Style", "Histogram")
nex.ModifyTemplate(doc, "Peri", "Graph|Line color", "1")
nex.ModifyTemplate(doc, "Peri", "Graph|Fill under line", "0")
```

To change the Y Axis parameter, you need to add `YAxis|` before the parameter name:

```
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "YAxis|Max Type", "Fixed")
nex.ModifyTemplate(doc, "Peri", "YAxis|Fixed Max", "50.")
```

To change the X Axis parameter, you need to add `XAxis|` before the parameter name.

```
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "Peri", "XAxis|Show numerics", "Never")
```

To modify Markers parameter in Perievent Rasters template, you can use the parameter values that start with either `Size:`, `Clear` or `AddMarker:`.

For example, `"Size:8"` sets the size of all markers at 8 points. `"Clear"` removes all marker events. See sample code below.

`"AddMarker:Event06,Triangle Down,0,0,0,0,255,0"` adds new marker event Event06 with marker shape `Triangle Down`, border color `(0,0,0)` (meaning red=0,green=0,blue=0) and interior color `(0,255,0)` (red=0,green=255,blue=0). Color values should be numbers from 0 to 255.

Valid marker shapes are: `Triangle Down, Triangle Up, Diamond, Square and Circle`.

```
import nex
doc = nex.GetActiveDocument()
nex.ModifyTemplate(doc, "PRaster", "Markers", "Size:8") # specifies marker␣
↪size in points (8)
nex.ModifyTemplate(doc, "PRaster", "Markers", "Clear") # removes all marker␣
↪events
nex.ModifyTemplate(doc, "PRaster", "Markers", "AddMarker:Event06,Triangle␣
↪Down,0,0,0,0,255,0") # adds new marker Event6
```

(continues on next page)

(continued from previous page)

```
nex.ModifyTemplate(doc, "PRaster", "Markers", "AddMarker:Neuron07a,Circle,255,
↪0,0,0,0,255") # adds new marker Neuron07a
```

GetTemplateParValue

Returns the value of the specified template parameter as string.

Syntax

```
GetTemplateParValue(templateName, paramName)
```

Parameters

Parameter	Type	Description
templateName	string	The name of the template. See *How To Specify Template Names* for details on how to specify template name parameter
paramName	string	The name of the parameter

Return

Returns the value of the specified template parameter as string.

Note: The parameter name should be specified exactly as it is shown in the left column of the *Properties Panel*.

For example, to get the bin value in the *Rate Histograms* template, you need to write:

```
valueAsString = GetTemplateParValue("Rate Histograms", "Bin (sec)")
```

Note that parameter name is not "Bin", but "Bin (sec)". You can select the parameter name in the left column of the *Properties Panel* and press Ctrl+C to copy the parameter name to the clipboard. Then, paste the name of the parameter into your script.

If you need to use a numeric value of the parameter, you need to convert the string to number using `StrToNum` function.

Examples

Python

```
import nex
valueAsString = nex.GetTemplateParValue("Autocorrelograms", "Bin (sec)")
```

RecalculateAnalysisInWindow

Forces recalculation of analysis in the specified graphic window.

Syntax

```
RecalculateAnalysisInWindow(doc, graphWindowNumber)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
graphWindowNumber	number	Index of the Graph window

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.RecalculateAnalysisInWindow(doc, 1)
```

NexScript

```
doc = GetActiveDocument()
RecalculateAnalysisInWindow(doc, 1)
```

SaveNumResults

Saves the numerical results to a text file with the specified name.

Syntax

```
SaveNumResults(doc, fileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
fileName	string	File path for saved results

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveNumResults(doc, "C:\\Data\\res1.txt")
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
ApplyTemplate(doc, "Autocorrelograms")
SaveNumResults(doc, "C:\Data\res1.txt")
```

SaveNumSummary

Saves the summary of numerical results to a text file with the specified name.

Syntax

```
SaveNumSummary(doc, filename)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
filename	string	File path for saved results

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
```

(continues on next page)

(continued from previous page)

```
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveNumSummary(doc, "C:\\Data\\res1summary.txt")
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
ApplyTemplate(doc, "Autocorrelograms")
SaveNumSummary(doc, "C:\Data\res1summary.txt")
```

DisableRecalcOnSelChange

Disables recalculation of analyses when the list of selected variables changes.

Syntax

```
DisableRecalcOnSelChange()
```

Parameters

None.

Return

None.

Note: If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or `ApplyTemplate` was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This function allows you to disable automatic recalculation on selection change.

Examples

Python

```
import nex
nex.DisableRecalcOnSelChange()
```

NexScript

```
DisableRecalcOnSelChange()
```

EnableRecalcOnSelChange

Enables recalculation of analyses when the list of selected variables changes.

Syntax

```
EnableRecalcOnSelChange ()
```

Parameters

None.

Return

None.

Note: If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or `ApplyTemplate` was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This automatic recalculation is disabled by default. This function allows you to enable automatic recalculation on selection change.

Examples

Python

```
import nex
nex.EnableRecalcOnSelChange()
```

NexScript

```
EnableRecalcOnSelChange ()
```

SaveResults

Saves numerical and graphical results as well as an analysis template for the first graphical view of the document.

Syntax

```
SaveResults(doc, fileName, comment)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
fileName	string	The path of the .nexresult file. Use **SavedResults | Open Saved Results File...** menu command to open saved result
comment	string	The user comment for the result

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveResults(doc, "C:\\Data\\MyResult.nexresult", "some comment")
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
SaveResults(doc, "C:\Data\MyResult.nexresult", "some comment")
```

SelectVariablesIn1DView

Syntax

```
SelectVariablesIn1DView(doc, listOfVarNames)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
listOfVarNames	string	A string containing comma-separated list of variable names

Return

None.

Note: There should be no spaces before or after commas in the variables list. If a variable name contains comma(s) and Python is used, the variable name should be enclosed in double quotes. See sample code below.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ExecuteMenuCommand('View|1D Data Viewer Window')
nex.SelectVariablesIn1DView(doc, 'Neuron04a,Neuron05b,Neuron05c')
# if we want to select a variable with a name containing comma(s),
# the variable name should be enclosed in double quotes. For example,
# if variable name is
# channel 1, cluster2:

nex.SelectVariablesIn1DView(doc, 'Neuron04a,Neuron05b,Neuron05c,"channel 1,␣
↪cluster2"')
```

NexScript

```
doc = GetActiveDocument()
ExecuteMenuCommand("View|1D Data Viewer Window")
SelectVariablesIn1DView(doc, "Neuron04a,Neuron05b,Neuron05c")
```

doc.GetAllAnalysisParameters

Returns all analysis parameters.

Syntax

```
doc.GetAllAnalysisParameters()
```

Parameters

None.

Return

Returns all analysis parameters.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
pars = doc.GetAllAnalysisParameters()
```

doc.GetPythonAnalysisInput

Returns Python analysis input as a JSON string. This function can only be run from the code in the script that is specified as **Python Analysis Script** in Python Analysis properties dialog.

Syntax

```
doc.GetPythonAnalysisInput()
```

Parameters

None.

Return

Returns Python analysis input as a JSON string.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
inputAsString = doc.GetPythonAnalysisInput()
```

doc.SetPythonAnalysisOutput

Sets Python analysis output as string.

Syntax

```
doc.SetPythonAnalysisOutput(theOutput)
```

Parameters

Parameter	Type	Description
theOutput	string	Python analysis output as a JSON string.

Return

Sets Python analysis output as string.

Examples

Python

```
import nex
import json
doc = nex.GetActiveDocument()
result = {}
result['XAxisLabel'] = 'Frequency'
result['YAxisLabel'] = 'Raw Spectrum'
result['XValues'] = []
result['YValues'] = []
theOutput = json.dumps(result)
doc.SetPythonAnalysisOutput(theOutput)
```

doc.GetPostProcessingScriptParameter

Returns post-processing script parameter as string.

Syntax

```
doc.GetPostProcessingScriptParameter()
```

Parameters

None.

Return

Returns post-processing script parameter as string.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
parAsString = doc.GetPostProcessingScriptParameter()
```

3.3.17 Numerical Results

GetAllNumericalResults

Returns the list of all values (as a list of columns) in the Numerical Results Window of the first graphical view of the document.

Syntax

```
doc.GetAllNumericalResults()
```

Parameters

None.

Return

Returns the list of all values (as a list of columns) in the Numerical Results Window of the first graphical view of the document.

Note: GetAllNumericalResults() returns a list object with all the numerical results. Each element of the list is a list containing results from a single column in Numerical Results table. For example, to get the 10-th value of the first column of numerical results, use `allNumRes[0][9]` (see Examples below).

You can also use global function `nex.GetAllNumericalResults(doc)`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
allNumRes = nex.GetAllNumericalResults(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
allNumRes = doc.GetAllNumericalResults()
# allNumRes now contains a list object with all numerical results.
# for example, to get the 10-th value of the first column of numerical␣
↪results, use allNumRes[0][9]
# thus, allNumRes[0][9] equals to the result of nex.GetNumRes(doc, 10, 1)
```

GetAllNumericalResultsAsCOM

Returns the list of all values (as a list or rows, similar to `GetNumericalResults()` COM method) in the Numerical Results Window of the first graphical view of the document.

Syntax

```
doc.GetAllNumericalResultsAsCOM()
```

Parameters

None.

Return

Returns the list of all values (as a list or rows) in the Numerical Results Window of the first graphical view of the document.

Note: GetAllNumericalResultsAsCOM() returns a list object with all the numerical results. Each element of the list is a list containing results from a single row in Numerical Results table. For example to get the 10-th value of the first column of numerical results, use `allNumRes[9][0]` (see Examples below).

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
allNumResAsCOM = doc.GetAllNumericalResultsAsCOM()
# allNumRes now contains a list object with all numerical results.
# for example, to get the 10-th value of the first column of numerical␣
↪results, use allNumResAsCOM[9][0]
# thus, allNumResAsCOM[9][0] equals to the result of nex.GetNumRes(doc, 10, 1)
```

GetAllNumResSummaryData

Returns the list of all the values in the Numerical Results Summary Window of the first graphical view of the document.

Syntax

```
doc.GetAllNumResSummaryData()
```

Parameters

None.

Return

Returns the list of all the values in the Numerical Results Summary Window of the first graphical view of the document.

Note: Available only in Python.

You can also use global function `nex.GetAllNumResSummaryData(doc)`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, 'Autocorrelograms')
summary = nex.GetAllNumResSummaryData(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
summary = doc.GetAllNumResSummaryData()
# summary now contains a list object with all the values of summary of␣
↪numerical results window
# for example, to get the 5-th value of the first column of summary of␣
↪numerical results, use summary[0][4]
```

GetNumResSummaryColumnNames

Returns the list of column names in the Summary of Numerical Results Window of the first graphical view of the document.

Syntax

```
doc.GetNumResSummaryColumnNames()
```

Parameters

None.

Return

Returns the list of column names in the Summary of Numerical Results Window of the first graphical view of the document.

Note: Python only.

You can also use global function `nex.GetNumResSummaryColumnNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
names = nex.GetNumResSummaryColumnNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
names = doc.GetNumResSummaryColumnNames()
```

GetNumResColumnNames

Returns the list of column names in the Numerical Results window of the first graphical view of the document.

Syntax

```
doc.GetNumResColumnNames()
```

Parameters

None.

Return

Returns the list of column names in the Numerical Results window of the first graphical view of the document.

Note: Python only.

You can also use global function `nex.GetNumResColumnNames(doc)`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
names = nex.GetNumResColumnNames(doc)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
names = doc.GetNumResColumnNames()
```

GetNumRes

Returns the value of the specified cell in the Numerical Results Window of the first graphical view of the document.

Syntax

```
GetNumRes(doc, row, col)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
row	number	1-based row number.
col	number	1-based column number.

Return

Returns the numeric value of the specified cell in the Numerical Results Window of the first graphical view of the document.

Note: In Python, you can also use NexDoc `GetAllNumericalResults()` method. Using `GetAllNumericalResults()` is much faster. `GetAllNumericalResults()` returns a list object with all the numerical results. Each element of the list is a list containing values from a single column in Numerical Results table. For example:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
allNumRes = doc.GetAllNumericalResults()
# allNumRes now contains a list object with all numerical results.
# for example, to get the 10-th value of the first column of numerical␣
↪results, use allNumRes[0][9]
# thus, allNumRes[0][9] equals to the result of nex.GetNumRes(doc, 10, 1)
```

There is also `GetNumResValue(row, col)` NexDoc method.

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
# get the value of the second bin of the first histogram
binCount = doc.GetNumResValue(2, 1)
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
# get the value of the second bin of the first histogram
binCount = nex.GetNumRes(doc, 2, 1)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
% get the value of the second bin of the first histogram
binCount = GetNumRes(doc, 2, 1)
```

GetNumResNCols

Returns the number of columns of the Numerical Results Window of the first graphical view of the document.

Syntax

```
GetNumResNCols(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the number of columns of the Numerical Results Window of the first graphical view of the document.

Note: In Python, you can also use doc method `GetNumResNCols`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numCols = doc.GetNumResNCols()
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numCols = nex.GetNumResNCols(doc)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
numCols = GetNumResNCols(doc)
```

GetNumResNRows

Returns the number of rows of the Numerical Results Window of the first graphical view of the document.

