

# **NeuroExplorer<sup>®</sup>**

## **User Manual and Reference**



# Table of Contents

1.	Getting Started with NeuroExplorer®	5
1.1.	Getting Started with NeuroExplorer	6
1.2.	NeuroExplorer Screen Elements	7
1.3.	NeuroExplorer File Viewer	10
1.4.	Opening Files and Importing Data	11
1.5.	Importing Files Created By Data Acquisition Systems	12
1.6.	Importing Data from Text Files	13
1.7.	Importing Data from Spreadsheets	15
1.8.	Importing Data from Matlab	16
1.9.	Reading and Writing NeuroExplorer Data Files	17
1.10.	1D Data Viewer	18
1.11.	Analyzing Data	19
1.12.	Selecting Variables for Analysis	20
1.13.	Adjusting Analysis Properties	21
1.14.	Analysis Templates	22
1.15.	Numerical Results	23
1.16.	Post-processing	24
1.17.	Saving Results as Power Point Slides	25
1.18.	Working with Matlab	26
1.19.	Working with Excel	27
1.20.	Saving Graphics	27
2.	Working with Graphics	28
2.1.	NeuroExplorer Graphics	28
2.2.	Graphics Modes	29
2.3.	Positioning Graphics Objects	30
2.4.	Text Labels	32
2.5.	Lines	34
2.6.	Rectangles	34
3.	NeuroExplorer Analysis Reference	35
3.1.	Introduction	35
3.2.	Data Types	36
3.2.1.	Spike Trains	36
3.2.2.	Events	37
3.2.3.	Intervals	38
3.2.4.	Markers	40
3.2.5.	Population Vectors	41
3.2.6.	Waveforms	42
3.2.7.	Continuously Recorded Data	43
3.3.	Data Selection Options	45
3.4.	Post-Processing Options	46
3.5.	Matlab Options	47
3.6.	Excel Options	47
3.7.	Confidence Limits for Perievent Histograms	48
3.8.	Cumulative Sum Graphs	49
3.9.	Rate Histograms	50
3.10.	Interspike Interval Histograms	51
3.11.	Autocorrelograms	52
3.12.	Perievent Histograms	53
3.13.	Crosscorrelograms	55
3.14.	Shift-Predictor for Crosscorrelograms	57

3.15.	Rasters.....	58
3.16.	Perievent Rasters .....	58
3.17.	Joint PSTH.....	60
3.18.	Cumulative Activity Graphs.....	61
3.19.	Instant Frequency .....	61
3.20.	Interspike Intervals vs. Time .....	62
3.21.	Poincare Maps.....	62
3.22.	Spike Distance vs. Time .....	62
3.23.	Trial Bin Counts .....	63
3.24.	Power Spectral Densities.....	64
3.25.	Burst Analysis .....	67
3.26.	Principal Component Analysis.....	69
3.27.	PSTH versus Time.....	70
3.28.	Correlations with Continuous Variable .....	71
3.29.	Regularity Analysis .....	72
3.30.	Place Cell Analysis .....	73
3.31.	Reverse Correlation.....	74
3.32.	Epoch Counts .....	76
3.33.	Coherence Analysis.....	77
3.34.	Spectrogram Analysis.....	77
4.	Programming with NexScript .....	79
4.1.	Introduction to NexScript Programming .....	79
4.2.	Variables.....	81
4.3.	Expressions .....	82
4.4.	Flow Control .....	83
4.5.	Functions .....	85
4.5.1.	Function categories.....	85
4.5.2.	Math Functions.....	85
4.5.3.	String Functions .....	86
4.5.4.	File Access Functions .....	87
4.5.4.1.	File Access Functions .....	87
4.5.4.2.	Opening and Saving Data Files .....	87
4.5.4.3.	Opening Multiple Data Files.....	88
4.5.4.4.	Saving Numerical Results.....	88
4.5.4.5.	Accessing Document Parameters .....	89
4.5.4.6.	Reading and Writing Text Files in NexScript .....	89
4.5.5.	Access to the File Variables.....	90
4.5.6.	Selecting Variables .....	92
4.5.7.	Modifying Existing Variables .....	93
4.5.8.	Creating and Deleting Variables .....	94
4.5.9.	Analysis Functions .....	95
4.5.10.	Matlab Functions .....	98
4.5.11.	Excel Functions .....	98
4.5.12.	User Interface Functions .....	98
4.5.13.	Debugging Functions.....	99
5.	Working with 3D Graphics .....	100
5.1.	Introduction.....	100
5.2.	Viewing Multiple Histograms in 3D.....	101
5.3.	3D Graphics Parameters.....	102
5.4.	Viewing the Neuronal Activity "Movie".....	103
5.5.	Activity Animation Parameters .....	105
Index	.....	106

# 1. Getting Started with NeuroExplorer®

## Installation

Before you can use NeuroExplorer, you must install NeuroExplorer program files, Sentinel system drivers and install the Sentinel hardware key.

## Running NeuroExplorer Setup

Before you begin installing NeuroExplorer and its components, exit all currently running applications.

- Place NeuroExplorer setup CD into your computer's CD-ROM drive.
- Navigate to **Nex3Setup.exe** file on the CD and double-click on the file. NeuroExplorer Version 3 setup screen appears.
- Follow the prompts on the setup dialogs and complete the installation.
- At the end of the installation process, Sentinel System Driver Setup will start automatically.
- Follow Sentinel Driver Setup prompts and complete the installation of Sentinel Drivers. You may need to restart your computer to complete the installation of Sentinel drivers.

Setup will create the following directory structure:

- Main NeuroExplorer directory (usually, C:\Program Files\Nex Technologies\Nex) and several subdirectories:
  - Scripts directory which contains script files
  - Templates directory which contains analysis template files
  - Sentinel Drivers directory. If you need to reinstall Sentinel Drivers, you can run the *Setup.exe* program in this directory.

## Installing Hardware Key

Before you can use NeuroExplorer, you need to install provided Sentinel Hardware Key on your computer. To install the parallel port key:

1. Locate an available parallel port on your computer. If your computer has one parallel port, you may need to temporarily remove any existing parallel port devices (such as printer or Zip drive) in order to connect the key. These devices may be reconnected to the key's outside connector after you installed the key.
2. Attach the key the parallel port connector.
3. Tighten the screws to connect the key securely to the port.
4. If necessary, reconnect any other parallel port devices to the outside connector of the key. Sentinel recommends using shielded printer cable if you are connecting a printer to your computer through Sentinel key.

Multiple Sentinel parallel port keys can be attached to the same parallel port. SentinelSuperPro keys (provided with your copy of NeuroExplorer) can be cascaded with other Rainbow Technologies keys that support cascading.

To install the USB key:

1. Make sure to run NeuroExplorer setup first
2. Reboot the computer after installing NeuroExplorer
3. Attach the key to the available USB port (New Hardware Found wizard will be shown)
4. Accept the defaults in the New Hardware Found wizard.

## 1.1. Getting Started with NeuroExplorer

This chapter, *Getting Started with NeuroExplorer*, describes the basics of using NeuroExplorer. The chapter offers the following topics on how to use NeuroExplorer:

- [NeuroExplorer Screen Elements](#)
- [Opening Files and Importing Data](#)
- [Importing Data from Text Files](#)
- [Importing Data from the Spreadsheets](#)
- [Analyzing Data](#)
- [Selecting Variables for Analysis](#)
- [Adjusting Analysis Properties](#)
- [Analysis Templates](#)
- [Numerical Results](#)
- [Post-processing](#)
- [Working with Matlab](#)
- [Working with Excel](#)
- [Saving Graphics](#)

Additional information is available in the following chapters:

- [Working with Graphics](#)
- [NeuroExplorer Analysis Reference](#)
- [Programming with NexScript](#)

### NeuroExplorer Technical Support

NeuroExplorer users can get help via e-mail [support@neuroexplorer.com](mailto:support@neuroexplorer.com)

Please visit NeuroExplorer Web site <http://www.neuroexplorer.com> for program updates and to get the latest information about NeuroExplorer.

### NeuroExplorer Updates

You can download the latest version of NeuroExplorer from the NeuroExplorer Web site.

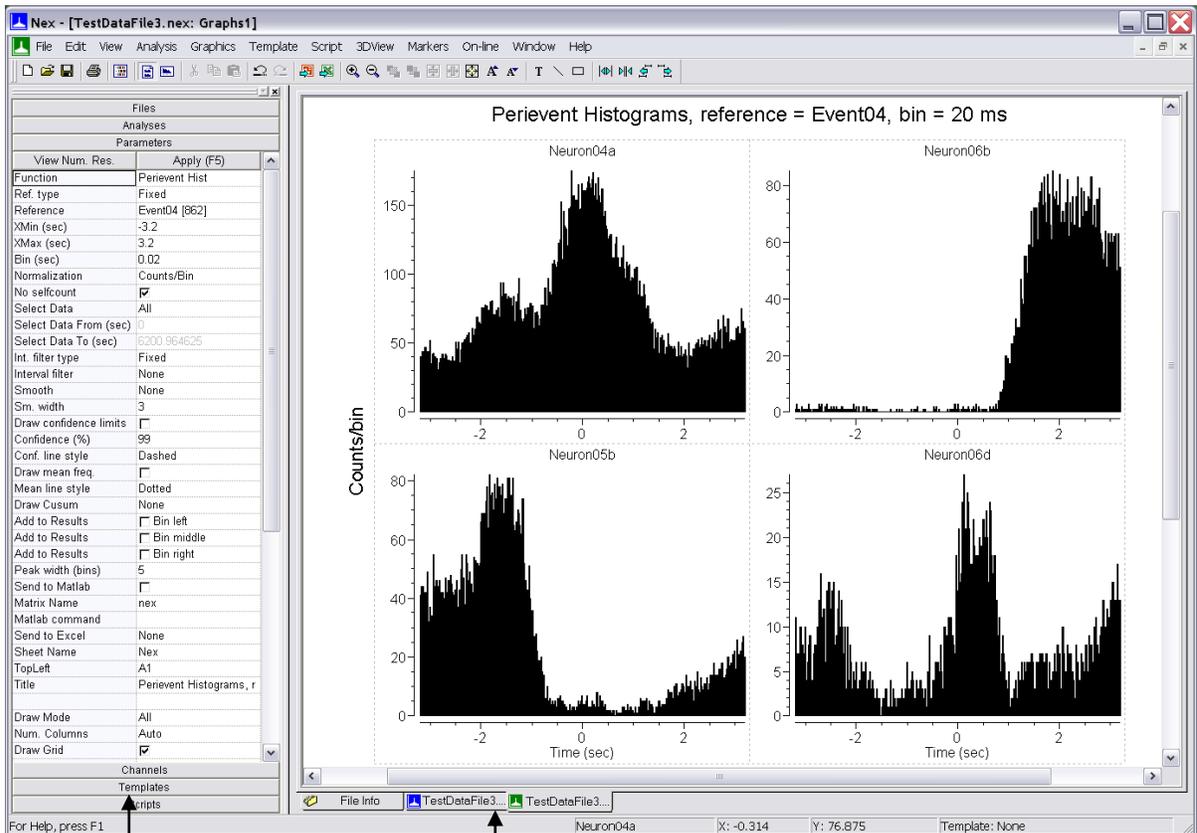
If you purchased NeuroExplorer Version 3.x and already installed NeuroExplorer from CD, download the file <http://www.neuroexplorer.com/updates/Nex3Update.exe> and run it from your local hard drive. This file will update NeuroExplorer executables and help files.

If you need a complete NeuroExplorer setup file, download the file <http://www.neuroexplorer.com/downloads/Nex3Setup.exe>. This file is a standard NeuroExplorer setup executable that is included in the NeuroExplorer CD. This setup file will execute the complete install – it will create folders, menu items in Start\Programs and create a desktop icon for NeuroExplorer.