Syntax

```
GetNumResNRows(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the number of rows of the Numerical Results Window of the first graphical view of the document.

Note: in Python, you can also use doc method `GetNumResNRows`:

```
import nex
doc = nex.GetActiveDocument()
numRows = doc.GetNumResNRows()
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numRows = nex.GetNumResNRows(doc)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
numRows = GetNumResNRows(doc)
```

GetNumResColumnName

Returns the name of the specified column of the Numerical Results Window.

Syntax

```
GetNumResColumnName(doc, col)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
col	number	1-based column number.

Return

Returns the name of the column of the Numerical Results Window of the first graphical view of the document.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
# get the name of the second column of the Numerical Results Window
colName = nex.GetNumResColumnName(doc, 2)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
% get the name of the second column of the Numerical Results Window
colName = GetNumResColumnName(doc, 2)
```

GetNumResSummaryNCols

Returns the number of columns of the Numerical Results Summary Window of the first graphical view of the document.

Syntax

```
GetNumResSummaryNCols(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the number of columns of the Numerical Results Summary Window of the first graphical view of the document.

Note: In Python, you can also use doc method `GetNumResSummaryNCols`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numCols = doc.GetNumResSummaryNCols()
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numCols = nex.GetNumResSummaryNCols(doc)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
numCols = GetNumResSummaryNCols(doc)
```

GetNumResSummaryNRows

Returns the number of rows of the Numerical Results Summary Window of the first graphical view of the document.

Syntax

```
GetNumResSummaryNRows(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

Returns the number of rows of the Numerical Results Summary Window of the first graphical view of the document.

Note: In Python, you can also use doc method `GetNumResSummaryNRows`:

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numRows = doc.GetNumResSummaryNRows()
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
numRows = nex.GetNumResSummaryNRows(doc)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
numRows = GetNumResSummaryNRows(doc)
```

GetNumResSummaryColumnName

Returns the name of the specified column of the Numerical Results Summary Window of the first graphical view of the document.

Syntax

```
GetNumResSummaryColumnName(doc, col)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
col	number	1-based column number.

Return

Returns the name of the column of the Numerical Results Summary Window of the first graphical view of the document.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
# get the name of the third column in Numerical Results Summary
colName = nex.GetNumResSummaryColumnName(doc, 3)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
% get the name of the third column in Numerical Results Summary
colName = GetNumResSummaryColumnName(doc, 3)
```

GetNumResSummaryData

Returns the string value of the specified cell in the Numerical Results Summary Window of the first graphical view of the document.

Syntax

```
GetNumResSummaryData(doc, row, col)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
row	number	1-based row number.
col	number	1-based column number.

Return

Returns the string value of the specified cell in the Numerical Results Summary Window of the first graphical view of the document.

Note: In Python, use `GetAllNumResSummaryData()` NexDoc method to get all the summary values at once.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")

# get the value of the cell in row 3, column 2
summaryCellString = nex.GetNumResSummaryData(doc, 3, 2)
```

(continues on next page)

(continued from previous page)

```
# get all the values in summary
summary = doc.GetAllNumResSummaryData()
# summary now contains a list object with all the values of summary of␣
↪numerical results window
# for example, to get the 5-th value of the first column of summary of␣
↪numerical results, use summary[0][4]
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
% get the value of the cell in row 3, column 2
summaryCellString = GetNumResSummaryData(doc, 3, 2)
```

SaveNumResults

Saves the numerical results to a text file with the specified name.

Syntax

```
SaveNumResults(doc, fileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
fileName	string	File path for saved results

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveNumResults(doc, "C:\\Data\\res1.txt")
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
ApplyTemplate(doc, "Autocorrelograms")
SaveNumResults(doc, "C:\Data\res1.txt")
```

SaveNumSummary

Saves the summary of numerical results to a text file with the specified name.

Syntax

```
SaveNumSummary(doc, filename)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
filename	string	File path for saved results

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SaveNumSummary(doc, "C:\\Data\\res1summary.txt")
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
ApplyTemplate(doc, "Autocorrelograms")
SaveNumSummary(doc, "C:\Data\res1summary.txt")
```

SendResultsToExcel

Sends numerical results (of the first graphics window of the document) to Excel.

Syntax

```
SendResultsToExcel(doc, fileName, worksheetName, useFirstEmptyRow, cellName,␣
↪includeHeader, includeFileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
fileName	string	Excel file path.
worksheetName	string	Excel worksheet name.
useFirstEmptyRow	number	If 1, NeuroExplorer will ignore cellName parameter and add the results to the first row where the cell in column A is empty. If 2, NeuroExplorer will ignore cellName parameter and add the results to the first column where the cell in row 1 is empty. If 0, NeuroExplorer will copy the data starting with the cell specified in **cellName.**
cellName	string	Excel cell where to copy the results. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.
includeHeader	number	If 1, NeuroExplorer will paste column names.
includeFileName	number	If 1, NeuroExplorer will add a column with the data file name.

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SendResultsToExcel(doc, "C:\\Data\\NexResults.xlsx", "Nex", 0, "A1", 1, 0)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
SendResultsToExcel(doc, "C:\Data\NexResults.xlsx", "Nex", 0, "A1", 1, 0)
```

SendResultsSummaryToExcel

Sends summary of numerical results (of the first graphics window of the document) to Excel.

Syntax

```
SendResultsSummaryToExcel(doc, fileName, worksheetName, useFirstEmptyRow,␣
↪cellName, includeHeader, includeFileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
fileName	string	Excel file path.
worksheetName	string	Excel worksheet name.

continues on next page

Table 226 – continued from previous page

Parameter	Type	Description
useFirstEmptyRow	number	If 1, NeuroExplorer will ignore cellName parameter and add the results to the first row where the cell in column A is empty. If 2, NeuroExplorer will ignore cellName parameter and add the results to the first column where the cell in row 1 is empty. If 0, NeuroExplorer will paste data starting with the cell specified in **cellName.**
cellName	string	Excel cell where to paste the results. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.
includeHeader	number	If 1, will paste column names.
includeFileName	number	If 1, will add a column with the data file name.

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
nex.SendResultsSummaryToExcel(doc, "C:\\Data\\res.xlsx", "FromNex", 0, "A1",␣
↪1, 0)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
SendResultsSummaryToExcel(doc, "C:\Data\res.xlsx", "FromNex", 0, "A1", 1, 0)
```

3.3.18 User Interface Functions

Dialog

Shows a dialog that can be used to specify the script parameters.

Syntax

```
Dialog(doc, par1, name1, type1, par2, name2, type2, ...)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document. Could be zero if all the type values are 'number' or 'string'.
par1	variable	A variable that will be assigned a value after Dialog exits. The variable should be created before the Dialog function is called
name1	string	Prompt that will be shown in the dialog.
type1	string	Parameter type. It should be one of: 'number', 'string', 'neuron', 'neuronorevent', 'event', 'interval', 'wave', 'continuous', 'marker' or 'all'.

Return

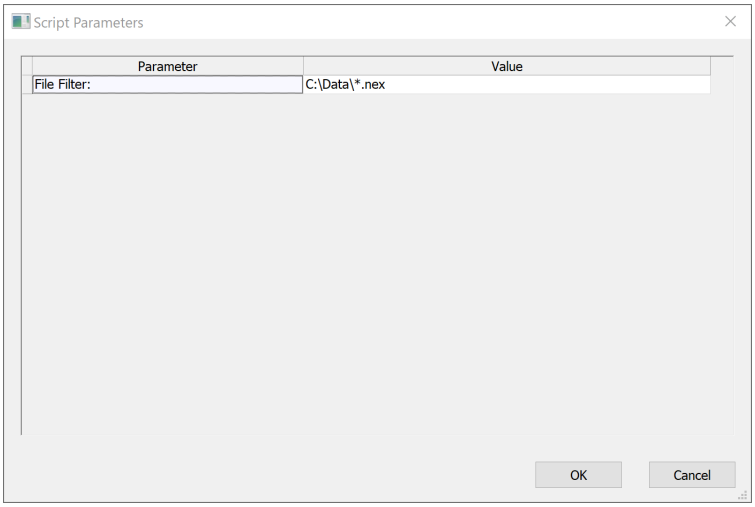
Dialog function returns 1 if user pressed OK button or 0, if user pressed Cancel button.

Examples

Python

```
filefilter = "C:\\Data\\*.nex"

# show the dialog to the user
__wrapper = []
res = nex.Dialog(0., filefilter, "File Filter:", "string", __wrapper )
filefilter = __wrapper[0]
```

The following dialog will be shown:



Now a user can type the new value in the File Filter edit box. If the user presses OK button, the Dialog function returns 1, otherwise, it returns 0.

The following script will allow a user to choose one of the neurons in the active document and select this neuron for analysis:

```
doc = GetActiveDocument()
Neuron_Number = 1
% choose a neuron
res = Dialog(doc, Neuron_Number, "Select Neuron", "neuron")
% get the neuron variable and select it
Neuron_Var = GetVar(doc, Neuron_Number, "neuron")
Select(doc, Neuron_Var)
```

Here is the Python equivalent:

```
import nex
doc = nex.GetActiveDocument()
Neuron_Number = 1
# choose a neuron
__wrapper = []
res = nex.Dialog(doc, Neuron_Number, "Select Neuron", "neuron", __wrapper )
Neuron_Number = __wrapper[0]
# get the neuron variable and select it
Neuron_Var = nex.GetVar(doc, Neuron_Number, "neuron")
nex.Select(doc, Neuron_Var)
```

NexScript

```
% create a string variable
filefilter = "C:\Data\*.nex"

% show the dialog to the user
res = Dialog(0., filefilter , "File Filter:", "string")
```

DialogEx

Shows a dialog that can be used to specify script parameters. Python only.

Syntax

```
DialogEx(dialogSpecString)
```

Parameters

Parameter	Type	Description
dialogSpecString	string	Dialog specification saved as JSON string. See example below.

Return

Returns JSON string representing a Python dictionary `dlgRes` with two fields:

- `dlgRes["dialogResult"]` is a boolean value (`True` if user pressed OK in dialog, `False` if user pressed Cancel in dialog).
- `dlgRes["parameters"]` is a list of dialog parameters with parameter values specified in the dialog.

Dialog parameter is a Python dictionary `par` with the following fields:

- `par["Type"]` - a string specifying parameter type. Should be `"Number"`, `"Integer"`, `"String"`, `"Boolean"` or `"Choice"`.
- `par["Name"]` - a string specifying parameter name in the dialog.

- `par["Description"]` - a string specifying parameter description in the dialog (shown in the lower panel of the dialog when a parameter is selected).
- `par["Value"]` - parameter value. Should be:
  - `float` for `"Number"` parameter type
  - `int` for `"Integer"` parameter type
  - `str` for `"String"` parameter type
  - `bool` for `"Boolean"` parameter type
  - `int` for `"Choice"` parameter type
- `par["List"]` - a list of choices if parameter type is `"Choice"`.

Examples

Python

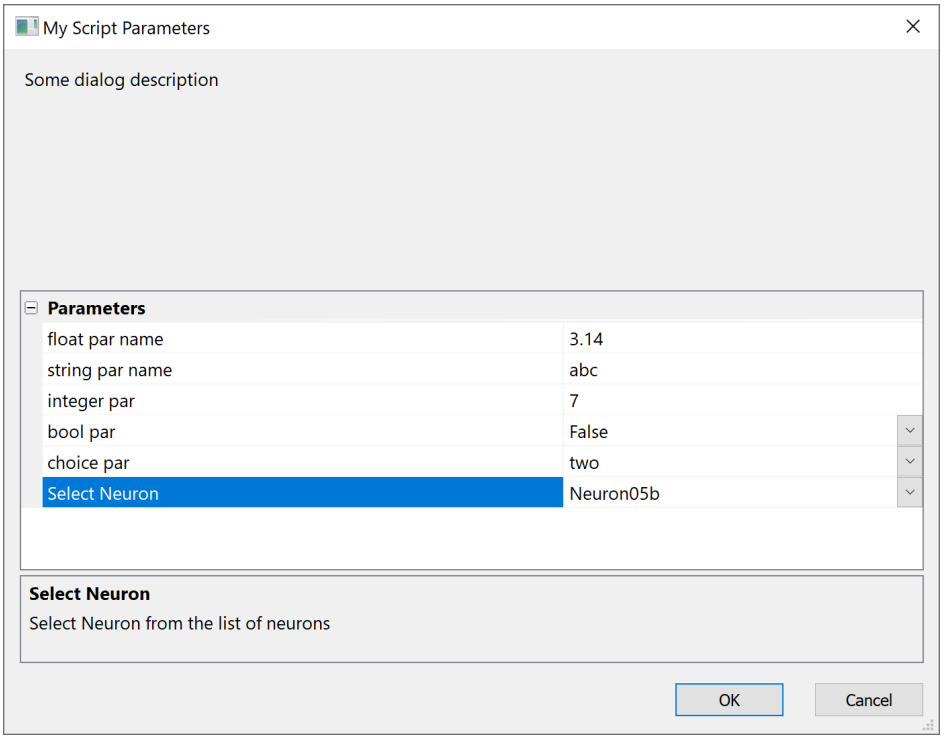
```
import nex
import json

def MakePar(typeString, nameString, descString, value, list=[]):
   """ Makes parameter dict used in nex.DialogEx. """
   return {"Type": typeString, "Name": nameString, "Description": descString,
↪"Value": value, "List": list }

p1 = MakePar("Number", "float par name", "float par description", 3.14 )
p2 = MakePar("String", "string par name", "string par description", "abc" )
p3 = MakePar("Integer", "integer par", "some description", 7 )
p4 = MakePar("Boolean", "bool par", "Bool description", False )
p5 = MakePar("Choice", "choice par", "Trying choice type", 1, ["one", "two",
↪"three" ] )

doc = nex.GetActiveDocument()
p6 = MakePar("Choice", "Select Neuron", "Select Neuron from the list of␣
↪neurons", 1, doc.NeuronNames() )

pars = [p1, p2, p3, p4, p5, p6]

dlgPars = {"DialogDescription": "Some dialog description", "DialogTitle": "My␣
↪Script Parameters", "Parameters": pars}
res = nex.DialogEx(json.dumps(dlgPars))
```

(continues on next page)

(continued from previous page)

```
dialogResultDict = json.loads(res)
print(dialogResultDict)
```

The following dialog will be shown:



ExecuteMenuCommand

Executes specified menu command.

Syntax

```
ExecuteMenuCommand(menuName)
```

Parameters

Parameter	Type	Description
menuName	string	Menu name. Submenus are separated by vertical bar (|). For example: `ExecuteMenuCommand('SavedResults|Quick Save Results')`.

Return

None.

Note: Some menu commands require user interaction. These menu commands have ellipsis (...) at the end of the menu title (for example, `'File|Open...'`). Since user interaction is required, these commands cannot be used in a script. Use `Script|History Script` menu command view these commands with the proper command parameters.

The following menu commands are supported:

```
File|New
File|Restore Last Analysis
File|Load Selected Cont. Channels from 3Brain Files
File|Close
File|Save
File|Connect to Plexon Server
View|1D Data Viewer Window
View|Numerical Results Window
View|Average/Overlay Chart Window
View|Results Folder Summary
Analysis|Increase X Range
Analysis|Decrease X Range
Results|Graphical Results|Copy Graphics to the Clipboard
Results|Numerical Results|View Numerical Results Window
Results|Numerical Results|Copy Numerical Results to the Clipboard
```

(continues on next page)

(continued from previous page)

```
Results|Numerical Results|Add Numerical Results as New Continuous Variables
Results|Numerical Results|Send Numerical Results to Matlab
SavedResults|Quick Save Results
SavedResults|Results Folder Summary
Graphics|Export Graphics|Copy Graphics to the Clipboard
Graphics|Fonts|Enable Font AutoScale
Graphics|Fonts|Increase font size
Graphics|Fonts|Decrease font size
Graphics|Zoom|Fit to Window
Template|Save As Default Template
3DView|View Histograms in 3D
3DView|View Histogram Variations in 3D
Matlab|Get Data From Matlab|Open Matlab As Engine
Matlab|Send Selected Variables to Matlab
Matlab|Send Numerical Results to Matlab
Online|Connect to Plexon Server
Online|Connect to Cerebus Server
Online|Connect to Neuralynx Server
Online|Connect to AlphaMap Server
Online|Pause Online Data Server Updates
Online|Reset Online Data File
Online|Disconnect from Online Data Server
Window|Numerical Results Window
Window|Close All Windows
Window|Close All Windows Except Currently Active Window
```

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "PerieventHistograms")
nex.ExecuteMenuCommand("SavedResults|Quick Save Results")
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "PerieventHistograms")
ExecuteMenuCommand("SavedResults|Quick Save Results")
```

ActivateWindow

Activates specified child window.

Syntax

```
ActivateWindow(docPath, windowName)
```

Parameters

Parameter	Type	Description
docPath	string	Document (data file) path.
windowName	string	Window name, for example [Graphs1]

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
docPath = nex.GetDocPath(doc)
nex.ApplyTemplate(doc,'Autocorrelograms')
nex.ActivateWindow(docPath, '[Graphs1]')
```

NexScript

```
doc = GetActiveDocument()
docPath = GetDocPath(doc)
ApplyTemplate(doc,"Autocorrelograms")
ActivateWindow(docPath, "[Graphs1]")
```

CloseWindow

Closes specified child window.

Syntax

```
CloseWindow(docPath, windowName)
```

Parameters

Parameter	Type	Description
docPath	string	Document (data file) path.
windowName	string	Window name, for example [Graphs1

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
docPath = nex.GetDocPath(doc)
nex.ApplyTemplate(doc,'Autocorrelograms')
nex.CloseWindow(docPath, '[Graphs1]')
```

NexScript

```
doc = GetActiveDocument()
docPath = GetDocPath(doc)
ApplyTemplate(doc,"Autocorrelograms")
CloseWindow(docPath, "[Graphs1]")
```

CloseNonDataWindows

Closes all analysis, 1D view or 3D view windows of the specified document. Python only.

Syntax

```
CloseNonDataWindows(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.CloseNonDataWindows(doc)
```

3.3.19 Matlab Functions

SendSelectedVarsToMatlab

Sends selected variables to Matlab.

Syntax

```
SendSelectedVarsToMatlab(doc)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
nex.SelectAllNeurons(doc)
nex.SendSelectedVarsToMatlab(doc)
```

NexScript

```
doc = GetActiveDocument()
DeselectAll(doc)
SelectAllNeurons(doc)
SendSelectedVarsToMatlab(doc)
```

ExecuteMatlabCommand

Sends the string command to Matlab and executes the command in Matlab.

Syntax

```
ExecuteMatlabCommand(command)
```

Parameters

Parameter	Type	Description
command	string	Matlab command to be run

Return

Returns a string with the result of the executed Matlab command.

Note: Any valid Matlab command that you can type at Matlab prompt can be used. For example, you can call a Matlab script or a function.

Examples

Python

```
import nex
nex.ExecuteMatlabCommand("x=randn(10,1);plot(x)")
varList = nex.ExecuteMatlabCommand("who")
```

NexScript

```
ExecuteMatlabCommand("x=randn(10,1);plot(x)")
varList = ExecuteMatlabCommand("who")
```

GetVarFromMatlab

Imports the specified Matlab timestamps vector as neuron or event variable.

Syntax

```
GetVarFromMatlab(doc, varname, isneuron)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
varname	string	The name of the matrix in Matlab workspace. The matrix should be a number matrix with either one row or one column of data containing timestamps in seconds.
isneuron	number	If isneuron is 1, the imported variable is added to the list of Neurons. Otherwise, the variable is added to the list of events.

Return

None.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# get the 1-column matrix ev1 and add it to the list of events
# for testing purposes, we create ev1 matrix using ExecuteMatlabCommand
nex.ExecuteMatlabCommand('ev1 = [0.5;2.3;4.7]')
nex.GetVarFromMatlab(doc, "ev1", 0)
```

NexScript

```
doc = GetActiveDocument()
% get the 1-column matrix ev1 and add it to the list of events
% for testing purposes, we create ev1 matrix using ExecuteMatlabCommand
ExecuteMatlabCommand("ev1 = [0.5;2.3;4.7]")
GetVarFromMatlab(doc, "ev1", 0)
```

GetContVarFromMatlab

Imports the specified matrix from Matlab. Each column of the matrix is imported as a continuous variable.

Syntax

```
GetContVarFromMatlab(doc, MatrixName, TimestampOfFirstValue, TimeStep)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
MatrixName	string	The name of a matrix in Matlab workspace.
TimestampOfFirstValue	number	The timestamp (in seconds) of the first value of each continuous variable

continues on next page

Table 236 – continued from previous page

Parameter	Type	Description
TimeStep	number	Digitizing time step (in seconds) of the imported variables.

Return

None.

Note: This function adds continuous variables to the specified document. The names of the variables include the MatrixName and the column number.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# import matrix contData from Matlab. first timestamp is 0, time step is 0.
↪001s.
# for testing purposes, we create contData matrix using ExecuteMatlabCommand
nex.ExecuteMatlabCommand('contData = [1 2; 11 12]')
nex.GetContVarFromMatlab(doc, "contData", 0, 0.001)
```

NexScript

```
doc = GetActiveDocument()
% import matrix contData from Matlab. first timestamp is 0, time step is 0.
↪001s.
% for testing purposes, we create contData matrix using ExecuteMatlabCommand
ExecuteMatlabCommand("contData = [1 2; 11 12]")
GetContVarFromMatlab(doc, "contData", 0, 0.001)
```

GetContVarWithTimestampsFromMatlab

Imports the specified 2-column matrix containing continuous variable data from Matlab.

Syntax

```
GetContVarWithTimestampsFromMatlab(doc, MatrixName, UseFirstDeltaAsDigRate)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
MatrixName	string	The name of the Matrix in Matlab workspace. The first column of the matrix should contain the values of a continuous variable, the second column - the corresponding timestamps.
UseFirstDeltaAsDigRate	number	Valid values are 0 or 1. If this parameter is positive, the difference between the second and the first timestamp is used as the variable digitizing rate.

Return

None.

Note: This function adds a continuous variable to the specified document. The name of the new variable is the MatrixName.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# for testing purposes, we create contData matrix using ExecuteMatlabCommand
```

(continues on next page)

(continued from previous page)

```
nex.ExecuteMatlabCommand('contData = [1 2; 11 12]')
nex.GetContVarWithTimestampsFromMatlab(doc, "contData", 1)
```

NexScript

```
doc = GetActiveDocument()
% for testing purposes, we create contData matrix using ExecuteMatlabCommand
ExecuteMatlabCommand("contData = [1 2; 11 12]")
GetContVarWithTimestampsFromMatlab(doc, "contData", 1)
```

GetIntervalVarFromMatlab

Imports the specified matrix containing intervals from Matlab.

Syntax

```
GetIntervalVarFromMatlab(doc, MatrixName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
MatrixName	string	The name of the 2-column matrix in Matlab workspace

Return

None.

Note: Imports the specified 2-column matrix from Matlab. The first column of the matrix should contain interval start times in seconds, the second column - interval end times in seconds.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
# for testing purposes, we create trials matrix using ExecuteMatlabCommand
# add intervals [1,10], [11,22] and [35,40] (seconds)
nex.ExecuteMatlabCommand('trials = [1 10; 11 22; 35 40]')
nex.GetIntervalVarFromMatlab(doc, "trials")
```

NexScript

```
doc = GetActiveDocument()
% for testing purposes, we create trials matrix using ExecuteMatlabCommand
ExecuteMatlabCommand("trials = [1 10; 11 22; 35 40]")
GetIntervalVarFromMatlab(doc, "trials")
```

3.3.20 Excel Functions

SetExcelCell

Sets the text value of the specified cell in Excel.

Syntax

```
SetExcelCell(worksheet, cell, text, excelFilePath)
```

Parameters

Parameter	Type	Description
worksheet	string	The name of the worksheet

continues on next page

Table 239 – continued from previous page

Parameter	Type	Description
cell	string	Excel cell specification, Should be in the form CR where C is Excel column name, R is the row number. For example, 'A1' is the top-left cell in the worksheet.
text	string	The text to be copied to the cell
excelFilePath	string	Full path of the Excel file. This parameter is optional. See Usage below.

Return

None.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
# this call will open Excel, Excel will create
# a new workbook (Excel file) and paste the text to the specified cell
nex.SetExcelCell("fromNex", "A1", "cell text")
# this call will set the cell in the specified Excel file
nex.SetExcelCell("fromNex", "A1", "cell text", "C:\\Data\\Results.xlsx")
```

NexScript

```
% this call will open Excel, Excel will create
% a new workbook (Excel file) and paste the text to the specified cell
SetExcelCell("fromNex", "A1", "cell text")

% this call will set the cell in the specified Excel file
SetExcelCell("fromNex", "A1", "cell text", "C:\Data\Results.xlsx")
```

SendResultsToExcel

Sends numerical results (of the first graphics window of the document) to Excel.

Syntax

```
SendResultsToExcel(doc, fileName, worksheetName, useFirstEmptyRow, cellName,␣
↪includeHeader, includeFileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
fileName	string	Excel file path
worksheetName	string	Excel worksheet name
useFirstEmptyRow	number	If 1, NeuroExplorer will ignore cellName parameter and add the results to the first row where the cell in column A is empty. If 2, NeuroExplorer will ignore cellName parameter and add the results to the first column where the cell in row 1 is empty. If 0, NeuroExplorer will copy the data starting with the cell specified in **cellName.**
cellName	string	Excel cell where to copy the results. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.
includeHeader	number	If 1, NeuroExplorer will paste column names.
includeFileName	number	If 1, NeuroExplorer will add a column with the data file name.

Return

None.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SendResultsToExcel(doc, "C:\\Data\\NexResults.xlsx", "Nex", 0, "A1", 1, 0)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
SendResultsToExcel(doc, "C:\Data\NexResults.xlsx", "Nex", 0, "A1", 1, 0)
```

SendResultsSummaryToExcel

Sends summary of numerical results (of the first graphics window of the document) to Excel.