## 1.2. NeuroExplorer Screen Elements

Typically, there are two windows visible in NeuroExplorer:

- one of the Views in the left panel
- a Graphics or Data window fills the rest of the NeuroExplorer main frame



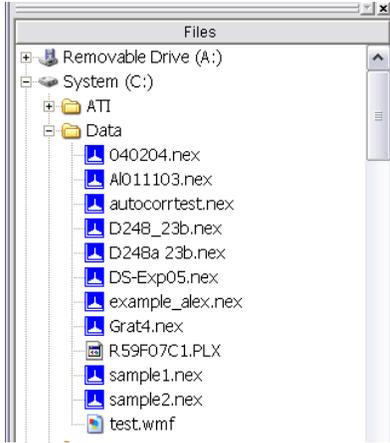
Tabs to switch between control panels

Tabs to switch between data and graphics windows

## Control Panel

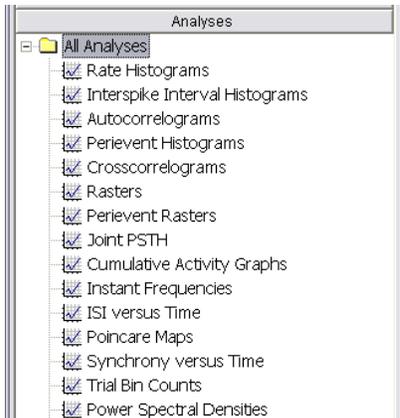
This window has 6 views: Files, Analyses, Parameters, Channels, Scripts and Templates.

### Files View



This view allows you to quickly browse through your data files. When you select (single-click) one of the data files, NeuroExplorer displays the file header information in the File Info View. To open the data file, simply double-click the file name.

### Analyses View



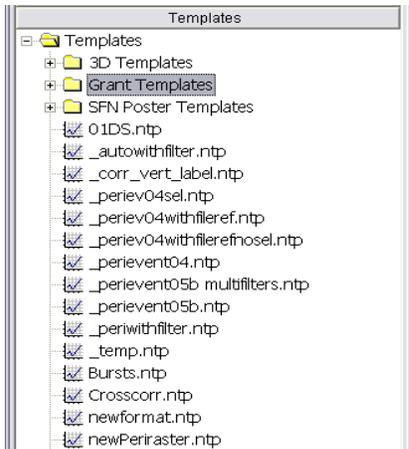
This view allows you to quickly select one of the analyses available in NeuroExplorer. To apply analysis to the active data file, double-click the analysis name.

### Parameters View

Parameters		
View Num.	Res.	Apply (F5)
Function		Perievent Hist
Ref. type	Fixed	
Reference	Event04 [862]	
XMin (sec)	-3.2	
XMax (sec)	3.2	
Bin (sec)	0.02	
Normalization	Counts/Bin	
No selfcount	<input checked="" type="checkbox"/>	
Select Data	All	
Select Data From (sec)		
Select Data To (sec)	6200.964625	
Int. filter type	Fixed	
Interval filter	None	
Smooth	None	
Sm. width	3	
Draw confidence limits	<input type="checkbox"/>	

The left column of the Parameters View lists the names of adjustable parameters for the selected object. The right column contains various controls that can be used to change the parameter values. To apply the changes, press the Apply button or hit the F5 key.

## Templates View

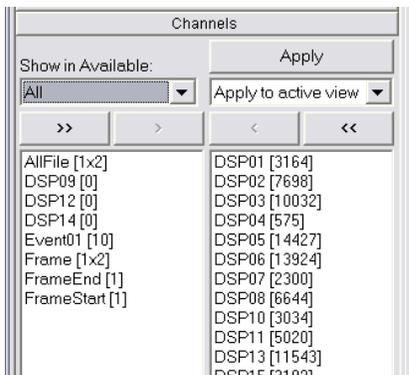


The Templates View can be used to quickly execute the Analysis Template. Just double-click the template name and the corresponding template will be immediately executed.

By default, the templates tree shown in the Templates View is a copy of the directory <NeuroExplorer Home>\Templates, where <NeuroExplorer Home> is your NeuroExplorer installation directory (usually, C:\Program Files\Nex Technologies\Nex). The template directory location can be specified using View | Options... menu command.

You can create subfolders in your template directory and then NeuroExplorer will allow you to navigate through the templates tree within the Templates View.

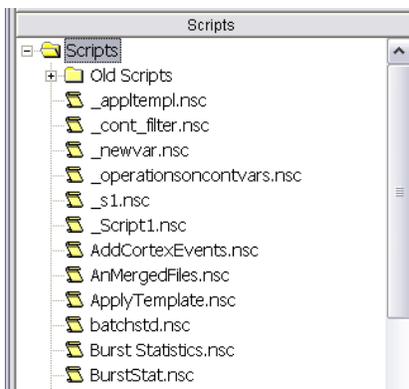
## Channels View



You can analyze all the variables in your data file, a subset of the variables, or may be just one variable. Channels View allows you to quickly select and deselect the variables used for analysis.

The left column shows the variables that are not currently selected; the right column shows the selected variables. To move variables from column to column, first select them, then press ">" (Select) or "<" (Deselect) buttons.

## Scripts View



This view can be used to select a script to be executed. Double-click the script name to run the selected script.

By default, the scripts tree shown in the Scripts View is a copy of the directory <NeuroExplorer Home>\Scripts, where <NeuroExplorer Home> is your NeuroExplorer installation directory (usually, C:\Program Files\Nex Technologies\Nex). The script directory location can be specified using View | Options... menu command.

You can create subfolders in your script directory and then NeuroExplorer will allow you to navigate through the scripts tree within the Scripts View.

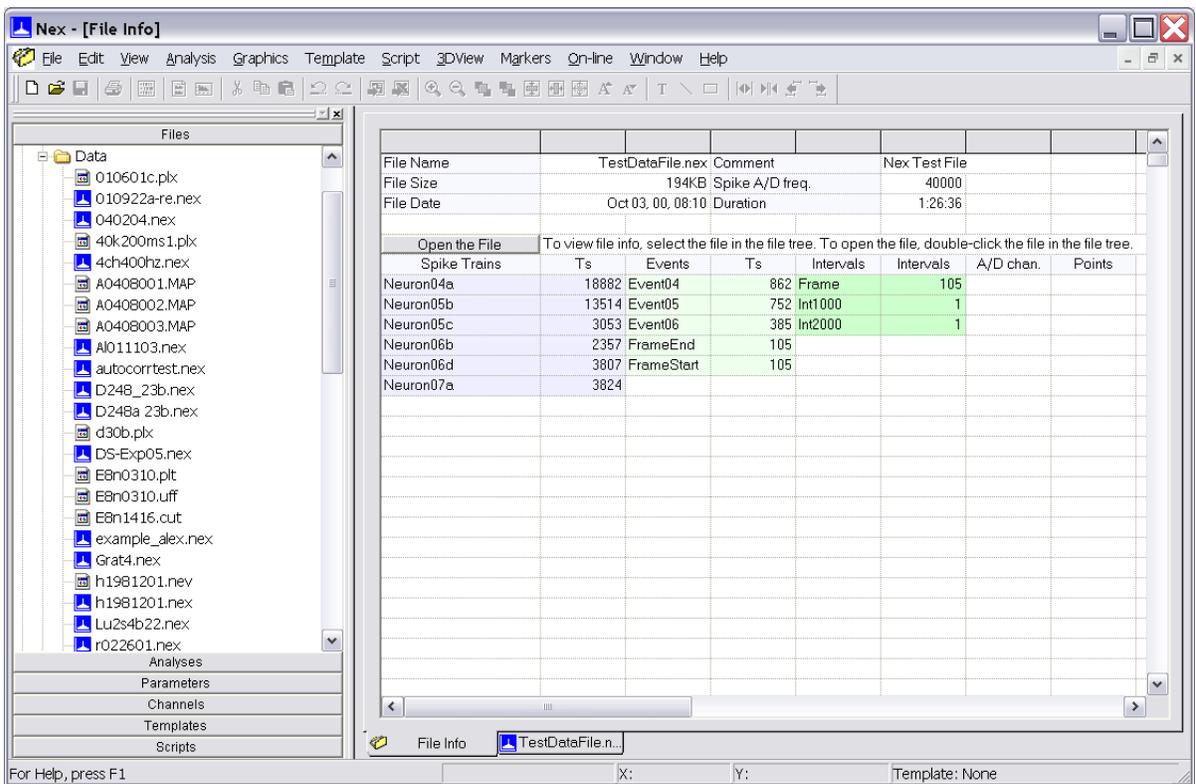
### 1.3. NeuroExplorer File Viewer

When you open NeuroExplorer, the first window that opens is the **File Info** window.

This window displays the information about the file that is selected in the **Files View** in the left panel. You can use the mouse as well as up and down arrow keys on the keyboard to change the selection in the Files View window.

To open selected file, press Enter or double-click the file name in the **Files View**.

To select the initial directory for the File tree, use **View | Data Import Options** to invoke *Data Import Options* dialog and specify your data directory in the *Default Data Directory* edit control.





## 1.5. Importing Files Created By Data Acquisition Systems

NeuroExplorer can read native data files created by popular data acquisition systems (Alpha Omega, CED Spike-2, Cortex, DataWave, Cyberkinetics, Instrutech, Multi Channel Systems, Neuralynx, Plexon and RC Electronics).

Since NeuroExplorer does not provide spike sorting capabilities yet, it is assumed that you have already sorted (clustered) the waveforms.

### Alpha Omega Files

NeuroExplorer can import Alpha Omega MAP, ISI, NDA, MAT and LSM files. All the spike trains, DIO events and continuous variables are imported. The waveforms can be imported as an option (see **File Import Options** below).

### Cyberkinetics Files

NeuroExplorer can import Cyberkinetics NEV files. All the spike trains and DIO events are imported. Analog channels can be imported as an option (see **File Import Options** below). The waveforms can be imported if the Neuroshare DLL option is selected in the File Import Options dialog.

### DataWave Files

NeuroExplorer can import both Discovery and Workbench files. In general, all the sorted spike trains, DIO events and trial descriptors are imported. The waveforms are not imported. Analog channels can be imported as an option (see **Options** below).

The following UFF types are imported from the Discovery files:

- S**, **E** and **U** UFF types are imported as [spike trains](#)

- B** UFF type records are imported as [events](#)

- T** UFF type records are imported as [markers](#)

- P** (position) UFF type records are imported as 4 continuous variables.

The following record types are imported from the Workbench files:

- Analysis records (with appropriate subtypes) are imported as [spike trains](#)

- Event records are imported as [events](#)

- Trial records are imported as [markers](#)

### CED Spike-2 Files

NeuroExplorer can import Spike-2 \*.smr files. All the sorted spike trains, events and markers are imported. The waveforms are not imported. Analog (Adc) channels can be imported as an option (see **File Import Options** below).

### Plexon Files

NeuroExplorer can read \*.plx and \*.ddt Plexon files. All the sorted spike trains and external events are imported. The waveforms and analog (slow) channels can be imported as an option (see **File Import Options** below).

### RC Electronics Files

NeuroExplorer can import both 12-bit RC Electronics files and 16-bit DATAMAX files. Since the current version of NeuroExplorer stores all the data in RAM, NeuroExplorer can import only relatively small (about the size of RAM on your PC) RC Electronics files.

### Multi Channel Systems Files

NeuroExplorer can import standard MCS data files. All the sorted spike trains and external events are imported. The waveforms are not imported. Analog channels can be imported as an option (see **File Import Options** below).

## Neuralynx Files

NeuroExplorer can import Neuralynx data files (tt\*.dat, se\*.dat, etc.) as well as the files created by **MClust** spike sorter (\*.t files). All the sorted spike trains are imported. External events are imported as [markers](#). Analog channels and waveforms can be imported as an option (see **File Import Options** below).

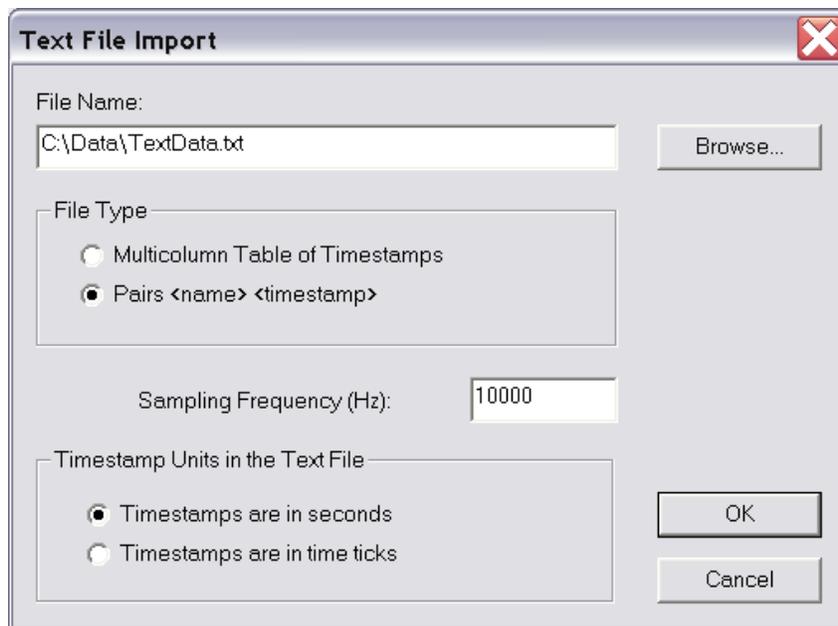
## File Import Options

The file import options can be set in the **Data Import** dialog. Use **View | Data Import Options** menu command to invoke the dialog.