Syntax

```
SendResultsSummaryToExcel(doc, fileName, worksheetName, useFirstEmptyRow,␣
↪cellName, includeHeader, includeFileName)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document
fileName	string	Excel file path
worksheetName	string	Excel worksheet name

continues on next page

Table 241 – continued from previous page

Parameter	Type	Description
useFirstEmptyRow	number	If 1, NeuroExplorer will ignore cellName parameter and add the results to the first row where the cell in column A is empty. If 2, NeuroExplorer will ignore cellName parameter and add the results to the first column where the cell in row 1 is empty. If 0, NeuroExplorer will paste data starting with the cell specified in **cellName.**
cellName	string	Excel cell where to paste the results. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.
includeHeader	number	If 1, will paste column names
includeFileName	number	If 1, will add a column with the data file name

Return

None.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SendResultsSummaryToExcel(doc, "C:\\Data\\res.xlsx", "FromNex", 0, "A1",␣
↪1, 0)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
SendResultsSummaryToExcel(doc, "C:\Data\res.xlsx", "FromNex", 0, "A1", 1, 0)
```

CloseExcelFile

Closes the specified Excel file if the file is open. Saves the file before closing.

Syntax

```
CloseExcelFile(filePath)
```

Parameters

Parameter	Type	Description
filePath	string	Full path of the Excel file

Return

None.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
```

(continues on next page)

(continued from previous page)

```
excelFilePath = r"C:\Data\NexResults.xlsx"
nex.SendResultsToExcel(doc, excelFilePath, "Nex", 0, "A1", 1, 0)
nex.CloseExcelFile(excelFilePath)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
excelFilePath = "C:\Data\NexResults.xlsx"
SendResultsToExcel(doc, excelFilePath, "Nex", 0, "A1", 1, 0)
CloseExcelFile(excelFilePath)
```

3.3.21 PowerPoint Functions

SendGraphicsToPowerPoint

Sends the contents of the first graphical window of the document to the specified PowerPoint presentation.

Syntax

```
SendGraphicsToPowerPoint(doc, presentationPath, slideTitle, comment,␣
↪addParameters, useBitmap)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
presentationPath	string	Path of the presentation file. File extension should be .ppt (NeuroExplorer cannot save .pptx files). If file extension is not .ppt, .ppt extension is added to the file path.
slideTitle	string	Slide title.
comment	string	Slide comment. Will be shown below graphics.

continues on next page

Table 243 – continued from previous page

Parameter	Type	Description
addParameters	number	If 1, add a text box with analysis parameter values.
useBitmap	number	If 1, transfer graphics as a bitmap, otherwise, transfer graphics as a metafile.

Return

None.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
nex.SendGraphicsToPowerPoint(doc, "C:\\Data\\NexResults.ppt", "Slide 1",
↪"Sample slide", 1, 0)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
SendGraphicsToPowerPoint(doc, "C:\Data\NexResults.ppt", "Slide 1", "Sample␣
↪slide", 1, 0)
```

ClosePowerPointFile

Closes the specified PowerPoint file if the file is open.

Syntax

```
ClosePowerPointFile(filePath)
```

Parameters

Parameter	Type	Description
filePath	string	Full path of the PowerPoint file

Return

None.

Note: See *Specifying Windows file paths in Python* for details on specifying file paths with backslashes in Python.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
nex.ApplyTemplate(doc, "Autocorrelograms")
pptFilePath = r"C:\Data\NexResults.ppt"
nex.SendGraphicsToPowerPoint(doc, pptFilePath, "Slide 1", "Sample slide", 1,␣
↪0)
nex.ClosePowerPointFile(pptFilePath)
```

NexScript

```
doc = GetActiveDocument()
ApplyTemplate(doc, "Autocorrelograms")
pptFilePath = "C:\Data\NexResults.ppt"
SendGraphicsToPowerPoint(doc, pptFilePath, "Slide 1", "Sample slide", 1, 0)
ClosePowerPointFile(pptFilePath)
```

3.3.22 Running Scripts

RunScript

Runs NexScript saved in a file with the specified name.

Syntax

```
RunScript(scriptName)
```

Parameters

Parameter	Type	Description
scriptName	string	The name of the script. See **Note** below for details on how to specify script name parameter

Return

None.

Note: How to specify script names.

When the script name is specified as `"MyScript"`, for example:

```
RunScript("MyScript")
```

NeuroExplorer will use the following script file

```
C:\Users\<USER>\Documents\NeuroExplorer 5\Scripts\MyScript.nsc
```

(where <USER> is your Windows user name).

You can also specify the script name relative to the main scripts folder

```
C:\Users\<USER>\Documents\NeuroExplorer 5\Scripts
```

For example, the script line

```
RunScript("Grant\MyScript.nsc")
```

means that the script file

```
C:\Users\<USER>\Documents\NeuroExplorer 5\Scripts\Grant\MyScript.nsc
```

will be used.

Finally, you can specify the absolute path of the script file:

```
RunScript("C:\MyScripts\Script.nsc")
```

In Python, you can run another **Python** script by defining a function and using import statement.

For example, if you have a script file MyScript1.py with the following contents:

```
def MyFunction():
   print('in MyFunction')
```

then, in another script file (in the same directory) you can call `MyFunction` using the following code:

```
from MyScript1 import MyFunction
MyFunction()
```

You can also run **NexScript** script from a Python script:

```
nex.RunScript('AnotherScript')
nex.RunScript('\\Grant\\AnotherScript.nsc')
nex.RunScript(r'C:\MyScripts\Grant\AnotherScript.nsc')
```

Examples

Python

```
import nex
# run NexScript
nex.RunScript(r'C:\Data\Scripts\ModifyTemplate.nsc')
```

NexScript

```
RunScript("MyOtherScript")
```

Sleep

Pauses execution of the script for nms milliseconds.

Syntax

```
Sleep(nms)
```

Parameters

Parameter	Type	Description
nms	number	The number of milliseconds to pause

Return

None.

Examples

Python

```
import nex
# pause for 2 seconds
nex.Sleep(2000)
```

NexScript

```
% pause for 2 seconds
Sleep(2000)
```

UsePython2

Selects Python version 2 as the running engine.

Syntax

```
UsePython2()
```

Parameters

None.

Return

None.

Examples

Python

```
import nex
nex.UsePython2()
```

UsePython3

Selects Python version 3 as the running engine.

Syntax

```
UsePython3()
```

Parameters

None.

Return

None.

Examples

Python

```
import nex
nex.UsePython3()
```

InstallPackageIfNeeded

If Python package is not installed (in NeuroExplorer python distribution), installs the package. Python 3 only.

Syntax

```
InstallPackageIfNeeded(packageName)
```

Parameters

Parameter	Type	Description
packageName	string	Package name

Return

None.

Examples

Python

```
import nex
# if we need scipy to use, for example, scipy signal processing functions,
# run the following code to install scipy:
nex.InstallPackageIfNeeded('scipy')
```

GetAppProperty

Returns specified global NeuroExplorer property.

Syntax

```
GetAppProperty(propertyName)
```

Parameters

Parameter	Type	Description
propertyName	string	Property name.

Return

Returns specified application property.

To retrieve a NeuroExplorer property specified in the Windows registry (`Computer\HKEY_CURRENT_USER\SOFTWARE\Nex Technologies\NeuroExplorer 5`) use property name `'Settings|Section|Name'`, where `Section` is a section of the main NeuroExplorer registry key and `Name` is the name of a key in the section. For example,

```
import nex
isUsingPython3 = nex.GetAppProperty('Settings|Python|EnablePython3')
```

Note: The following additional property names can be used:

- `'PythonAnalysisScriptParameters'`
- `'ScriptDirectory'`
- `'TemplateDirectory'`
- `'Version'`
- `'BuildDate'`

Examples

Python

```
import nex
scriptDirectory = nex.GetAppProperty('ScriptDirectory')
```

SetAppProperty

Sets application property.

Syntax

```
SetAppProperty(propertyName, propertyValue)
```

Parameters

Parameter	Type	Description
propertyName	string	The name of the property.
propertyValue	string	Property value.

To set a NeuroExplorer property specified in the Windows registry (`Computer\HKEY_CURRENT_USER\SOFTWARE\Nex Technologies\NeuroExplorer 5`) use property name `'Settings|Section|Name'`, where `Section` is a section of the main NeuroExplorer registry key and `Name` is the name of a key in the section. For example,

```
import nex
# enable saving time filter parameters in templates
nex.SetAppProperty('Settings|General|SaveFilterPars', '1')
```

Note: The following property names can also be used:

- `'PythonAnalysisScriptResult'`

Return

None.

Examples

Python

```
import nex
# enable saving time filter parameters in templates
nex.SetAppProperty('Settings|General|SaveFilterPars', '1')
```

3.3.23 Math Functions

seed

Initializes the random number generator with a given seed value.

Syntax

```
seed(iseed)
```

Parameters

Parameter	Type	Description
iseed	number	Numeric seed value. Should be a positive integer.

Return

None.

Note: In Python scripts, use Python built-in `random` module. For example, you can initialize the random number generator with a seed value using `random.seed(iseed)`. See (https://www.tutorialsteacher.com/python/random-module) for more details.

Examples

Python

```
import nex
nex.seed(1717)
```

NexScript

```
seed(1717)
```

rand

Returns a number with uniform random distribution from 0 to 1.

Syntax

```
rand()
```

Parameters

None.

Return

Returns a number with uniform random distribution from 0 to 1.

Note: In Python scripts, use Python built-in `random` module. See, for example, (https://www.tutorialsteacher.com/python/random-module) for details.

Examples

Python

```
import nex
r = nex.rand()
```

NexScript

```
doc = GetActiveDocument()
r = rand()
```

expo

Returns a random value exponentially distributed with the specified mean.

Syntax

```
expo(fmean)
```

Parameters

Parameter	Type	Description
fmean	number	The mean of the exponential distribution.

Return

Returns a random value exponentially distributed with the specified mean.

Examples

Python

```
import nex
y = nex.expo(10)
```

NexScript

```
y = expo(10)
```

floor

Returns the largest integer that is less than or equal to the specified number.

Syntax

```
floor(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value

Return

Returns the largest integer that is less than or equal to x.

Examples

Python

```
import math
x = 1.7
y = math.floor(x)
# y now is equal to 1
```

NexScript

```
x = 1.7
y = floor(x)
% y now is equal to 1
```

ceil

Returns the smallest integer that is greater than or equal to the specified number.

Syntax

```
ceil(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value

Return

Returns the smallest integer that is greater than or equal to x.

Examples

Python

```
import math
x = math.ceil(1.01)
# x now is equal to 2
```

NexScript

```
x = ceil(1.01)
% x now is equal to 2
```

round

Rounds the specified number to the nearest integer.

Syntax

```
round(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value

Return

Returns the integer that is closest to x

Examples

Python

```
y = round(1.3)
# y now is 1
y = round(1.6)
# y now is 2
```

NexScript

```
y = round(1.3)
% y now is 1
y = round(1.6)
% y now is 2
```

abs

Returns absolute value of the specified number.

Syntax

```
abs(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value

Return

Returns absolute value of x.

Examples

Python

```
# this script tests abs function
x =  - 2
ax = abs(x)
# ax is equal to 2
```

NexScript

```
% this script tests abs function
x = -2
ax = abs(x)
% ax is equal to 2
```

sqrt

Returns the square root of the specified number.

Syntax

```
sqrt(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (should be non-negative)

Return

Returns the square root of x.

Examples

Python

```
import math
y = math.sqrt(2)
```

NexScript

```
y = sqrt(2)
```

pow

Returns x raised to the power of y.

Syntax

```
pow(x, y)
```

Parameters

Parameter	Type	Description
x	number	Number to be raised to the specified power
y	number	The power.

Return

Returns x raised to the power of y.

Examples

Python

```
import math
z = math.pow(2, 3)
# z now is equal to 8
```

NexScript

```
z = pow(2, 3)
% z now is equal to 8
```

exp

Returns exponential of x.

Syntax

```
exp(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value

Return

Returns exponential of x.

Examples

Python

```
import math
y = math.exp(2.5)
```

NexScript

```
y = exp(2.5)
```

min

Returns the minimum of two numbers.

Syntax

```
min(x, y)
```

Parameters

Parameter	Type	Description
x	number	Numeric value
y	number	Numeric value

Return

Returns minimum of x and y.

Examples

Python

```
x = 4
y = 2
z = min(x, y)
```

NexScript

```
x = 4
y = 2
z = min(x, y)
```

max

Returns maximum of two numbers.

Syntax

```
max(x, y)
```

Parameters

Parameter	Type	Description
x	number	Numeric value
y	number	Numeric value

Return

Returns maximum of x and y.

Examples

Python

```
x = 7
y = max(x, 5)
```

NexScript

```
x = 7
y = max(x, 5)
```

log

Returns logarithm of the specified number.

Syntax

```
log(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (the value should be positive)

Return

Returns logarithm of x.

Examples

Python

```
import math
y = math.log(2.5)
```

NexScript

```
y = log(2.5)
```

sin

Returns sine of the specified number.

Syntax

```
sin(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (sine parameter in radians)

Return

Returns the sine of x.

Examples

Python

```
import math
x = 10
y = math.sin(x)
```

NexScript

```
x = 10
y = sin(x)
```

cos

Returns cosine of the specified number.

Syntax

```
cos(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (cosine parameter in radians)

Return

Returns cosine of x.

Examples

Python

```
import math
y = math.cos(0.5)
```

NexScript

```
y = cos(0.5)
```

tan

Returns tangent of the specified number.

Syntax

```
tan(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (tangent parameter in radians)

Return

Returns tangent of x.

Examples

Python

```
import math
y = math.tan(1)
```

NexScript

```
y = tan(1)
```

acos

Returns the arccosine (in radians) of the specified number.