## 1.6. Importing Data from Text Files

NeuroExplorer can import data from two major types of text files – text files with timestamps and text files with continuous data. To import data from text files with timestamps, use **File | Import Data | Text File** menu command. To import text files with continuous data, use **File | Import Data | Text File with Continuous Variables** menu command.

When you import data from **text files with timestamps**, specify text import options in the 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  $1./\text{Sampling\_Frequency}$ .

**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:

$$\text{Internal\_Timestamp} = \text{Timestamp\_In\_Seconds} * \text{Sampling\_Frequency}$$

NeuroExplorer can import data stored in the following text formats:

### 1. Multicolumn table of timestamps

In this format, each column in the text file contains the timestamps of a neuron. The first element in each column is a **neuron name**, that is, the first line of the file contains names of all the variables. Each **name** should be less than 64 characters long and should contain only letters, digits or the underscore sign. The first character of the name should be a letter. The **timestamps** are numbers representing the neuron firing time (or event time) in seconds or in time ticks. NeuroExplorer assumes that the columns are separated by tabs.

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

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

NeuroExplorer can also export data in this format (use **File | Save Data | As a Text File** menu command).

### 2. Pairs <name> <timestamp>

The text file in this format should contain the pairs of the type

```
<name>    <timestamp>
```

where

**<name>** is a character string that is less than 32 characters long and contains only letters, digits and the underscore sign. The first character of the name should be a letter. If the number is used for the name, NeuroExplorer will add "event" at the beginning of the 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 represented in seconds:

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

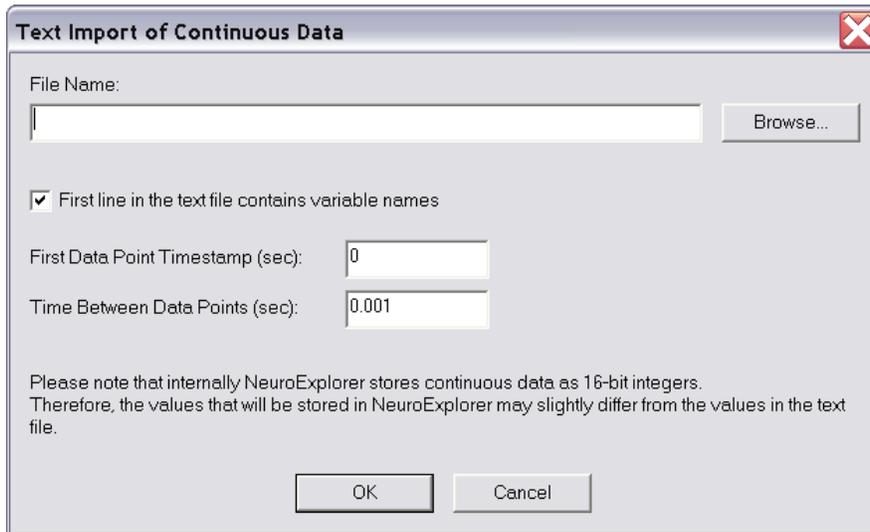
### 3. Multicolumn text files with continuous data

NeuroExplorer can also import **continuous data** from a multicolumn text file where each column corresponds to a continuous variable:

```
ContChannel1  ContChannel2
114.74609     -63.47656
56.15234      -358.88672
-187.98828    -63.47656
-48.82813     388.18359
-26.85547     285.64453
-7.32422      -180.66406
-102.53906    -283.20313
-78.125       -31.73828
```

The columns may be separated by any number of spaces, tabs or commas.

When you import **continuous data** from text files, the following dialog is 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 timestamp is calculated using the following formula:

$$\text{DataRowTimestamp} = \text{FirstDataPoint} + (N-1) * \text{TimeBetweenDataPoints}$$

## 1.7. Importing Data from Spreadsheets

Timestamped data can be pasted directly into the NeuroExplorer data table. To paste the following two timestamped variables (BarPress and Reward) from a spreadsheet to NeuroExplorer:

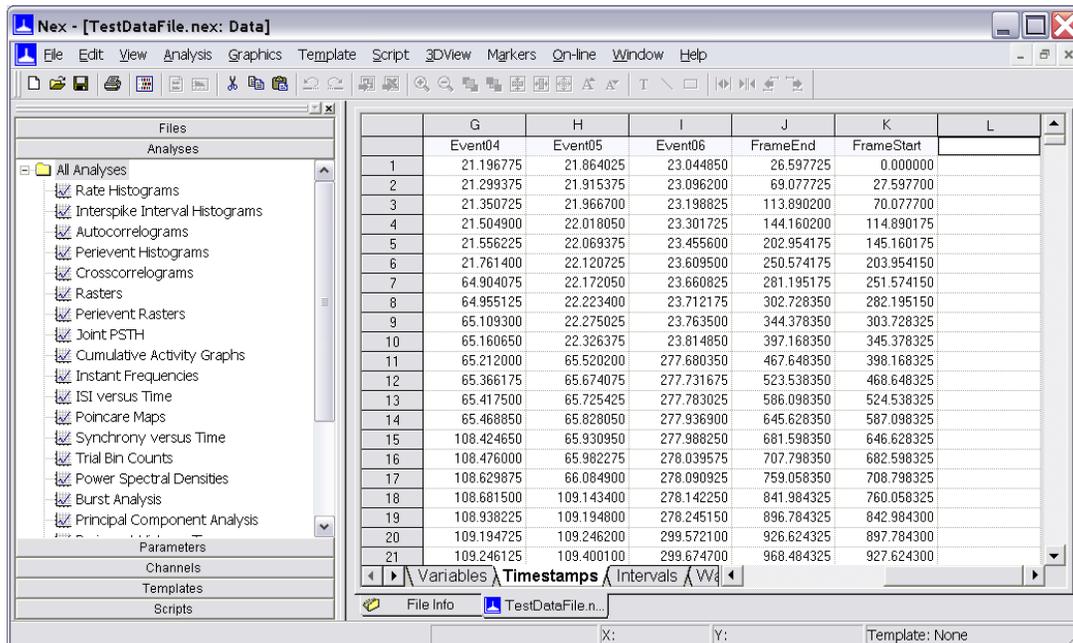
	A	B	C	D	E	F	G
1	BarPress	Reward					
2	20.3	23.08					
3	55.12	59.04					
4	122.4	127.01					
5							
6							
7							
8							
9							
10							

In Excel:

- Using the mouse, select the cell range with both variables (A1 to B5)
- Select **Edit | Copy** menu command

In NeuroExplorer:

- Select Data view of the file in NeuroExplorer
- Select the Timestamps tab of the Data view
- Scroll the timestamps window to the right and select the topmost cell of the empty column:



- Select **Edit | Paste** menu command.

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

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

## 1.8. Importing Data from Matlab

You can create spike trains (1xN or Nx1 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 **File | Get From Matlab... | Open Matlab As Engine** menu command. NeuroExplorer will open Matlab as engine.
- In opened Matlab command window, run your scripts and create spike train variables.
- To import timestamp variables, in NeuroExplorer, select **File | Get From Matlab... | Get Timestamp Variables...** menu command. NeuroExplorer will open a dialog with the list of available Matlab variables.
- In the dialog, select the variables you want to transfer to NeuroExplorer and press OK.
- To import continuous variables, in NeuroExplorer, select **File | Get From Matlab... | Get Continuous Variables...** menu command. NeuroExplorer will open a dialog with the list of

- available Matlab variables.
- In the dialog, select one of the variables you want to transfer to NeuroExplorer and press OK. NeuroExplorer will open a dialog where you will specify the variable options.

## 1.9. Reading and Writing NeuroExplorer Data Files

NeuroExplorer Data file has the following structure:

```
file header
variable header 1
variable header 2
...
data for variable i
data for variable j
...
```

Here is the minimal code that reads NeuroExplorer data file:

```
void ReadNex(){
    NexFileHeader fh;
    // assume that file has less than 1000 variables
    NexVarHeader  vh[1000];
    int i;
    FILE* fp;

    fp = fopen("text.nex", "rb");
    // 1. read file header
    fread(&fh, sizeof(NexFileHeader), 1, fp);

    // 2. read the variable headers
    for(i=0; i<fh.NumVars; i++){
        fread(&vh[i], sizeof(NexVarHeader), 1, fp);
    }

    // 3. read timestamp data for the first variable
    // assuming the first variable is neuron or event
    // and it contains less than 10000 timestamps
    int timestamps[10000];
    if(vh[0].Type == 0 || vh[0].Type == 1){
        // seek to the start of data
        fseek(fp,  vh[0].DataOffset, SEEK_SET);
        // read the timestamps, 4 bytes per timestamp
        fread(timestamps, vh[0].Count*4, 1, fp);
    }
    fclose(fp);
}
```

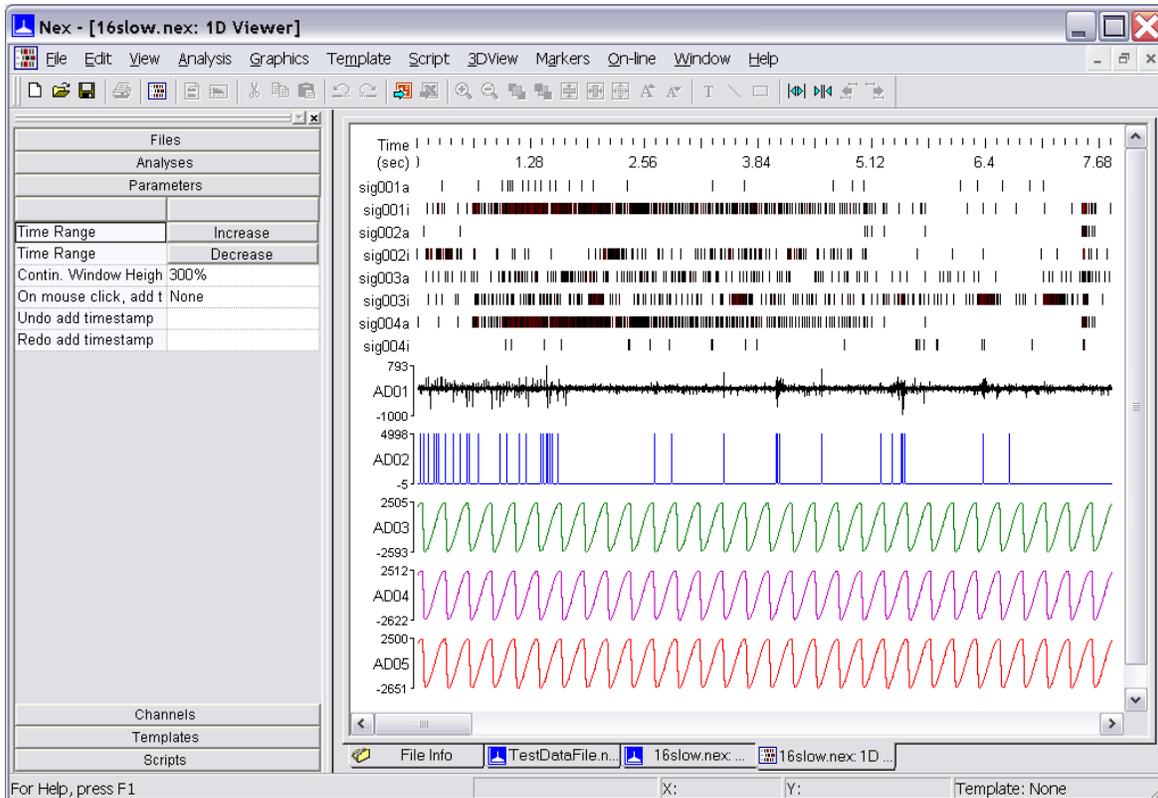
The complete code of the program that reads and writes all NeuroExplorer data types is available at NeuroExplorer web site:

<http://www.neuroexplorer.com/updates/HowToReadAndWriteNexFiles.zip>

## 1.10. 1D Data Viewer

NeuroExplorer provides a window that displays graphically all the selected variables. To open this window, use **View | 1D Data Viewer Window** menu command.

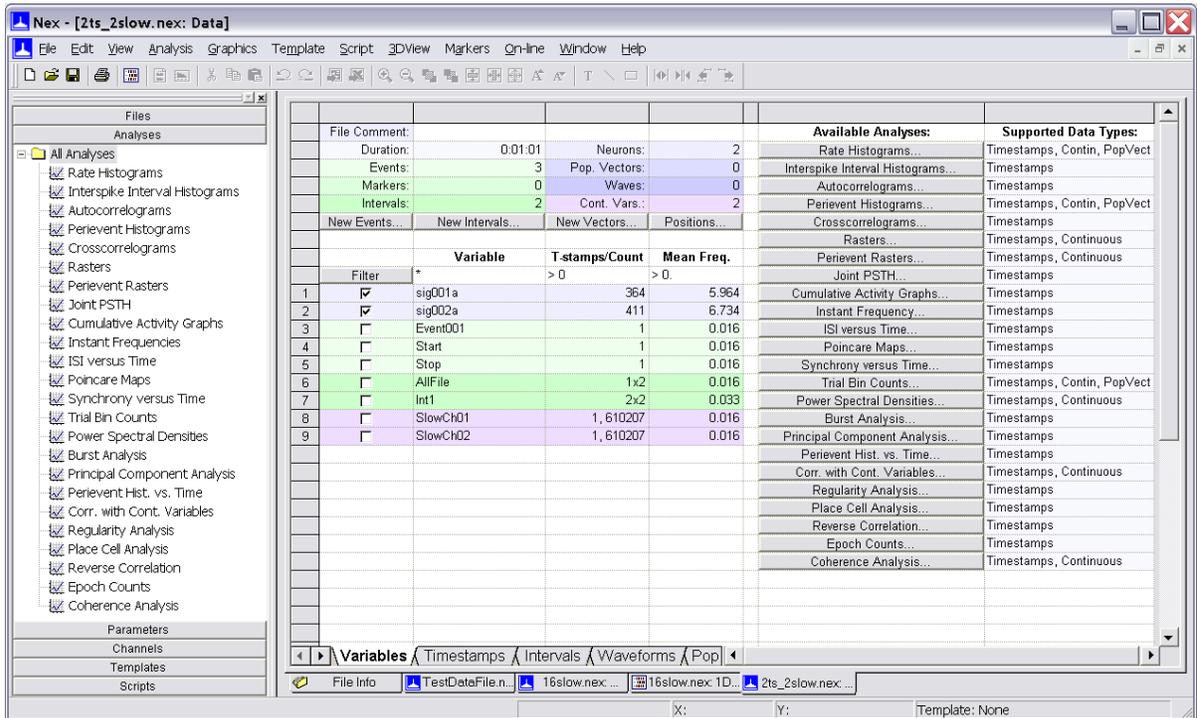
You can also use 1D view to manually add events to the data file. Simply press the left mouse button when the pointer is in the 1D view and NeuroExplorer will add a new timestamp to the variable **ManualEvent**. You can also specify to what Event variable NeuroExplorer will add timestamps when you click in 1D view (use *On mouse click* parameter in the Parameters view).



Use **Analysis | Increase X Range** and **Analysis | Decrease X Range** menu commands to adjust the time range of the windows. You can also use the window controls in the upper-left panel (Parameters window).

## 1.11. Analyzing Data

NeuroExplorer provides a variety of spike train analysis methods. Each method has a number of parameters and options. You can apply any available analysis by pressing the corresponding analysis button in the Data Window (you can also use **Analysis | Go to Analysis** menu commands):



After you press the button corresponding to one of the analyses, NeuroExplorer will open a dialog that will allow you to edit the analysis parameters. When you click OK in this dialog, NeuroExplorer will apply the specified analysis to all the selected variables.

Any combination of analysis parameters and options (together with all the graphics options) can be saved as a template. To save the current configuration as a template, right-click in the Graph Window and select the **Save As New Template** menu command. The template names are shown in the Templates view of the control panel.

When you open a new data file, you can simply double-click on the template name to apply the specific analysis with the selected parameter values.

## 1.12. Selecting Variables for Analysis

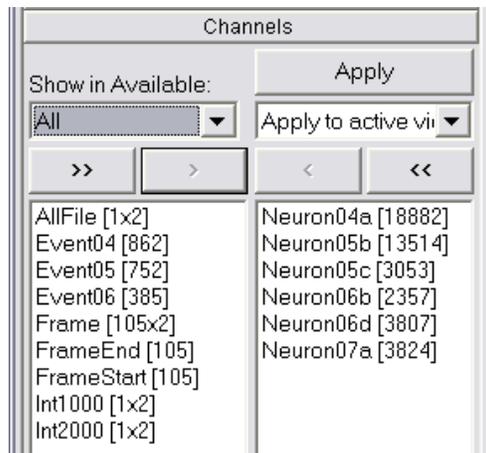
NeuroExplorer can analyze at once any group of variables in the file.  
To select the variables to be analyzed, use one of the following methods:

- Select the variables directly in the Variables Window by using the check boxes located next to the variable names:

	Filter	*
1	<input checked="" type="checkbox"/>	Neuron04a
2	<input checked="" type="checkbox"/>	Neuron05b
3	<input checked="" type="checkbox"/>	Neuron05c
4	<input checked="" type="checkbox"/>	Neuron06b
5	<input checked="" type="checkbox"/>	Neuron06d
6	<input checked="" type="checkbox"/>	Neuron07a
7	<input type="checkbox"/>	Event04
8	<input type="checkbox"/>	Event05
9	<input type="checkbox"/>	Event06
10	<input type="checkbox"/>	FrameEnd
11	<input type="checkbox"/>	FrameStart
12	<input type="checkbox"/>	Frame
13	<input type="checkbox"/>	Int1000

To select or deselect multiple variables in the Variables Window:

- Select several check boxes with the mouse;
  - Press the right mouse button;
  - Choose "Select All" or "Deselect All" in the pop-up menu.
- Select the variables using the Channels view of the Control Panel (see [NeuroExplorer Screen Elements](#)):



The right column shows the selected variables, the left column shows the variables that are not currently selected. To move variables from column to column, first select them, then press ">" (Select) or "<" (Deselect) buttons.

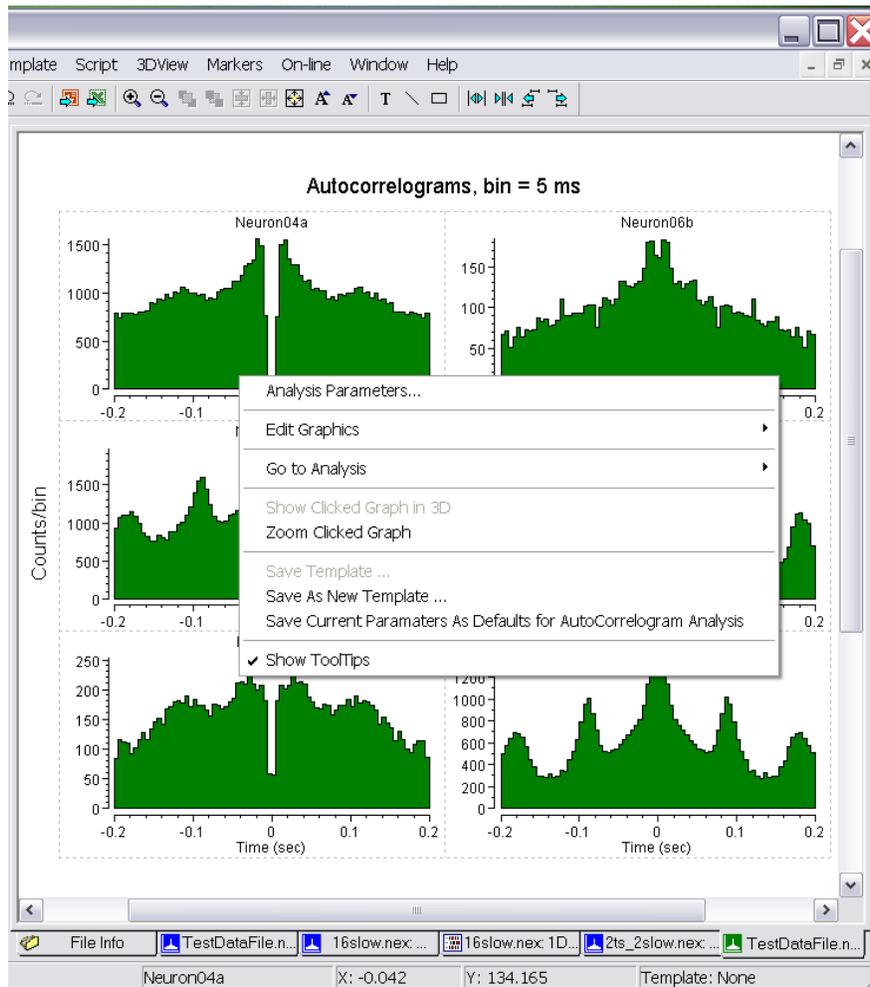
- Select the variables using **Analysis | Select Variables** menu command. This command will invoke the Variable Selection dialog similar to Channels View.

## 1.13. Adjusting Analysis Properties

There are several methods to adjust the analysis parameters:

- Use **Analysis | Edit Parameters** menu command to invoke the Analysis Parameters dialog
- Right-click in the Graph Window and choose **Analysis Parameters** item in the floating menu:

The floating menu is the fastest way to adjust any analysis or graphics parameters. Just double-click (or right-click) anywhere in the Graph Window to invoke this menu:



This menu allows you to go directly to analysis properties, graph properties, axes properties, etc. You can also use it to save the current template and save the current configuration as a new template.

- Single-click in the graph area of the Graph Window and adjust analysis parameters in the Properties Window:

Parameters	
View Num. Res.	Apply (F5)
Function	Perievent Hist
Ref. type	Fixed
Reference	Event04 [862]
XMin (sec)	-3.2
XMax (sec)	3.2
Bin (sec)	0.02
Normalization	Counts/Bin
No selfcount	<input checked="" type="checkbox"/>
Select Data	All
Select Data From (sec)	0
Select Data To (sec)	6200.964625
Int. filter type	Fixed
Interval filter	None
Smooth	None
Sm. width	3
Draw confidence limits	<input type="checkbox"/>

## 1.14. Analysis 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.)
- graph parameters (graph type, graph color, grid lines, etc)
- parameters of X and Y axes (labels, numerics, lines, colors, etc.)
- graph and page labels and other elements

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 Template**.

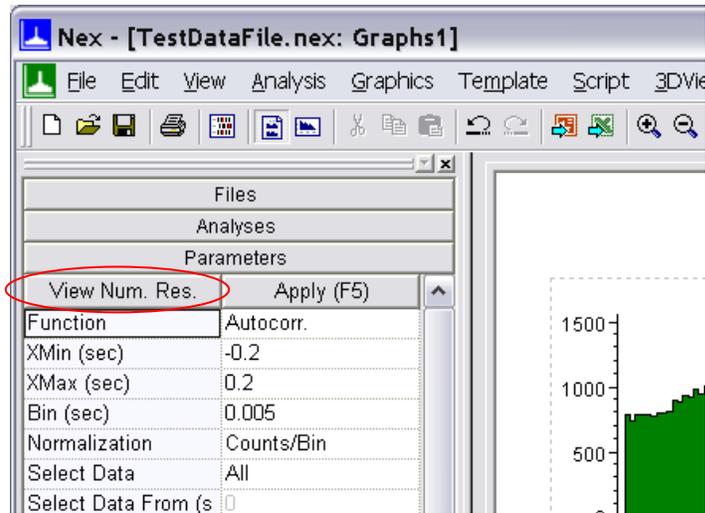
To save the current analysis as a template:

- Select the menu command **Template | Save As New Template**, or
- Right-click in the graph and choose the **Save As New Template** command in the floating menu.