Syntax

```
acos(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (the value should be from -1 to +1)

Return

Returns y such that x = cos(y).

Examples

Python

```
import math
x = 0.5
y = math.acos(x)
# y is 1.047197551
```

NexScript

```
x = 0.5
y = acos(x)
% y is 1.047197551
```

asin

Returns the arcsine of the specified number.

Syntax

```
asin(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value (the value should be from -1 to +1)

Return

Returns y such that x = sin(y).

Examples

Python

```
import math
x = 0.5
y = math.asin(x)
```

NexScript

```
x = 0.5
y = asin(x)
```

atan

Returns the arctangent of the specified number.

Syntax

```
atan(x)
```

Parameters

Parameter	Type	Description
x	number	Numeric value

Return

Returns y such that x = tan(y).

Examples

Python

```
import math
x = 2
y = math.atan(x)
```

NexScript

```
x = 2
y = atan(x)
```

BitwiseAnd

Returns the result of the bitwise AND operation.

Syntax

```
BitwiseAnd(value1, value2)
```

Parameters

Parameter	Type	Description
value1	number	Numeric value
value2	number	Numeric value

Return

Result of the bitwise AND operation.

Note: value1 and value2 are converted to integers and then bitwise AND operation is applied to these integers.

In Python scripts, use Python bitwise operations. See (https://wiki.python.org/moin/BitwiseOperators) for details.

Examples

Python

```
x = 7 & 1
# x now is equal to 1
```

NexScript

```
x = BitwiseAnd(7, 1)
% x now is equal to 1
```

BitwiseOr

Returns the result of the bitwise OR operation.

Syntax

```
BitwiseOr(value1, value2)
```

Parameters

Parameter	Type	Description
value1	number	Numeric value
value2	number	Numeric value

Return

Result of the bitwise OR operation.

Note: value1 and value2 are converted to integers and then the bitwise OR operation is applied to these integers.

In Python scripts, use Python bitwise operations. See (https://wiki.python.org/moin/BitwiseOperators) for details.

Examples

Python

```
x = 2 | 3
# x now is equal to 3
```

NexScript

```
x = BitwiseOr(2, 1)
% x now is equal to 3
```

GetBit

Returns the value of the specified bit (1 to 32).

Syntax

```
GetBit(x, bitNumber)
```

Parameters

Parameter	Type	Description
x	number	Numeric value.
bitNumber	number	1-based bin number. 1 is the least significant bit, 32 is the most significant bit

Return

Returns the value (0 or 1) of the specified bit.

Note: The first parameter is converted to an unsigned 32-bit integer and then the bit value of this unsigned integer is returned.

Examples

Python

```
import nex
# get the second bit of 3
b2 = nex.GetBit(3, 2)
# b2 is now equal to 1
```

NexScript

```
% get the second bit of 3
b2 = GetBit(3, 2)
% b2 is now equal to 1
```

RoundToTS

Rounds the specified number to the nearest timestamp value.

Syntax

```
RoundToTS(doc, time)
```

Parameters

Parameter	Type	Description
doc	documentReference	Reference to the document.
time	number	The time value (in seconds) to be rounded

Return

The document timestamp value (in seconds) nearest to the specified time.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
ts = nex.RoundToTS(doc, 1.234)
```

NexScript

```
doc = GetActiveDocument()
ts = RoundToTS(doc, 1.234)
```

GetFirstGE

Returns the index of the first timestamp in the specified variable that is greater than or equal to the specified number.

Syntax

```
GetFirstGE(var, time)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable
time	number	Time value in seconds.

Return

Returns the index of the first timestamp in the specified variable that is greater than or equal to the specified number.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
neuron = doc["Neuron04a"]
# get the index of the first spike at or after 5.7s
index = nex.GetFirstGE(neuron, 5.7)
```

NexScript

```
doc = GetActiveDocument()
neuron = doc["Neuron04a"]
% get the index of the first spike at or after 5.7s
index = GetFirstGE(neuron, 5.7)
```

GetFirstGT

Returns the index of the first timestamp in the specified variable that is greater than the specified number.

Syntax

```
GetFirstGT(var, time)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to the variable
time	number	Time value in seconds.

Return

Returns the index of the first timestamp in the specified variable that is greater than the specified number.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
neuron = doc["Neuron04a"]
# get the index of the first spike after 5.7s
index = nex.GetFirstGT(neuron, 5.7)
```

NexScript

```
doc = GetActiveDocument()
neuron = doc["Neuron04a"]
% get the index of the first spike after 5.7s
index = GetFirstGT(neuron, 5.7)
```

GetBinCount

Calculates the number of timestamps in the specified time range.

Syntax

```
GetBinCount(var, timeMin, timeMax)
```

Parameters

Parameter	Type	Description
var	variableReference	Reference to a variable. Should be a reference to a neuron, event, interval or continuous variable
timeMin	number	Time range minimum (in seconds).
timeMax	number	Time range maximum (in seconds).

Return

The number of timestamps of the specified variable in the specified time range.

Examples

Python

```
import nex
doc = nex.GetActiveDocument()
neuron = doc["Neuron04a"]
# calculate how many timestamps of Neuron04a are in the interval [5.3s, 10.2s]
count = nex.GetBinCount(neuron, 5.3, 10.2)
```

NexScript

```
doc = GetActiveDocument()
neuron = doc["Neuron04a"]
% calculate how many timestamps of Neuron04a are in the interval [5.3s, 10.2s]
count = GetBinCount(neuron, 5.3, 10.2)
```

PoissonSurprise

Returns Poisson Surprise - the probability that `k` or more spikes occur randomly in a period of length `time`

Syntax

```
PoissonSurprise(k, freq, time)
```

Parameters

Parameter	Type	Description
k	number	Number of spikes
freq	number	Firing rate (spikes per second)
time	number	Time period duration

Return

Returns Poisson Surprise - the probability that `k` or more spikes occur randomly in a period of length `time`

Examples

Python

```
import nex
k = 3
freq = 10
time = 2
surprise = nex.PoissonSurprise(k, freq, time)
```

3.3.24 String Functions

Left

Returns a substring that starts at the beginning of the string.

Syntax

```
Left(string, nchar)
```

Parameters

Parameter	Type	Description
string	string	String parameter.
nchar	number	Number of characters in the substring

Return

Returns a substring that starts at the beginning of the string.

Note: In Python, use string slicing: https://pythoncentral.io/cutting-and-slicing-strings-in-python/

Examples

Python

```
s = 'abcdefg'
sub = s[:3]
# sub now is "abc"
```

NexScript

```
sub = Left("abcdefg", 3)
% sub now is "abc"
```

Mid

Returns the specified substring.

Syntax

```
Mid(string, nstartchar, nchar)
```

Parameters

Parameter	Type	Description
string	string	String parameter.
nstartchar	number	1-based index of the start character
nchar	number	Number of characters to select.

Return

Returns the substring that starts at character nstartchar and contains nchar characters.

Note: In Python, use string slicing: https://pythoncentral.io/cutting-and-slicing-strings-in-python/

Examples

Python

```
s = 'abcdefg'
sub = s[2:2+3]
# sub now is "cde"
```

NexScript

```
sub = Mid("abcdefg", 2, 3)
% sub now is "cde"
```

Right

Returns a substring that ends at the end of the string.

Syntax

```
Right(string, nchar)
```

Parameters

Parameter	Type	Description
string	string	String parameter.
nchar	number	Number of characters in the substring

Return

Extracts right nchar characters from string, returns string.

Note: In Python, use string slicing: https://pythoncentral.io/cutting-and-slicing-strings-in-python/

Examples

Python

```
s = 'abcdefg'
sub = s[-3:]
# sub now is "efg"
```

NexScript

```
sub = Right("abcdefg", 3)
% sub now is "efg"
```

Find

Looks for a substring inside a specified string.

Syntax

```
Find(string1, string2)
```

Parameters

Parameter	Type	Description
string1	string	The string where we look for a substring
string2	string	The substring that we are trying to find

Return

Looks for a string string2 inside the string string1, returns a number - 1-based position of the first character of string2 in the string1. Returns zero is string2 is not found.

Note: In Python, use find() method of a string object: https://www.tutorialspoint.com/python/string_find.htm.

Examples

Python

```
import nex
# this script selects only the neurons that have "05" in their name
```

(continues on next page)

(continued from previous page)

```
doc = nex.GetActiveDocument()
nex.DeselectAll(doc)
for i in range( 1, nex.GetVarCount(doc, "neuron") + 1):
    name = nex.GetVarName(doc, i, "neuron")
    if name.find("05") >= 0:
        nex.SelectVar(doc, i, "neuron")
```

NexScript

```
% this script selects only the neurons that have "05" in their name
doc = GetActiveDocument()
DeselectAll(doc)
for i=1 to GetVarCount(doc, "neuron")
    name = GetVarName(doc, i, "neuron")
    if Find(name, "05")> 0
        SelectVar(doc, i, "neuron")
    end
end
```

StrLength

Calculates the number of characters in the string.

Syntax

```
StrLength(stringVal)
```

Parameters

Parameter	Type	Description
stringVal	string	The string parameter

Return

Returns the number of characters in the string.

Examples

Python

```
import nex
n = nex.StrLength("abcd")
```

NexScript

```
n = StrLength("abcd")
```

NumToStr

Converts number to string using an optional format string.

Syntax

```
NumToStr(number, formatString)
```

Parameters

Parameter	Type	Description
number	number	Number to be converted to string.
formatString	string	Optional format string (a standard C/C++ format specifier). See, for example, https://www.cplusplus.com/reference/clibrary/cstdio/printf

Return

Returns string representing the specified number.

Examples

The following scripts generate strings:

`Event001`, `Event002`, ..., `Event016`

Python

```
import nex
for i in range(1, 17):
    str = "Event0" + nex.NumToStr(i, "%02.0f")
```

NexScript

```
for i=1 to 16
    str = "Event0" + NumToStr(i, "%02.0f")
end
```

StrToNum

Converts string to number.

Syntax

```
StrToNum(stringRepresentingNumber)
```

Parameters

Parameter	Type	Description
stringRepresentingNumber	string	A string containing a valid representation of the number, for example, '1', '002', '123.456'

Return

Returns the number corresponding to the specified string.

Examples

Python

```
import nex
x = nex.StrToNum("003")
# x now is equal to 3
```

NexScript

```
x = StrToNum("003")
% x now is equal to 3
```

GetNumFields

Returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas.

Syntax

```
GetNumFields(stringWithFields)
```

Parameters

Parameter	Type	Description
stringWithFields	string	The string containing fields. The field is a substring that does not contain spaces, tabs or commas

Return

Returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas.

Note: In Python, use split() method of string object: https://www.tutorialspoint.com/python/string_split.htm .

Examples

Python

```
import nex
numFields = nex.GetNumFields("One two 3 4")
# numFields is now equal to 4
```

NexScript

```
numFields = GetNumFields("One two 3 4")
% numFields is now equal to 4
```

GetField

Returns the string field.

Syntax

```
GetField(stringWithFields, fieldnumber)
```

Parameters

Parameter	Type	Description
stringWithFields	string	The string containing multiple data fields. The field is a substring that does not contain spaces, tabs or commas
fieldnumber	number	The field number.

Return

Returns the field with the specified number as a string. The field is a substring that does not contain spaces, tabs or commas. For example,

`GetField("One two 3 4", 3)`returns "3".

Note: In Python, use split() method of string object: https://www.tutorialspoint.com/python/string_split.htm .

Examples

Python

```
import nex
secondField = nex.GetField("Neuro04a Neuron05b", 2)
# secondField now is equal to "Neuron05b"
```

NexScript

```
secondField = GetField("Neuro04a Neuron05b", 2)
% secondField now is equal to "Neuron05b"
```

CharToNum

Converts a one-character string to a number (a character's ASCII code).

Syntax

```
CharToNum(oneCharString)
```

Parameters

Parameter	Type	Description
oneCharString	string	A string of length 1 (for example, 'a', '5')

Return

The string character's ASCII code.

Note: In Python script, use python function `ord()`.

Examples

Python

```
x = ord("1")
# x now is equal to 49
```

NexScript

```
x = CharToNum("1")
% x now is equal to 49
```

NumToChar

Converts a number to a one-character string containing a character with the ASCII code equal to the number.

Syntax

```
NumToChar(number)
```

Parameters

Parameter	Type	Description
number	number	ASCII code of the character

Return

Returns a one-character string containing the character with the ASCII code equal to the specified number.

Examples

Python

```
import nex
oneAsString = nex.NumToChar(49)
# oneAsString now is equal to "1"
```

NexScript

```
oneAsString = NumToChar(49)
% oneAsString now is equal to "1"
```

3.3.25 Debug Functions

Trace

Prints arguments to the output window. In **Python**, use **print()** function instead.

Syntax

```
Trace(arg1, arg2, ...)
```

Parameters

Parameter	Type	Description
arg1	any type	String, number or any other valid NexScript value
arg2	any type	String, number or any other valid NexScript value

Return

None.

Note: Converts each parameter to string and the prints the result to the output window. In **Python**, use **print()** function instead.

Examples

Python

```
x = 30
# print the value of x
print("x={}".format(x))
```

NexScript

```
x = 30
% print the value of x
Trace("x=", x)
```

MsgBox

This function is equivalent to *Trace* function. In **Python**, use **print()** function instead.

Prints arguments to the output window.

Syntax

```
MsgBox(arg1, arg2, ...)
```

Parameters

Parameter	Type	Description
arg1	any type	String, number or any other valid NexScript value
arg2	any type	String, number or any other valid NexScript value

Return

None.

Note: Converts each parameter to string and the prints the result to the output window. In **Python**, use **print()** function instead.

Examples

Python

```
x = 30
# print the value of x
print("x={}".format(x))
```

NexScript

```
x = 30
% print the value of x
Trace("x=", x)
```

3.4 Controlling NeuroExplorer from Matlab

NeuroExplorer exposes COM (Component Object Model, or ActiveX) interfaces that allow other applications to launch and control NeuroExplorer.

For example, the following Matlab code starts NeuroExplorer, opens a data file and loads Neuron04a timestamps into the Matlab workspace:

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
neuron = doc.Variable('Neuron04a');
timestamps = neuron.Timestamps();
```

The table below lists the objects that NeuroExplorer exposes via COM methods and properties.