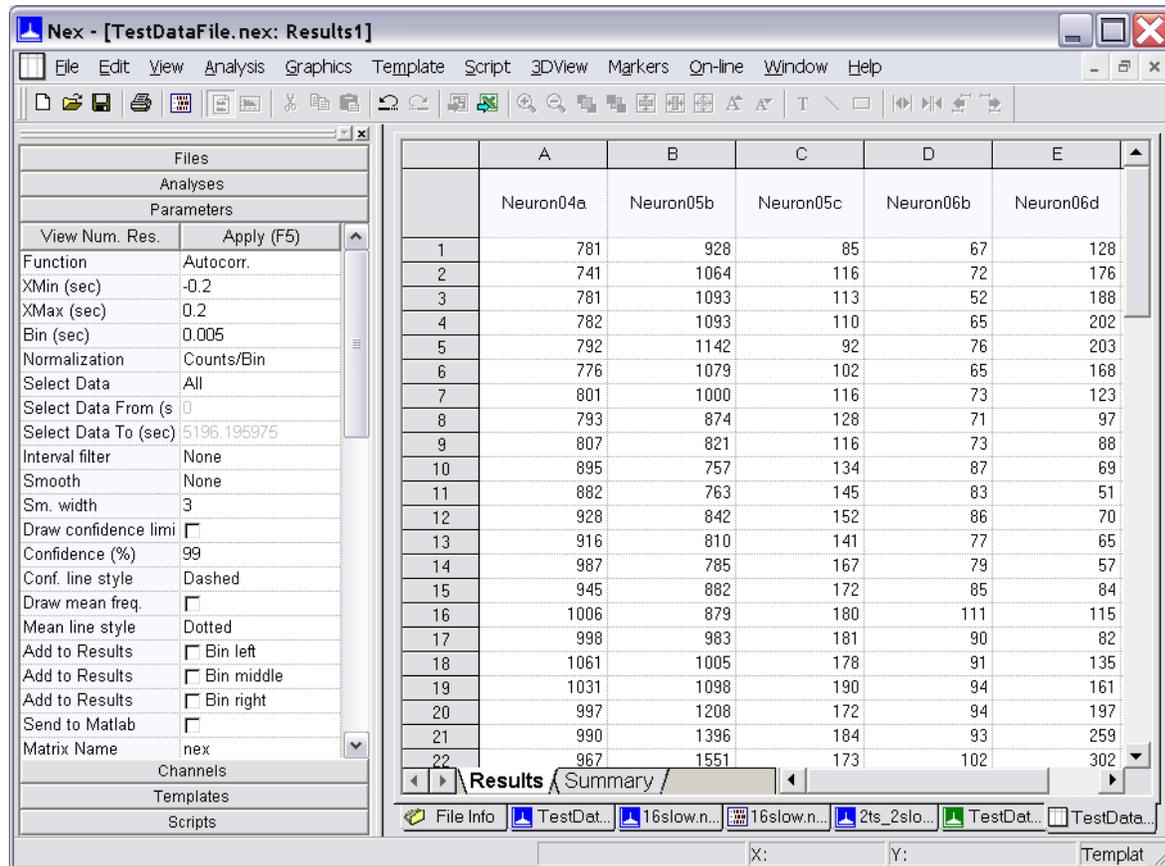
After you saved the template, the name of the template appears in the Templates View (see [NeuroExplorer Screen Elements](#)). You can apply the template to the currently opened file by double-clicking the template name in the Templates View.

## 1.15. Numerical Results

To view the analysis numerical results, select **View | Numerical Results** menu command. You can also press the "View. Num. Res." button in the upper-left corner of the Properties Window:



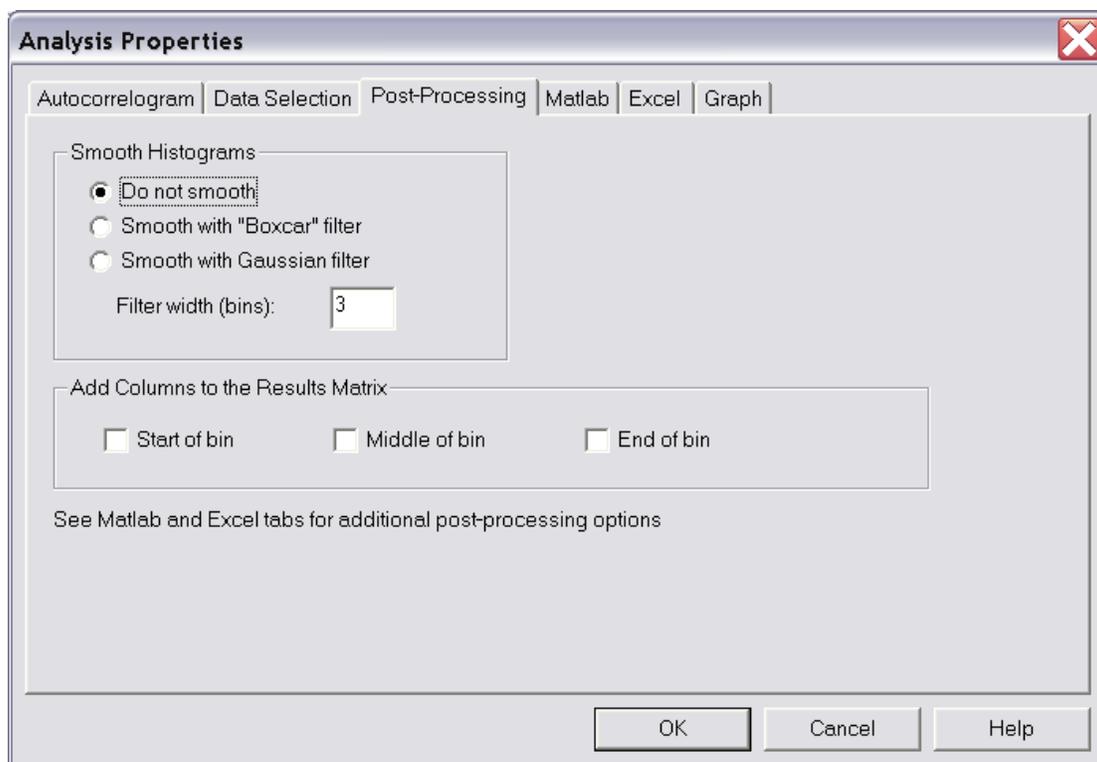
NeuroExplorer will open the Numerical Results Window:



Note that the window has two Excel-style sheets -- the Results sheet and the Summary sheet. Results sheet usually contains bin counts or other histogram values. The Summary sheet contains summary statistics of the analyses such as the mean firing rate, the number of spikes used in the analysis, etc.

## 1.16. Post-processing

For the histogram-style analyses, NeuroExplorer has an optional **Post-processing** analysis step. You can select post-processing options using Post-processing tab in the Analysis Parameters dialog (double-click in Graph window and select **Analysis Parameters** menu item):



You can smooth the resulting histogram with Gaussian or Boxcar filters. You can also add bin information to the matrix of results.

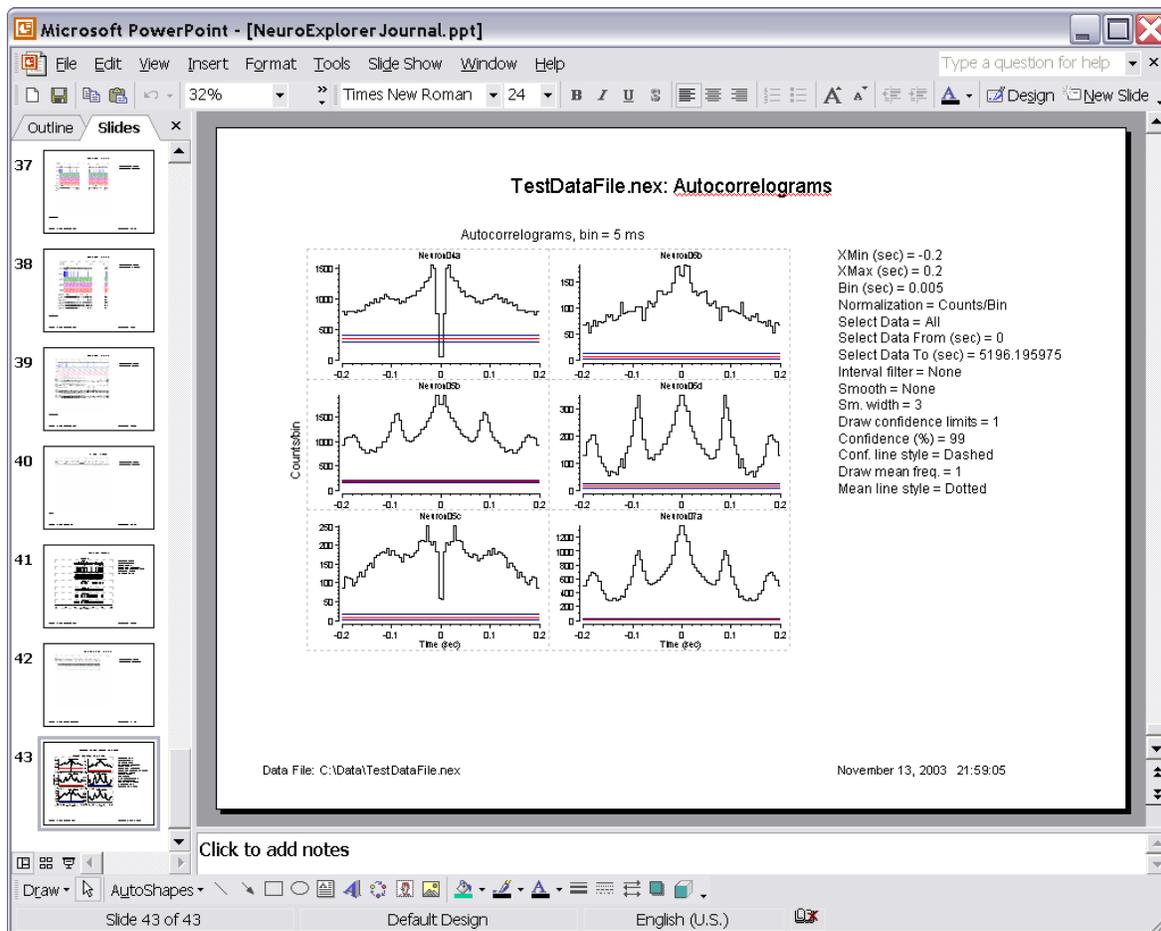
See [Post Processing page in Analysis chapter](#), [Saving Results as Power Point Slides](#), [Working with Matlab](#) and [Working with Excel](#) for more information on NeuroExplorer post-processing options.

## 1.17. Saving Results as Power Point Slides

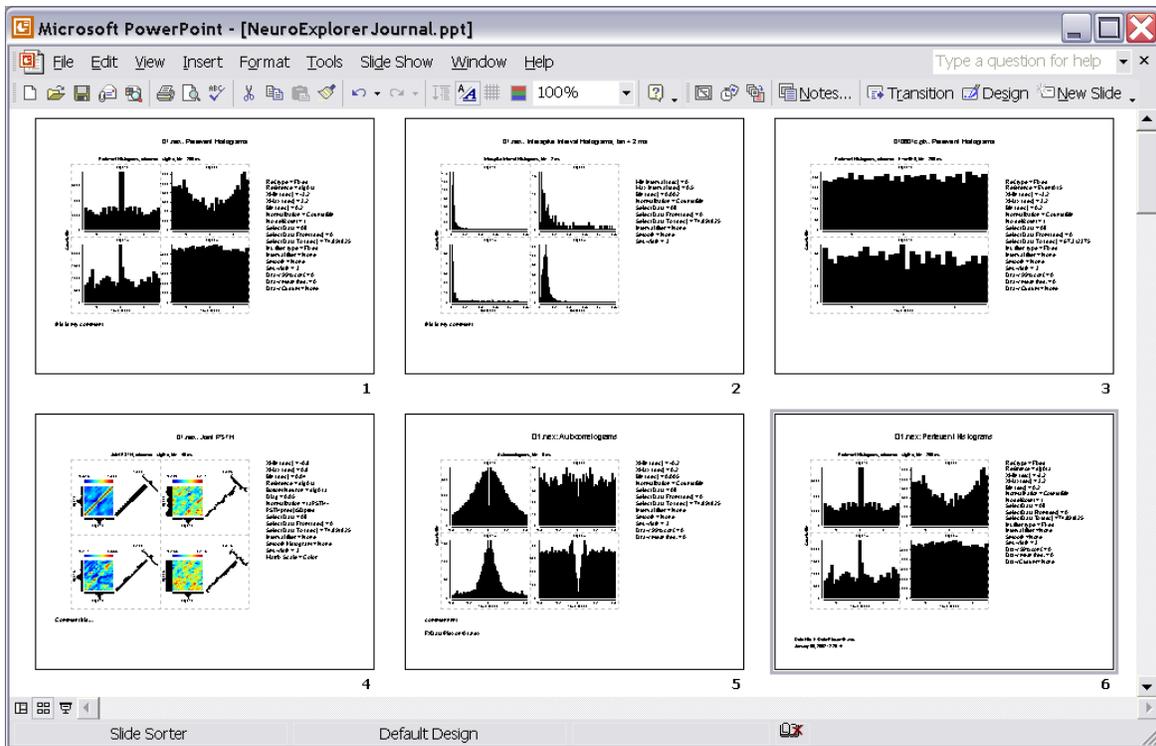
NeuroExplorer version 3 provides a powerful new option that allows you to save graphics, analysis parameters and your annotations as slides in a Power Point Presentation.

To save your results to Power Point, press a toolbar button , or select **Graphics | Export Graphics | Export to Power Point** menu command. NeuroExplorer will start Power Point if Power Point is not running, and will add a new slide to the specified presentation file. The slide will contain:

- Copy of the graphics
- Analysis parameters
- Your comment
- Data file name
- Current date and time



Power point presentation then becomes your lab book where you can save your NeuroExplorer analysis results:



## 1.18. Working with Matlab

NeuroExplorer can interact with Matlab via COM interface using Matlab as a powerful post-processing engine. Immediately after calculating the histograms, NeuroExplorer can send the resulting matrix of histograms to Matlab and then ask Matlab to execute any series of Matlab commands. You need to use Matlab 5.0 or later to be able to use these features of NeuroExplorer.

Use the Matlab tab in the Analysis parameters dialog to specify the matrix name and the Matlab command string.

NeuroExplorer can also send its data variables to Matlab. To transfer the variables to Matlab, use **File | Send Data | Send Selected Variables to Matlab** menu command.

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. Therefore, for a continuous variable (named ContVar1), the following 3 vectors will be created in Matlab:

- **ContVar1** - array of all A/D values in millivolts
- **ContVar1\_ts** - array of timestamps in seconds. Each timestamp is for the first A/D value of the fragment.
- **ContVar1\_ind** - array of indexes. Each index is the position of the first data point of the fragment in the A/D array. If ContVar1\_ind (2) = 201, it means that the second fragment is ContVar1 [201], ContVar1 [202], etc.

You can also import timestamped and continuous variables directly from Matlab. See [Importing Data From Matlab](#) for details.

## 1.19. Working with Excel

The easiest way to transfer data and numerical results from NeuroExplorer to Excel is by using the clipboard:

- Select a range of cells in NeuroExplorer and choose **Edit| Copy**
- Switch to Excel and use the Paste command (select a cell and choose **Edit | Paste**).

NeuroExplorer can also send numerical results directly to Excel. There two ways to send 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.

## 1.20. Saving Graphics

NeuroExplorer can save the contents of the Graph window in a file using the Windows Metafile format. The file can then be opened in any program that can use the *.wmf* format files (a word processor, graphics editor, etc.).

To save the graphics in a metafile, use **File | Save Graphics** menu command.

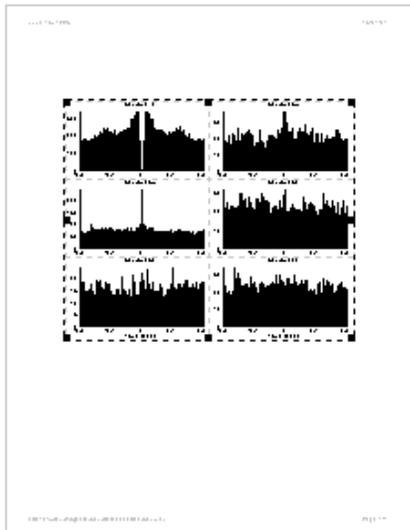
You can also copy the graphics to the clipboard and then paste in another application. To copy the graphics, single-click in the graph area and then select **Edit | Copy** menu command.

See also [Saving Results as Power Point Slides](#).

## 2. Working with Graphics

### 2.1. NeuroExplorer Graphics

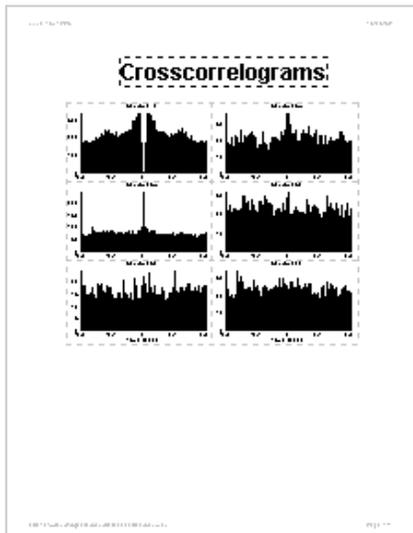
NeuroExplorer Graph Window shows the graphs as they will be printed on a page. If you press **Zoom Out** button  several times, you'll see the whole page with the graphs, header and footer:



NeuroExplorer Graph Window shows one or more graphics objects. The page to the left has only one object - a block of graphs. You can add more objects by using **Insert** menu commands.

For example, to add a label to the NeuroExplorer Graph Window:

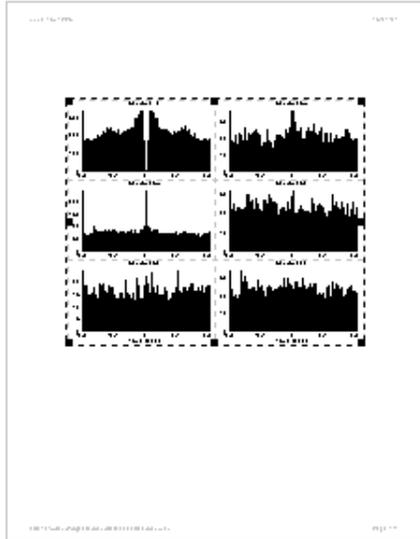
- select **Insert | Text** menu command
- click in the Graph window where you want the new label to be placed
- edit the label properties in the Text Dialog



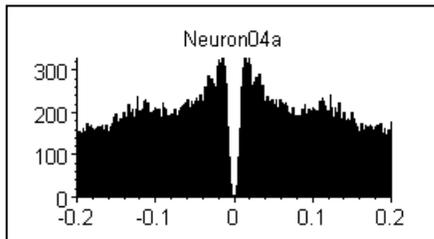
Here is the page shown above with an additional Text object ("Crosscorrelograms" label).

## 2.2. Graphics Modes

The standard display of the NeuroExplorer Graph Window shows the graphics in the Page Mode:



In this mode, you can add, delete and edit **page** graphics objects, i.e. the graphics elements that you want to have on the page.



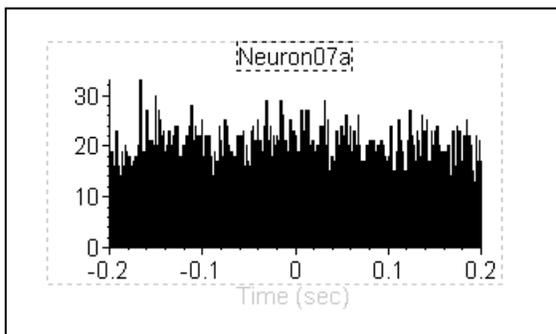
Each **graph** in the page, however, has its own set of graphics objects. Thus, the default graph usually contains the text label that displays the variable name (Neuron04a in this picture).

To edit the properties of the graphical objects (like the label "Neuron04a" above) that belong to the **graph**, you need to switch the Graph Window to the **Graph Mode**.



Use **Edit | Graph Layout Mode** and **Edit | Page Layout Mode** commands to switch the graphics mode in NeuroExplorer. You can also use the toolbar buttons shown to the left to switch modes.

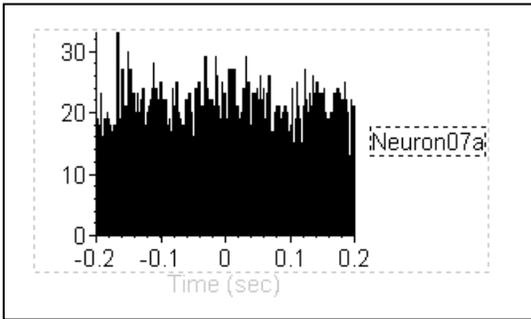
The left button will switch the Graph Window to the Page Mode, the right button will switch it to the Graph Mode.



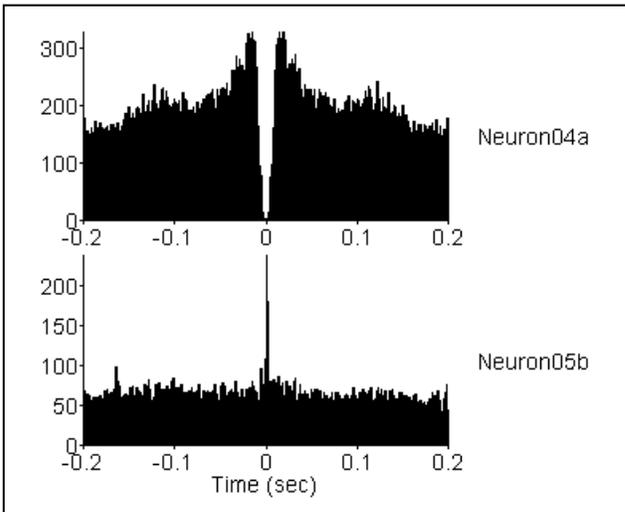
Here is the typical picture in the Graph Mode:

Only one graph is visible in this mode. While in the Graph Mode, you can select, drag, resize and edit the graphical objects that will appear in every graph.

For example, to move the variable name label to a different position relative to the graph:



- Switch to the Graph Mode as described above
- Click on the text label to select it
- Drag the label to a different position



Switch back to Page Mode.

Note that the labels of **all** graphs are now in new positions.

## 2.3. Positioning Graphics Objects

Every graphical object in NeuroExplorer has its parent. Thus, the Block of Graphs is a parent of all the page graphics objects. The Graph is a parent of all the objects inside the graph (X and Y axes, variable label, etc.).

A position of any graphics object is always specified as relative to its parent. When you move or resize the parent, other objects will move together with their parent.

There are several types of coordinate units that you can use to position the objects.

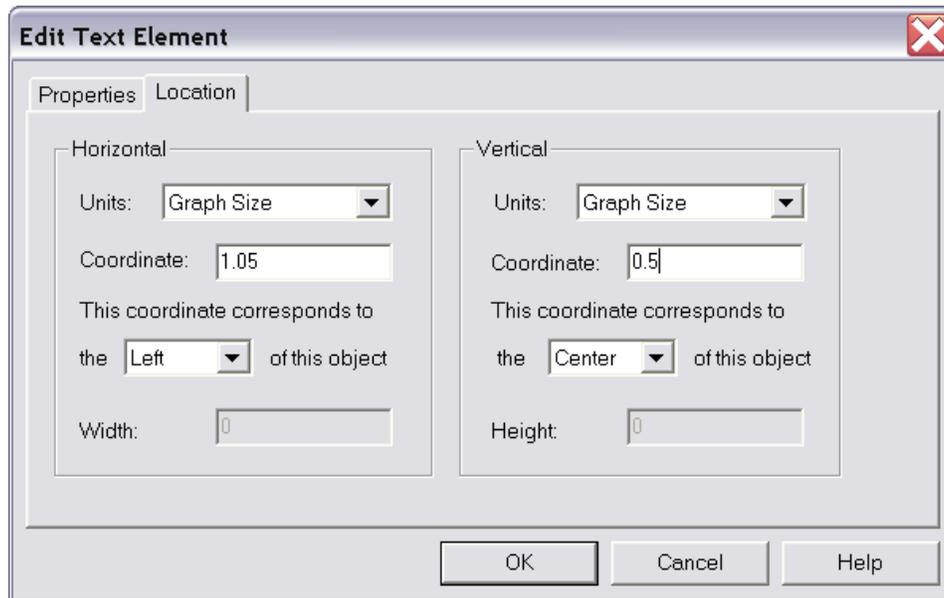
For X coordinates, NeuroExplorer has the following options:

- **Graph Width.** With these coordinate units, 0.0 corresponds to the left edge of the parent, 1.0 corresponds to the right edge of the parent.
- **Inches From Left.** Here 0.0 corresponds to the left edge of the parent. If you want an object to be 0.3 inches to the left of the parent, specify X as -0.3.
- **Inches From Right.** Here 0.0 corresponds to the right edge of the parent. If you want an object to be 0.3 inches to the right of the parent, specify X as 0.3.
- **Graph Data.** With these coordinate units, X coordinate corresponds to the graph X axis.