3.4.1 Application

External applications can open an instance of NeuroExplorer using its Program ID **'NeuroExplorer.Application'**.

For example, to open NeuroExplorer (or to connect to a running instance of NeuroExplorer) in Matlab , you can use the following command:

```
nex = actxserver('NeuroExplorer.Application');
```

The object returned by the `actxserver` command is an Application object. The properties and methods of this object are listed below.

ActiveDocument

Read-only property that returns Document object that represents the active document (the document corresponding to the active window of the application). Returns null if there are no open documents.

Syntax

```
ActiveDocument
```

Return

Returns Document object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.ActiveDocument;
```

DocumentCount

Read-only property that returns the number of open documents (data files).

Syntax

```
DocumentCount
```

Return

Returns the number of open documents (data files).

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
numDocs = nex.DocumentCount;
```

Document

Returns Document object for the specified document index.

Syntax

```
Document(documentIndex)
```

Parameters

Parameter	Type	Description
documentIndex	int	1-based document index

Return

Returns Document object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
% get the first opened document
doc = nex.Document(1);
```

OpenDocument

Opens the specified data file. Returns Document object if succeeded.

Syntax

```
OpenDocument(documentPath)
```

Parameters

Parameter	Type	Description
documentPath	string	Full data file path

Return

Returns Document object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
```

RunNexScript

Runs the specified NexScript. Returns true if succeeded.

Syntax

```
RunNexScript(scriptPath)
```

Parameters

Parameter	Type	Description
scriptPath	string	Script path

Return

Returns true if script succeeded, otherwise, returns false.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
res = nex.RunNexScript('C:\Data\Scripts\TestScript.nsc');
```

RunNexScriptCommands

Runs the specified NexScript text. Returns true if succeeded.

Syntax

```
RunNexScriptCommands(script)
```

Parameters

Parameter	Type	Description
script	string	Script text. Script lines should be separated by \n

Return

Returns true if the script succeeded, otherwise, returns false.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
res = nex.RunNexScriptCommands('doc=GetActiveDocument()\ndoc.NewEvent =␣
↪Sync(doc["Neuron04a"], doc["Neuron05b"], -0.01, 0.01)')
res = nex.RunNexScriptCommands('doc=GetActiveDocument()\nApplyTemplate(doc,
↪"AutoCorrelograms")');
```

Version

Read-only property that returns a string with the current version of NeuroExplorer.

Syntax

```
Version
```

Return

Returns a string with the current version of NeuroExplorer.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
version = nex.Version;
```

Visible

Boolean read/write property that controls the visibility of NeuroExplorer.

Syntax

```
Visible
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
isVisible = nex.Visible;
% make sure that NeuroExplorer is visible
nex.Visible = true;
```

Sleep

Pauses the application.

Syntax

```
Sleep(millisecondsToSleep)
```

Parameters

Parameter	Type	Description
millisecondsToSleep	int	Number of milliseconds to sleep

Return

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
% pause NeuroExplorer for 1 second
nex.Sleep(1000);
```

3.4.2 Document

NeuroExplorer Application object provides access to the open documents. For example, to open a document in NeuroExplorer in Matlab script, you can use:

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
```

Here OpenDocument method returns a Document object corresponding to the specified data file.

The properties and methods of the Document object are listed below.

Path

Read-only property that returns a string with the full path of the document.

Syntax

```
Path
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
path = doc.Path;
```

FileName

Read-only property that returns a string with the file name.

Syntax

```
FileName
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
path = doc.Path;
fileName = doc.FileName;
% path is 'C:\Data\MyDataFile.nex'
% fileName is 'MyDataFile.nex'
```

Comment

Read-only property that returns a string with the data file comment.

Syntax

```
Comment
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
comment = doc.Comment;
```

TimestampFrequency

Read-only property that returns the timestamp frequency of the document in Hertz.

Syntax

```
TimestampFrequency
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
tsFrequency = doc.TimestampFrequency;
```

StartTime

Read-write property that specifies the document data start time (minimum timestamp) in seconds.

Syntax

```
StartTime
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
startTime = doc.StartTime;
```

EndTime

Read-write property that specifies the document data end time (maximum timestamp) in seconds.

Syntax

```
EndTime
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
endTime = doc.EndTime;
```

VariableCount

Read-only property that returns the number of variables in the document.

Syntax

```
VariableCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numVars = doc.VariableCount;
```

NeuronCount

Read-only property that returns the number of neurons in the document.

Syntax

```
NeuronCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numNeurons = doc.NeuronCount;
```

EventCount

Read-only property that returns the number of event variables in the document.

Syntax

```
EventCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numEventVars = doc.EventCount;
```

IntervalCount

Read-only property that returns the number of interval variables in the document.

Syntax

```
IntervalCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numIntervalVars = doc.IntervalCount;
```

MarkerCount

Read-only property that returns the number of marker variables in the document.

Syntax

```
MarkerCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numMarkerVars = doc.MarkerCount;
```

WaveCount

Read-only property that returns the number of waveform variables in the document.

Syntax

```
WaveCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numWaveVars = doc.WaveCount;
```

ContinuousCount

Read-only property that returns the number of continuous variables in the document.

Syntax

```
ContinuousCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numContinuousVars = doc.ContinuousCount;
```

Variable

Returns Variable object for the specified variable index or name.

Syntax

```
Variable(variableIndexOrName)
```

Parameters

Parameter	Type	Description
variableIndexOr-Name	int or string	1-based variable index or a string with the variable name

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first variable
var1 = doc.Variable(1);
% get the variable with the name Neuron04a
neuron = doc.Variable('Neuron04a');
```

Neuron

Returns *Variable* object for the specified neuron variable index.

Syntax

```
Neuron(neuronIndex)
```

Parameters

Parameter	Type	Description
neuronIndex	int	1-based neuron variable index

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
% get the last neuron variable
neuronLast = doc.Neuron(doc.NeuronCount);
```

Event

Returns *Variable* object for the specified event variable index.

Syntax

```
Event(eventIndex)
```

Parameters

Parameter	Type	Description
eventIndex	int	1-based event variable index

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first event variable
Event1 = doc.Event(1);
% get the last event variable
EventLast = doc.Event(doc.EventCount);
```

Marker

Returns *Variable* object for the specified marker variable index.

Syntax

```
Marker(markerIndex)
```

Parameters

Parameter	Type	Description
markerIndex	int	1-based marker variable index

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first marker variable
Marker1 = doc.Marker(1);
% get the last marker variable
MarkerLast = doc.Marker(doc.MarkerCount);
```

Interval

Returns *Variable* object for the specified interval variable index.

Syntax

```
Interval(IntervalIndex)
```

Parameters

Parameter	Type	Description
IntervalIndex	int	1-based interval variable index

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first interval variable
Interval1 = doc.Interval(1);
% get the last interval variable
IntervalLast = doc.Interval(doc.IntervalCount);
```

Wave

Returns *Variable* object for the specified waveform variable index.

Syntax

```
Wave(waveIndex)
```

Parameters

Parameter	Type	Description
waveIndex	int	1-based waveform variable index

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first waveform variable
Wave1 = doc.Wave(1);
% get the last waveform variable
WaveLast = doc.Wave(doc.WaveCount);
```

Continuous

Returns *Variable* object for the specified continuous variable index.

Syntax

```
Continuous(continuousIndex)
```

Parameters

Parameter	Type	Description
continuousIndex	int	1-based continuous variable index

Return

Returns *Variable* object.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
Continuous1 = doc.Continuous(1);
% get the last continuous variable
ContinuousLast = doc.Continuous(doc.ContinuousCount);
```

DeselectAll

Deselects all the data variables in the document.

Syntax

```
void DeselectAll()
```

Return

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select one variable
doc.Variable('Neuron01').Select();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
```

SelectAllNeurons

Selects all the neuron variables in the document. Selected variables are used in analysis when `ApplyTemplate` document method is called.

Syntax

```
SelectAllNeurons()
```

Return

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all neurons
doc.SelectAllNeurons();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;
```

SelectAllContinuous

Selects all the continuous variables in the document.

Syntax

```
SelectAllContinuous()
```

Return

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all continuous variables
doc.SelectAllContinuous();
% run Perievent Histogram analysis saved in 'PSTH' template
doc.ApplyTemplate('PSTH');
% get numerical results
results = doc.GetNumericalResults();
```

ApplyTemplate

Runs the analysis specified in the analysis template.

Syntax

```
ApplyTemplate(templateName)
```

Parameters

Parameter	Type	Description
templateName	string	template name

Return

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all neurons
doc.SelectAllNeurons();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;
```

GetNumericalResults

Returns 2-dimensional array of numerical results for the first graph view of the document.

Syntax

```
GetNumericalResults()
```

Return

Returns 2-dimensional array of numerical results.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% apply AutoCorrelograms analysis template
nex.RunNexScriptCommands('doc=GetActiveDocument()\nApplyTemplate(doc,
↪"AutoCorrelograms")');
% get numerical results
results = doc.GetNumericalResults();
```

Close

Closes the document.

Syntax

```
Close()
```

Parameters

None.

Return

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% close the document
doc.Close();
```

3.4.3 Variable

NeuroExplorer Application object provides access to the open files and variables contained in the files. For example, to open a document in NeuroExplorer and get the first event variable in the document in Matlab, you can use:

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
event1 = doc.Event(1);
```

Here Event() method returns a Variable object corresponding to the first event variable in the file.

The properties and methods of the Variable object are listed below.

Name

Read-only property that returns a string with the variable name.

Syntax

```
Name
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
neuron1Name = neuron1.Name;
```

TimestampCount

Read-only property that returns the number of timestamps in the variable. For interval variables, returns the number of intervals. For waveforms variable, returns the number of waveforms. For continuous variables, returns the total number of data points.

Syntax

```
TimestampCount
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
numTimestamps = neuron1.TimestampCount;
```

SamplingRate

Read-only property that returns the sampling rate (in Hz) of a continuous or waveform variable. For other variable types, returns zero.

Syntax

```
SamplingRate
```

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1= doc.Continuous(1);
% get sampling rate
samplingRate = cont1.SamplingRate;
```

Timestamps

Returns all the timestamps of the variable in seconds. Timestamps are returned as an array (vector) of double values. For interval variables, returns the interval starts. For waveforms variable, returns the waveform timestamps. For continuous variables, returns the timestamps corresponding to all the variable data points.

Syntax

```
Timestamps()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
% get all the timestamps
ts = neuron1.Timestamps();
% now ts is a vector of timestamps
%
% get the first continuous variable
cont1 = doc.Continuous(1);
% get all the timestamps for continuous variable
cont1_ts = cont1.Timestamps();
```

IntervalStarts

For an interval variable, returns the interval start values in seconds. This method is valid only for interval variables.

Syntax

```
IntervalStarts()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first interval variable
int1= doc.Interval(1);
% get all the interval starts
int1Starts = int1.IntervalStarts();
```

IntervalEnds

For an interval variable, returns the interval end values in seconds. This method is valid only for interval variables.

Syntax

```
IntervalEnds()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first interval variable
int1= doc.Interval(1);
% get all the interval ends
int1Ends = int1.IntervalEnds();
```

FragmentTimestamps

For continuous variable, returns the fragment timestamp values in seconds. This method is valid only for continuous variables.

In general, a continuous variable may contain several fragments of data. Each fragment may be of a different length. NeuroExplorer does not store the timestamps for all the A/D values since they would use too much space. Instead, for each fragment, it stores the timestamp of the first A/D value in the fragment and the index of the first data point in the fragment. The timestamps of the first A/D value in each fragment are returned by this method.

Syntax

```
FragmentTimestamps()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1= doc.Continuous(1);
% get all the fragment timestamps
cont1FragmentTs = cont1.FragmentTimestamps();
```

FragmentCounts

For continuous variable, returns the number of data points in fragments. This method is valid only for continuous variables.

In general, a continuous variable may contain several fragments of data. Each fragment may be of a different length. NeuroExplorer does not store the timestamps for all the A/D values since they would use too much space. Instead, for each fragment, it stores the timestamp of the first A/D value in the fragment and the index of the first data point in the fragment. The number of data points in each fragment are returned by this method.

Syntax

```
FragmentCounts()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1= doc.Continuous(1);
% get all the fragment counts
cont1FragmentCounts = cont1.FragmentCounts();
```

ContinuousValues

For continuous variable, returns all the A/D values in milliVolts. This method is valid only for continuous variables.