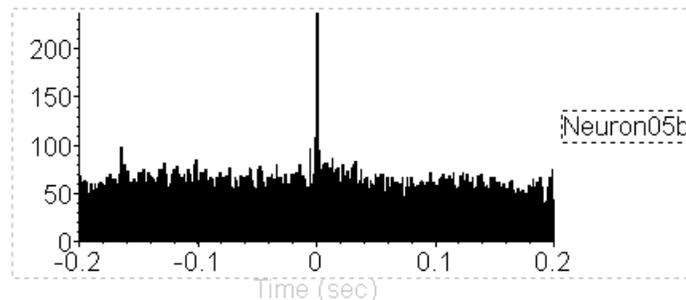
For Y coordinates, NeuroExplorer has the following options:

- **Graph Height.** With these coordinate units, 0.0 corresponds to the bottom edge of the parent, 1.0 corresponds to the top edge of the parent.
- **Inches From Bottom.** Here 0.0 corresponds to the bottom edge of the parent. If you want on object to be 0.3 inches below of the parent, specify Y as -0.3.
- **Inches From Top.** Here 0.0 corresponds to the top edge of the parent. If you want on object to be 0.3 inches above the parent, specify Y as 0.3.
- **Graph Data.** With these coordinate units, Y coordinate corresponds to the graph Y axis.

Use Location tab in the graphics object Edit dialog to specify the object's position:



The coordinates and alignment options shown in this dialog are for the text label in the following graph:



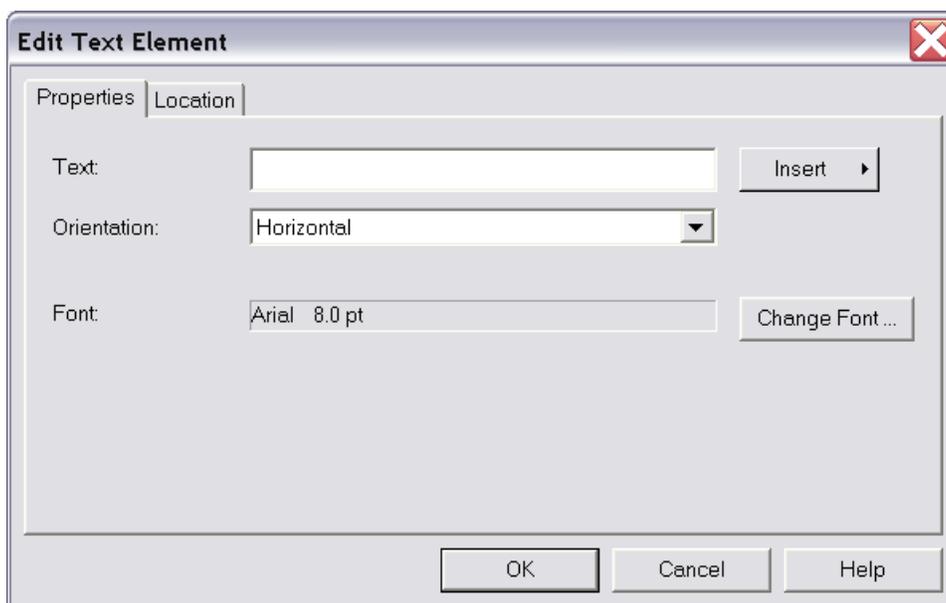
The text object is located to the right of the graph, so the X coordinate is 1.05 with the Graph Size units and horizontal alignment Left.

The object is centered vertically relative to the graph, so its Y coordinate is 0.5 with the Graph Size units and vertical alignment Center.

## 2.4. Text Labels

To add a label in the NeuroExplorer Graph Window:

- Select **Insert | Text** menu command
- Move mouse pointer to the Graph Window. Note that the cursor changes to the picture of a hand
- Click in the Graph window where you want the new label to be placed
- Edit the label properties in the displayed Text Dialog



To specify the font, press **Change Font** button.

To specify the exact position of the text object, use the **Location** page of the dialog. See [Positioning the Graphics Objects](#) for details.

To specify the text of the label, type the text of the label in the **Text** field.

In many cases there is a need to specify a text that depends on the opened data file, the variable in the graph, etc. NeuroExplorer uses **Template Strings** to accomplish this task.

**Template String** is a string enclosed in brackets (for example, *<VarName>*). During the drawing process, NeuroExplorer tries to find the actual string for each of the template strings. If the actual string is found, the template string is replaced by the actual string. For example, if the variable used in the graph is *Neuron04a*, the *<VarName>* string in the text object is replaced by *Neuron04a*.

**Insert** button in the Text Edit Dialog opens a menu that allows you to insert various Template Strings into the text object. You can also manually type in the template strings into the text field of the dialog.

Here is the list of available template strings:

**<VarName>** - replaced by the name of the variable used in a graph. Works in the graph labels only.

**<ColVarName>** - replaced by the name of the variable used for a column of graphs. Works in the page labels only when the analysis has a reference variable and the reference type is Table.

**<RowVarName>** - replaced by the name of the variable used for a row of graphs. Works in the page labels only when the analysis has a reference variable and the reference type is Table.

**<Bin>** - replaced by the bin value in seconds.

**<BinMS>** - replaced by the bin value in milliseconds.

**<RefName>** - replaced by the name of the reference variable. Works only when the analysis has a reference variable.

**<NumRef>** - replaced by the number of reference events. Works only when the analysis has a reference variable.

**<FileName>** - replaced with the name of the data file.

**<FilePath>** - replaced with the full path of the data file.

**<Date>** - replaced with current date.

**<Time>** - replaced with current time.

**<Smooth>** - replaced with one of the three strings "None", "Boxcar" or "Gaussian", depending of the current smooth selection. Works with histogram-type analyses only.

**<SmoothWidth>** - replaced with the smooth filter width. Works with histogram-type analyses only.

**<IntFilter>** - replaced with the name of the Interval Filter.

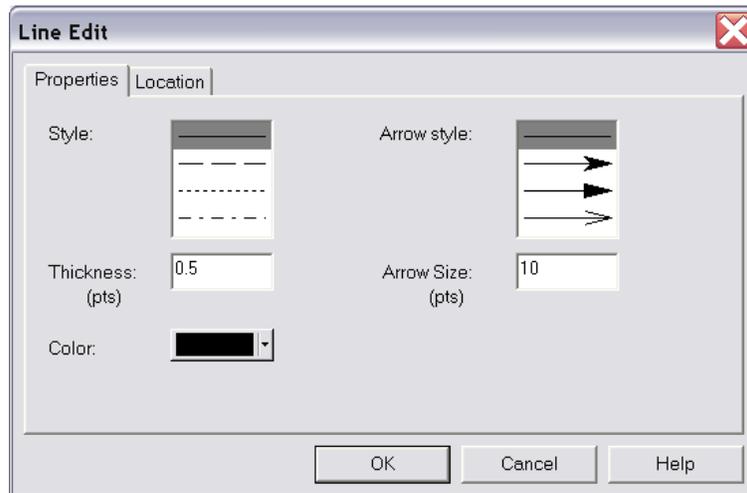
**<FirstInt>** - replaced with the first interval of the Interval Filter (if Interval Filter is specified), or the interval from 0 to the end of experiment (if the filter is not specified).

## 2.5. Lines

To add a line in the NeuroExplorer Graph Window:

- Select **Insert | Line** menu command
- Move mouse pointer to the Graph Window. Note that the cursor changes to the picture of a hand
- Click in the Graph window where you want the new line to start and keep the left mouse button down
- Drag the mouse pointer to the place where you want the new line to end
- Release the mouse button

To edit the properties of the line, double-click at the line. The following dialog will be displayed:



Line thickness and the size of the arrow are specified in points (1/72th of the inch).

To specify the exact position of the line, use the **Location** page of the dialog. See [Positioning the Graphics Objects](#) for details.

## 2.6. Rectangles

To add a rectangle in the NeuroExplorer Graph Window:

- Select **Insert | Rectangle** menu command
- Move mouse pointer to the Graph Window. Note that the cursor changes to the picture of a hand
- Click in the Graph window where you want to position the top-left corner of the new rectangle and keep the left mouse button down
- Drag the mouse pointer to the place where you want to position the bottom-right corner of the new rectangle
- Release the mouse button

To edit the properties of the rectangle, double-click anywhere in the rectangle and specify rectangle parameters in Rectangle Properties dialog.

To specify the exact position of the rectangle, use the **Location** page of the dialog. See [Positioning the Graphics Objects](#) for details.

## 3. NeuroExplorer Analysis Reference

### 3.1. Introduction

This chapter, *NeuroExplorer Analysis Reference*, covers the following topics:

- [Data Selection Options](#)
- [Post-Processing Options](#)
- [Matlab Options](#)
- [Excel Options](#)
- [Confidence Limits for Perievent Histograms](#)
- [Rate Histograms](#)
- [Interspike Interval Histograms](#)
- [Autocorrelograms](#)
- [Perievent Histograms](#)
- [Crosscorrelograms](#)
- [Rasters](#)
- [Perievent Rasters](#)
- [Joint PSTH](#)
- [Cumulative Activity Graphs](#)
- [Instant Frequency Graphs](#)
- [Interspike Intervals vs Time](#)
- [Poincare Maps](#)
- [Spike Distance vs Time](#)
- [Trial Bin Counts](#)
- [Power Spectral Densities](#)
- [Burst Analysis](#)
- [Principal Component Analysis](#)
- [Perievent Histograms vs. Time](#)
- [Correlations with Continuous Variables](#)
- [Regularity Analysis](#)
- [Place Cell Analysis](#)
- [Reverse Correlation](#)
- [Epoch Counts](#)
- [Coherence Analysis](#)
- [Spectrogram Analysis](#)

## 3.2. Data Types

### 3.2.1. Spike Trains

In NeuroExplorer, the most commonly used data type is a spike train. A spike train in NeuroExplorer does not contain the spike waveforms, it represents only the spike timestamps (the times when the spikes occurred). A special [Waveform](#) data type is used to store the spike waveform values.

[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. (for example, spike trains contain positions of neurons used in the [3D activity "movie"](#)).

Internally, the timestamps are stored as 32-bit signed integers. These integers are usually the timestamps recorded by the data acquisition system and they represent the time in the so-called time ticks. For example, the typical time tick for the Plexon system is 25 microseconds, so an event recorded at 1 sec will be stored internally as 40000.

#### Limitations

The maximum number of timestamps (in each spike train) in NeuroExplorer is 2,147,483,647. In reality, the limiting factor is the amount of virtual memory on your machine, since the timestamps of all the variables of a data file should fit into memory. NeuroExplorer requires that in each spike train all the timestamps are in ascending order, that is

$\text{timestamp}[i+1] > \text{timestamp}[i]$  for all  $i$ .

NeuroExplorer also requires that

$\text{timestamp}[i] \geq 0$  and  $\text{timestamp}[i] < 2,147,483,647$  for all  $i$ .

The upper timestamp limit defines the maximum experiment length that can be safely analyzed in NeuroExplorer. Thus, for the standard Plexon setup, the maximum duration of the experiment is  $2147483647/40000$  seconds, or 14 hours and 54 minutes.