Syntax

```
ContinuousValues()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1= doc.Continuous(1);
% get all the values
cont1Values = cont1.ContinuousValues();
```

MarkerValues

For a marker variable, returns all the marker values as strings. The values are returned in a two-dimensional array. Each row of the array represents all the marker strings for one timestamp.

This method is valid only for marker variables.

Syntax

```
MarkerValues()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first markervariable
marker1 = doc.Marker(1);
% get all the marker values
marker1Values = marker1.MarkerValues();
```

WaveformValues

For waveform variable, returns all the waveform values in milliVolts. The values are returned in a two-dimensional array. Each row of the array represents one waveform.

This method is valid only for waveform variables.

Syntax

```
WaveformValues()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first waveform variable
wave1= doc.Wave(1);
% get all the values
wave1Values = wave1.WaveformValues();
```

Select

Selects the variable. Selected variables are used in analysis when `ApplyTemplate` document method is called.

Syntax

```
Select()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select variable with the name 'Neuron01'
doc.Variable('Neuron01').Select();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;
```

Deselect

Deselects the variable. Only selected variables are used in analysis when `ApplyTemplate` document method is called.

Syntax

```
Deselect()
```

Parameters

None.

Matlab Example

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all neurons
doc.SelectAllNeurons();
% deselect variable with the name 'Neuron01'
doc.Variable('Neuron01').Deselect();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;
```

CHAPTER

FOUR

NEUROEXPLORER PYTHON PACKAGES

4.1 nex Python Package

nex Python Package provides a simple way to run Python scripts that control NeuroExplorer (open data file, get file data, run analysis, save results, etc.) in any Python IDE.

For NeuroExplorer scripting documentation, see *Scripting Reference*.

To use an external Python environment to run Python scripts in NeuroExplorer:

- In NeuroExplorer, select **Script | Enable Running Python Scripts in External Editor** menu command
- Install `nex` Python package. Run this command in Windows Command Prompt:

```
python.exe -m pip install -U nex
```

- If you get an error message `pip: command not found`, run this command first to install pip:

```
python.exe -m ensurepip --upgrade
```

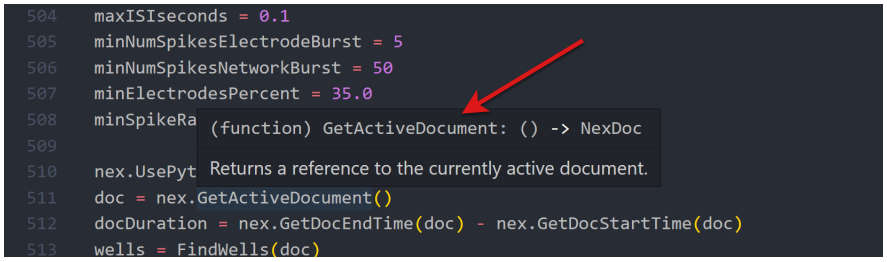
- If you are using Anaconda:
  - Type `Anaconda Prompt` in Windows Search box
  - Select a version of Anaconda Prompt in Best Match panel above Windows Search box
  - Run this command in Anaconda Prompt

```
pip install -U nex
```

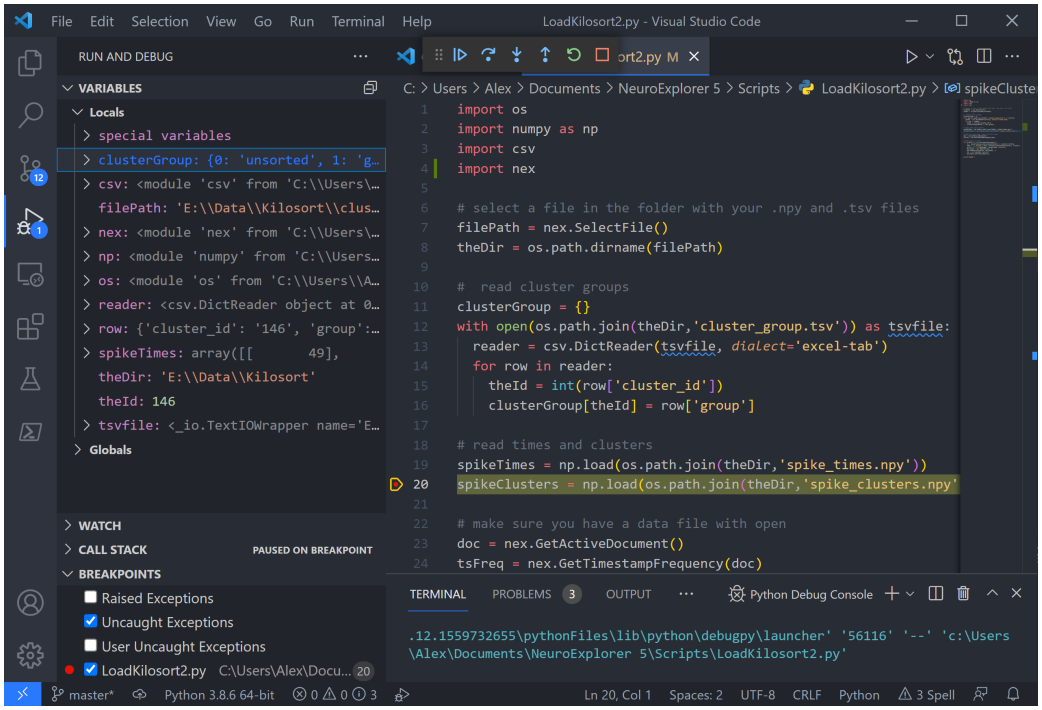
No changes in your script are required.

If you are using Visual Studio Code:

- Open Python script in Visual Studio Code or, in NexScript, select menu command **File | Open in VS Code**
- In Visual Studio Code, you will see full IntelliSense support (code completion, parameter info, etc.) for nex package:



- Press F5 to debug your script in Visual Studio Code:



4.2 nex5file Python Package

nex5file Python package allows you to read, write and edit data stored in NeuroExplorer .nex and .nex5 files.

4.2.1 Getting Started

Install nex5file package

Run this command in Windows Command Prompt:

```
python.exe -m pip install -U nex5file
```

If you get an error message `pip: command not found`, run this command first to install pip:

```
python.exe -m ensurepip --upgrade
```

Read .nex and .nex5 Files

`ReadNexFile` method of `nex5file.reader.Reader` class reads and parses contents of a .nex or a .nex5 file and returns an instance of `nex5file.filedata.FileData` object. This method reads all the data in the file.

```
from nex5file.reader import Reader
from nex5file.filedata import FileData
nexFilePath = r"C:\path\to\your\file.nex"
reader = Reader()
data = reader.ReadNexFile(nexFilePath)
```

If you need to read only some channels from a .nex or .nex5 file, use `ReadNexFileVariables` method:

```
# read only two continuous channels: cont1 and cont2
from nex5file.reader import Reader
from nex5file.filedata import FileData
nexFilePath = r"C:\path\to\your\file.nex"
reader = Reader()
data = reader.ReadNexFileVariables(nexFilePath, ['cont1', 'cont2'])
```

To retrieve channel names from a file, use `ReadNexHeadersOnly` method. Here is the code to read only continuous channels:

```
# read only continuous channels
from nex5file.reader import Reader
from nex5file.filedata import FileData
reader = Reader()
data = reader.ReadNexHeadersOnly(r"C:\path\to\your\file.nex")
contNames = data.ContinuousNames()
data_cont = reader.ReadNexFileVariables(r"C:\path\to\your\file.nex",␣
↪contNames)
```

Access Data in a FileData Object

Retrieving information from `FileData` object is similar to retrieving values using `nex` package. The difference is that `nex` package requires NeuroExplorer to be installed and running, while `nex5file` package is pure Python.

The syntax for accessing data is similar to `nex` package syntax. Many method names in `nex5file` package are the same as in `nex` package.

Here is a script to get continuous channels information using `nex`:

```
import nex
nexFilePath = r"C:\path\to\your\file.nex"
doc = nex.OpenDocument(nexFilePath)
# print continuous channel name, sampling rates and continuous values
for name in doc.ContinuousNames():
   rate = doc[name].SamplingRate()
   values = doc[name].ContinuousValues()
   print(name, rate, values)
```

Here is the same functionality using `nex5file` package:

```
from nex5file.reader import Reader
from nex5file.filedata import FileData

reader = Reader()
nexFilePath = r"C:\path\to\your\file.nex"
data = reader.ReadNexFile(nexFilePath)
# print continuous channel names, sampling rates and continuous values
for name in data.ContinuousNames():
   rate = data[name].SamplingRate()
   values = doc[name].ContinuousValues()
   print(name, rate, values)
```

Modify Data in a FileData Object

You can use the following *FileData* methods to modify data:

- `DeleteVariable`
- `AddEvent`
- `AddNeuron`
- `AddIntervalAsPairsStartEnd`
- `AddMarker`
- `AddContVarWithFloatsSingleFragment`
- `AddContSingleFragmentValuesInt16`
- `AddContVarWithFloatsAllTimestamps`
- `AddWaveVarWithFloats`

Write .nex and .nex5 Files

Use `WriteNexFile` method of `nex5file.writer.Writer` class

```
from nex5file.writer import Writer
from nex5file.filedata import FileData
import numpy as np

freq = 100000
data = FileData(freq)

eName = "event 001"
eventTs = [1, 2, 3.5]
data.AddEvent(eName, np.array(eventTs))

nName = "neuron 002"
neuronTs = [0.001, 2.54, 8.99]
data.AddNeuron(nName, np.array(neuronTs))

nexFilePath = r"C:\path\to\your\file.nex"
writer = Writer()
writer.WriteNexFile(data, nexFilePath)
```

nex5file

nex5file package

Submodules

nex5file.filedata module

class nex5file.filedata.**FileData**(*tsFrequency: float = 10000*, *comment: str = ''*)

Bases: object

FileData: Class for Managing Data in .nex and .nex5 Data Files.

Parameters

timestamp_frequencyHz (float) – The timestamp frequency in Hertz. Defaults to 100,000 Hertz.

Raises

ValueError – If timestamp_frequencyHz is less than or equal to 0.

Example:

```
from nex5file.filedata import FileData

# Create a FileData instance with a custom timestamp frequency
file_data = FileData(timestamp_frequencyHz=50000, comment="Sample Data")

# Add an event variable to the data
file_data.AddEvent("EventVariable", [1.0, 2.0, 3.0])

# Retrieve a variable by name
event_var = file_data["EventVariable"]

# Get the timestamp frequency
timestamp_frequency = file_data.GetTimestampFrequency()
```

AddContSingleFragmentValuesInt16(*contName: str*, *samplingRate: float*, *startTimestamp: float*, *contValuesAsInt16: list*, *rawToMV: float*, *rawOffset: float*) → None

Add a continuous variable with a single fragment by specifying int16 values and scaling to the FileData instance.

Parameters

- **contName** (str) – The name of the continuous variable.

- **samplingRate** (float) – The sampling rate of the continuous variable in Hertz.
- **startTimestamp** (float) – The timestamp at the start of the first fragment in seconds.
- **contValuesAsInt16** (numpy array of type np.int16) – The continuous values as int16 values.
- **rawToMV** (float) – Conversion factor from AD units to millivolts (millivolts).
- **rawOffset** (float) – Offset in millivolts.

Example:

```
file_data.AddContSingleFragmentValuesInt16("ContVariable", 1000.0, 0.
↪0, [100, 200, 300], 0.1, 0.0)
```

AddContVarWithFloatsAllTimestamps(*contName: str*, *samplingRate: float*, *allTimestamps: list*, *contValues: list*) → None

Add a continuous variable with float values and all the timestamps to the data.

Parameters

- **contName** (str) – The name of the continuous variable.
- **samplingRate** (float) – The sampling rate of the continuous variable in Hertz.
- **allTimestamps** (numpy array of type np.float64) – The timestamps for all data points in seconds.
- **contValues** (numpy array of type np.float32) – The continuous values in millivolts.

Example:

```
timestamps = [0.0, 0.001, 0.002]
values = [1.0, 2.0, 3.0]
file_data.AddContVarWithFloatsAllTimestamps("ContVariable", 1000.0,␣
↪timestamps, values)
```

AddContVarWithFloatsSingleFragment(*contName: str*, *samplingRate: float*, *startTimestamp: float*, *contValues: list*) → None

Add a continuous variable with float values and a single fragment to the FileData instance.

Parameters

- **contName** (str) – The name of the continuous variable.
- **samplingRate** (float) – The sampling rate of the continuous variable in Hertz.
- **startTimestamp** (float) – The timestamp at the start of the first fragment in seconds.
- **contValues** (List[float]) – The continuous values in millivolts.

Example:

```
file_data.AddContVarWithFloatsSingleFragment("ContVariable", 1000.0,␣
↪0.0, [1.0, 2.0, 3.0])
```

AddEvent(*evName: str*, *evTimestamps: list*) → None

Add an event variable to the FileData instance.

Parameters

- **evName** (str) – The name of the event variable.
- **evTimestamps** (List[float]) – Event timestamps in seconds.

Example:

```
file_data.AddEvent("EventVariable", [1.0, 2.0, 3.0])
```

AddIntervalAsPairsStartEnd(*intName: str*, *intervalsAsPairs*) → None

Add an interval variable to the FileData instance using start and end pairs.

Parameters

- **intName** (str) – The name of the interval variable.
- **intervalsAsPairs** (list of tuples) – List of interval start and end pairs in seconds.

Example:

```
intervals = [(0.0, 1.0), (1.5, 2.0)]
file_data.AddIntervalAsPairsStartEnd("IntervalVariable", intervals)
```

AddMarker(*markerName*, *timestamps: list*, *fieldNames: list*, *fields*)

Add a marker variable to the FileData instance.

Parameters

- **markerName** (str) – The name of the marker variable.
- **timestamps** (List[float]) – The timestamps in seconds.
- **fieldNames** (List[str]) – The names of marker fields.

- **fields** (list) – List of marker fields. Each element of the list contains values for a field.

Raises

ValueError – If the number of field names does not match the number of fields or if the length of any field does not match the length of timestamps.

Example:

```
field_names = ["Field1", "Field2"]
fields = [[1.0, 2.0], [3.0, 'abc']]
file_data.AddMarker("MarkerVariable", [0.0, 1.0], field_names,␣
↪fields)
```

AddNeuron(*nrName: str*, *nrTimestamps: list*, *wire: int = 0*, *unit: int = 0*, *xPosition: float = 0*, *yPosition: float = 0*) → None

Add a neuron variable to the FileData instance.

Parameters

- **nrName** (str) – The name of the neuron variable.
- **nrTimestamps** (List[float]) – Neuron timestamps in seconds.
- **wire** (int) – The wire number. Defaults to 0.
- **unit** (int) – The unit number. Defaults to 0.
- **xPosition** (float) – The x-position in range [0,100]. Defaults to 0.
- **yPosition** (float) – The y-position in range [0,100]. Defaults to 0.

Example:

```
file_data.AddNeuron("NeuronVariable", [1.0, 2.0, 3.0])
```

AddWaveVarWithFloats(*waveName: str*, *samplingRate: float*, *timestamps: list*, *waveValues: list*) → None

Add a waveform variable with float values to the data.

Parameters

- **waveName** (str) – The name of the waveform variable.
- **samplingRate** (float) – The sampling rate of the waveform variable in Hertz.
- **timestamps** (numpy array of type np.float64) – The timestamps in seconds.

- **waveValues** (numpy array of type np.float32) – The waveform values as a numpy array. Each column represents a waveform.

Example:

```
wave_name = "WaveformVariable"
sampling_rate = 1000.0  # Hertz
timestamps = [0.0, 1.0, 2.0]
wave_values = [[2, 3, 4, 1], [5, 6, 7, 2]

file_data.AddWaveVarWithFloats(wave_name, sampling_rate, timestamps,␣
↪wave_values)
```

ContinuousNames() → list

Get the list of continuous variable names in the FileData instance.

Returns
A list of continuous variable names.

Return type
List[str]

DeleteVariable(*name: str*) → None

Delete a variable from the FileData instance by its name.

Parameters
name (str) – The name of the Variable to be deleted.

Raises
ValueError – If no Variable with the specified name exists in the FileData instance.

EventNames() → list

Get the list of event variable names in the FileData instance.

Returns
A list of event variable names.

Return type
List[str]

GetDocComment() → str

Get the comment of the FileData instance.

Returns
The comment of the FileData instance.

Return type
str

GetDocEndTime() → float

Get the end time of the data in the FileData instance in seconds.

Returns
The end time in seconds.

Return type
float

GetDocStartTime() → float

Get the start time of the data in the FileData instance in seconds.

Returns
The start time in seconds.

Return type
float

GetTimestampFrequency() → float

Get the timestamp frequency of the FileData instance in Hertz.

Returns
The timestamp frequency in Hertz.

Return type
float

IntervalNames() → list

Get the list of interval variable names in the FileData instance.

Returns
A list of interval variable names.

Return type
List[str]

MarkerNames() → list

Get the list of marker variable names in the FileData instance.

Returns
A list of marker variable names.

Return type
List[str]

NeuronNames() → list

Get the list of neuron variable names in the FileData instance.

Returns
A list of neuron variable names.

Return type
List[str]

PopVectorNames() → list

Get the list of population vector variable names in the FileData instance.

Returns
A list of population vector variable names.

Return type
List[str]

WaveNames() → list

Get the list of waveform variable names in the FileData instance.

Returns
A list of waveform variable names.

Return type
List[str]

nex5file.variables module

class nex5file.variables.**ContinuousVariable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: Variable

Represents a continuous variable. The variable contains continuous values and their timestamps.

ContinuousValues() → list

Get a copy of the continuous values in millivolts.

Returns
A copy of the continuous values as a numpy array.

Return type
numpy array of type np.float64

FragmentCounts() → list

Get a copy of the fragment counts.

Returns
A copy of the fragment counts as a numpy array.

Return type
numpy array of type np.int64

FragmentTimestamps() → list

Get a copy of the fragment timestamps in seconds.

Returns
A copy of the fragment timestamps as a numpy array.

Return type
numpy array of type np.float64

SamplingRate() → float

Get the sampling rate of the continuous data.

Returns
The sampling rate.

Return type
float

class nex5file.variables.**EventVariable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: Variable

Represents an event variable (event name and a list of timestamps).

Timestamps() → list

Get a copy of the timestamps in seconds.

Returns
A copy of the timestamps (in seconds) as a numpy array.

Return type
numpy array of type np.float64

class nex5file.variables.**IntervalVariable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: Variable

Represents an interval variable (a variable name and interval start and end times).

Intervals() → list

Get a list of intervals represented as two lists – the first list is a list of interval starts, the second list is a list of interval ends.

Returns
a list of intervals represented as two lists – the first list is a list of interval starts, the second list is a list of interval ends.

Return type
list

class nex5file.variables.**MarkerVariable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: Variable

Represents a marker variable with. Each marker is a timestamp with one or more associated string values.

MarkerFieldNames() → list

Get a copy of the marker field names.

Returns
A copy of the marker field names.

Return type
List[str]

Markers() → list

Get a copy of the marker fields.

Returns
A copy of the marker fields.

Return type
List[List[str]]

Timestamps() → list

Get a copy of the timestamps associated with the markers.

Returns
A copy of the timestamps as a numpy array.

Return type
numpy array of type np.float64

class nex5file.variables.**NeuronVariable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: EventVariable

Represents a neuron variable (a variable name, timestamps and wire and unit information).

Timestamps() → list

Get a copy of the timestamps in seconds.

Returns
A copy of the timestamps (in seconds) as a numpy array.

Return type
numpy array of type np.float64

class nex5file.variables.**NexFileVarType**

Bases: object

Constants for .nex and .nex5 variable types.

class nex5file.variables.**PopulationVector**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: Variable

Represents a population vector variable (a variable name and a list of weights).

Weights() → list

Get a copy of the population vector weights.

Returns
A copy of the weights as a numpy array.

Return type
List[float]

class nex5file.variables.**Variable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: object

Represents a variable stored in .nex of .nex5 file.

Metadata() → dict

Get the metadata associated with the variable.

Returns
A dictionary containing metadata about the variable (name, wire number for neurons etc.).

Return type
dict

class nex5file.variables.**WaveformVariable**(*varHeader: VariableHeader = VariableHeader(Type=-1, Version=0, Name='', DataOffset=0, Count=0, TsDataType=0, ContDataType=0, SamplingRate=0, Units='', ADtoMV=0, MVOffset=0, NPointsWave=0, PreThrTime=0, MarkerDataType=0, NMarkers=0, MarkerLength=0, ContFragIndexType=0, Padding='', Gain=0, Filter=0, Wire=0, Unit=0, XPos=0, YPos=0)*)

Bases: Variable

Represents a waveform variable (each waveform is a timestamps and array of waveform values).

NumPointsInWave() → int

Get the number of points in the waveform.

Returns
The number of points in the waveform.

Return type
int

PreThresholdTime() → float

Get the pre-threshold time.

Returns
The pre-threshold time.

Return type
float

SamplingRate() → float

Get the sampling rate of the waveform.

Returns
The sampling rate.

Return type
float

Timestamps() → list

Get a copy of the timestamps.

Returns
A copy of the timestamps as a numpy array.

Return type
numpy array of type np.float64

WaveformValues() → list

Get a copy of the waveform values.

Returns
A copy of the waveform values as a numpy array.

Return type
numpy array of type np.float32

nex5file.reader module

class nex5file.reader.**Reader**

Bases: object

This class provides functionality for reading both .nex and .nex5 files and extracting their data.

ReadNex5File(*filePath: str*) → *FileData*

Read a .nex5 file and return its data.

Parameters
filePath (str) – Path to the .nex5 file.

Returns
A FileData object containing the data from the file.

Return type
FileData

Raises
ValueError – If the file format is invalid.

Example:

```
reader = Reader()
data = reader.ReadNex5File(r"C:\path\to\your\file.nex5")
```

ReadNex5FileVariables(*filePath: str*, *varNames: list*) → *FileData*

Read specified variables from .nex5 file and return a FileData object.

Parameters

- **filePath** (str) – Path to the .nex5 file.
- **list** (varNames) – a list of variable names.

Returns
A FileData object containing variable data from the file.

Return type
FileData

Raises
ValueError – If the file format is invalid.

Example

Read only continuous channels from .nex file

```
reader = Reader()
data = reader.ReadNex5HeadersOnly(r"C:\path\to\your\file.nex5")
contNames = data.ContinuousNames()
data_cont = reader.ReadNex5FileVariables(r"C:\path\to\your\file.nex5
↪", contNames)
```

ReadNex5HeadersOnly(*filePath: str*) → *FileData*

Read .nex5 file headers and return a FileData object with no data. The returned FileData object can be used to obtain the names of variables in the file. The names can then be used to load only specific variables from a .nex5 file using ReadNex5FileVariables method.

Parameters
filePath (str) – Path to the .nex5 file.

Returns
A FileData object containing variable headers from the file.

Return type
FileData

Raises
ValueError – If the file format is invalid.

Example:

```
# read only continuous channels from .nex file
reader = Reader()
data = reader.ReadNex5HeadersOnly(r"C:\path\to\your\file.nex5")
contNames = data.ContinuousNames()
data_cont = reader.ReadNex5FileVariables(r"C:\path\to\your\file.nex5
↪", contNames)
```

ReadNexFile(*filePath: str*) → *FileData*

Read a .nex file and return its data.

Parameters
filePath (str) – Path to the .nex file.

Returns
A FileData object containing the data from the file.

Return type
FileData

Raises
ValueError – If the file format is invalid.

Example:

```
reader = Reader()
data = reader.ReadNexFile(r"C:\path\to\your\file.nex")
```

ReadNexFileVariables(*filePath: str*, *varNames: list*) → *FileData*

Read specified variables from .nex file and return a FileData object.

Parameters

- **filePath** (str) – Path to the .nex file.
- **list** (varNames) – a list of variable names.

Returns
A FileData object containing variable data from the file.

Return type
FileData

Raises
ValueError – If the file format is invalid.

Example

Read only continuous channels from .nex file

```
reader = Reader()
data = reader.ReadNexHeadersOnly(r"C:\path\to\your\file.nex")
contNames = data.ContinuousNames()
data_cont = reader.ReadNexFileVariables(r"C:\path\to\your\file.nex",␣
↪contNames)
```

ReadNexHeadersOnly(*filePath: str*) → *FileData*

Read .nex file headers and return a FileData object with no data. The returned FileData object can be used to obtain the names of variables in the file. The

names can then be used to load only specific variables from a .nex file using ReadNexFileVariables method.

Parameters
filePath (str) – Path to the .nex file.

Returns
A FileData object containing variable headers from the file.

Return type
FileData

Raises
ValueError – If the file format is invalid.

Example

Read only continuous channels from .nex file

```
reader = Reader()
data = reader.ReadNexHeadersOnly(r"C:\path\to\your\file.nex")
contNames = data.ContinuousNames()
data_cont = reader.ReadNexFileVariables(r"C:\path\to\your\file.nex",␣
↪contNames)
```

nex5file.writer module

class nex5file.writer.**Writer**

Bases: object

.nex and .nex5 writer class

This class provides methods for writing data to .nex and .nex5 files.

WriteNex5File(*data:* FileData, *filePath: str*) → None

Write data to a .nex5 file.

This method writes data from a FileData object to a .nex5 file located at the specified 'filePath'.

If filePath ends with '.nex', writes data using nex data format.

Parameters

- **data** (FileData) – A FileData object containing the data to be written.
- **filePath** (str) – The path to the .nex5 file to be created or updated.

Example

To write data to a .nex5 file:

```
writer = Writer()
writer.WriteNex5File(file_data, r"C:path\to\your\file.nex5")
```

WriteNexFile(*data:* FileData, *filePath: str*) → None

Write data to a .nex file.

This method writes data from a FileData object to a .nex file located at the specified 'filePath'.

If filePath ends with '.nex5', writes data using nex5 data format.

Parameters

- **data** (FileData) – A FileData object containing the data to be written.
- **filePath** (str) – The path to the .nex file to be created or updated.

Raises

ValueError – If the maximum timestamp exceeds the 32-bit range, preventing it from being saved as a .nex file.

Example

To write data to a .nex file:

```
writer = Writer()
writer.WriteNexFile(file_data, r"C:\path\to\your\file.nex")
```

Module contents

PYTHON MODULE INDEX

n