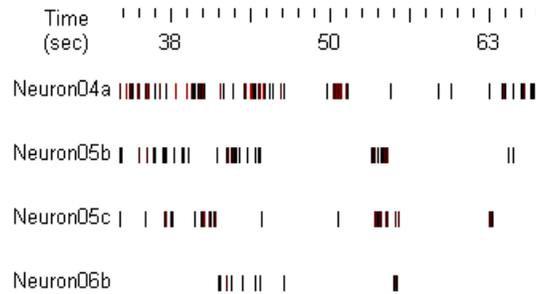
#### Timestamped Variables in NexScript

You can get access to any timestamp in the current file. For example, to assign the value 0.5 (sec) to the third timestamp of the variable DSP01a, you simply write:

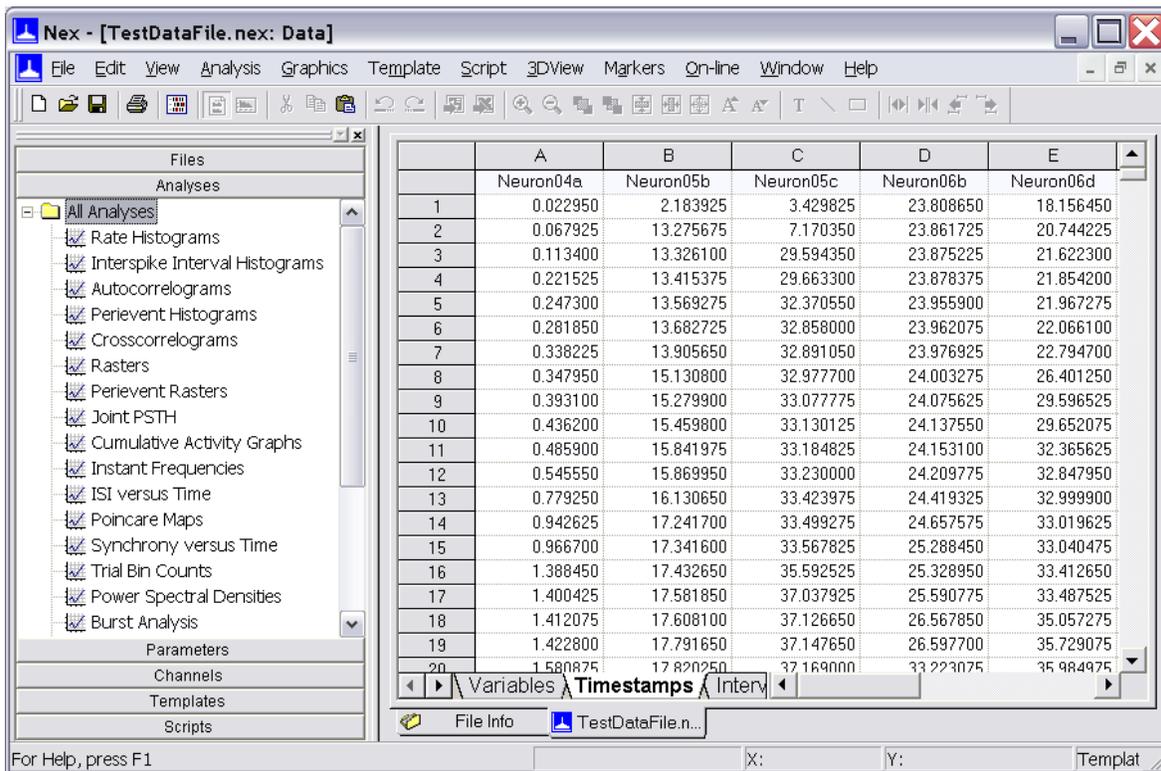
```
doc = GetActiveDocument()  
doc.DSP01a[3] = 0.5
```

#### Viewers

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:



### 3.2.2. 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.

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 Train](#) timestamps. See [Spike Trains](#) for more information about the timestamps in NeuroExplorer.

## Creating Event Variables

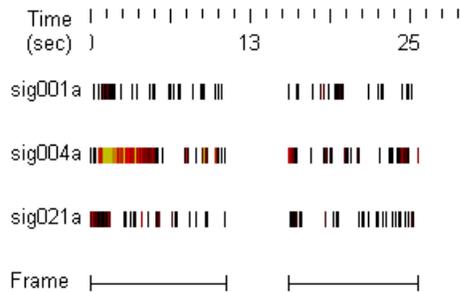
You can add events to the data file using **Copy | Paste** command in the Timestamps view (see [Importing Data from Spreadsheets](#) for details).

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

## 3.2.3. Intervals

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

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



## Creating Interval Variables

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**

[Burst analysis](#) creates interval variables (one for each neuron) that contain all the detected bursts as time intervals.

## Interval Variables in 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, 2)

doc.MyInterval[1,1] = 0.
doc.MyInterval[1,2] = 100.

doc.MyInterval[2,1] = 200.
doc.MyInterval[2,2] = 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).

## 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:

	A	B	C	D	E
	Frame start	Frame end	Int1000 start	Int1000 end	Int2000 start
1	0.000000	26.597725	0.000000	1000.000000	1000.000000
2	27.597700	69.077725			
3	70.077700	113.890200			
4	114.890175	144.160200			
5	145.160175	202.954175			
6	203.954150	250.574175			
7	251.574150	281.195175			
8	282.195150	302.728350			
9	303.728325	344.378350			
10	345.378325	397.168350			
11	398.168325	467.648350			
12	468.648325	523.538350			
13	524.538325	586.098350			
14	587.098325	645.628350			
15	646.628325	681.598350			
16	682.598325	707.798350			
17	708.798325	759.058350			
18	760.058325	841.984325			

### 3.2.4. 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.

#### Viewers

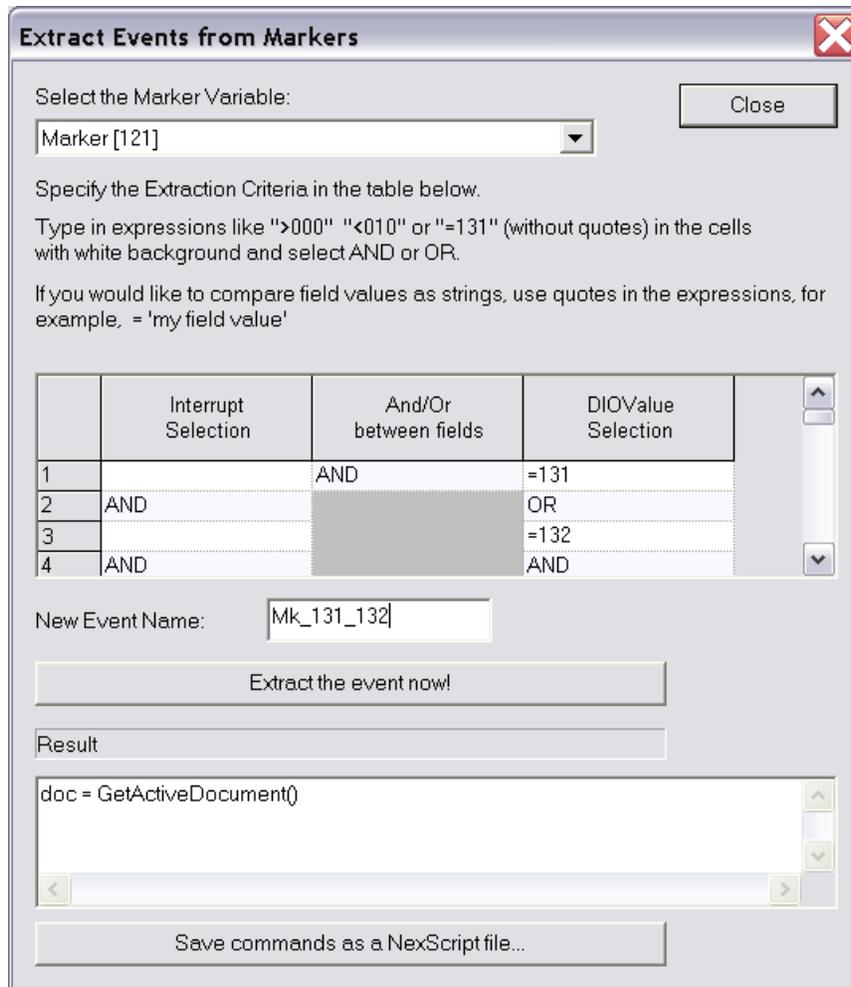
You can view both the marker timestamps and additional marker information in the Markers tab of the Data window:

	A	B	C
	Marker Timestamp	Marker Interrupt	Marker DIOValue
1	41.802500	001	000
2	42.009200	002	000
3	165.859500	003	131
4	168.110400	004	132
5	168.360500	005	132
6	168.610700	006	132
7	168.860700	007	132
8	169.110900	008	132
9	169.361000	009	132
10	169.611100	010	132
11	169.861200	011	132
12	170.111300	012	132
13	170.361400	013	132
14	170.611500	014	132
15	170.861600	015	132
16	171.111700	016	132
17	171.361800	017	132
18	171.611900	018	132
19	171.862000	019	132
20	171.892100	020	131

#### Extracting Events

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

For example, you may want to extract from the marker variable shown in the figure above only the markers with DIOValue 131 or 132. **Marker | Extract...** menu command will open the Extract dialog box in which you can specify multiple criteria for extracting events from the marker variable:



### 3.2.5. 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.

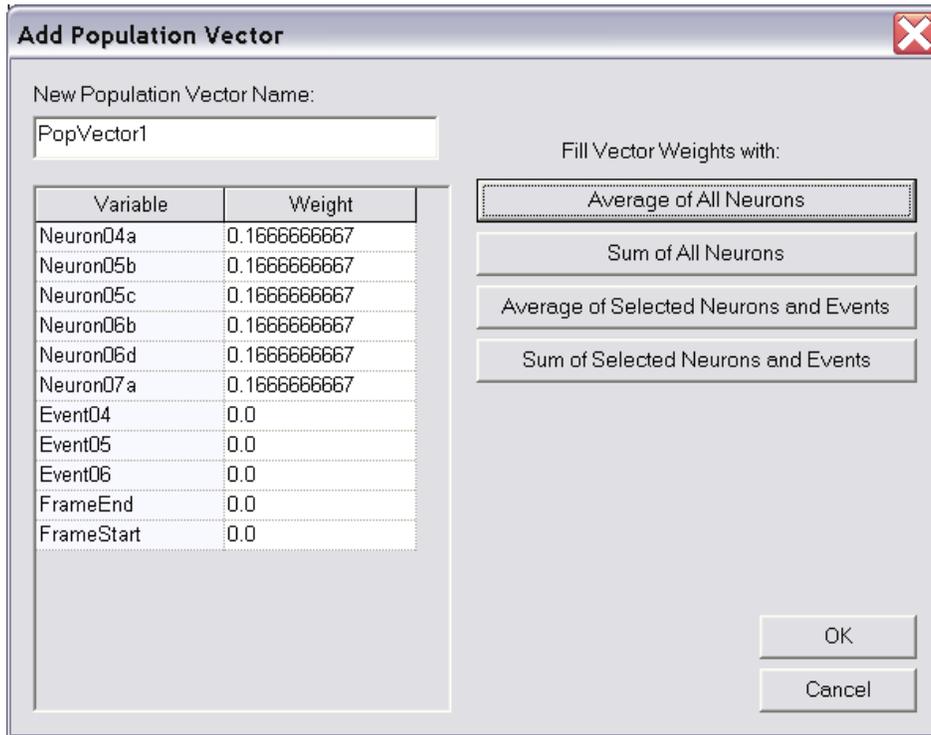
Population vector assigns a weight to each spike train or event variable in the file. You can then use population vectors in the following analyses:

- Rate Histograms
- Perievent Histograms
- Trial Counts

If you select the population vector for analysis and then run one of the three 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:



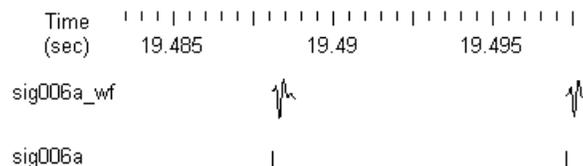
### Creating Population Vectors

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

[Principal Component Analysis](#) creates a set of population vectors based on correlations between activities of individual neurons.

## 3.2.6. Waveforms

The waveform data type is used in NeuroExplorer to store the spike waveform values together with the spike timestamps. The following figure shows the waveform variable sig006a\_wf together with the corresponding spike train sig006a:



The waveform data may not be imported by default from the data files created by the data acquisition systems. Use **View | Data Import Options** menu command to specify which data types NeuroExplorer will import from the data files.

The waveform variables can be used in the following analyses:

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

## Viewers

You can view the waveforms of the selected variables in the graphical display (**View | 1D Data Viewer** menu command, see figure above). Numerical values of the waveforms and their timestamps are shown in the **Waveforms** sheet of the Data view.

	Variable		1	2	3
	sig001a_wf	TimeStamp	w/f value 1	w/f value 2	w/f value 3
1		0.027850	3.311821	2.420177	1.974355
2		0.537000	-1.528533	-2.738621	-3.184443
3		0.580375	4.840353	6.305197	8.024796
4		1.189150	1.273777	4.394531	7.897418
5		3.003075	0.700577	1.337466	1.019022
6		4.336100	-9.425951	-7.897418	-3.566576
7		4.374075	-7.578974	-5.986753	-2.356488
8		4.775850	9.425951	9.298573	10.317595
9		5.360400	-2.038043	-1.082711	0.636889
10		5.459625	-1.019022	1.65591	6.878397
11		5.933150	2.674932	0.191067	-0.636889
12		6.135750	1.082711	4.012398	5.540931
13		7.299375	-6.496264	-4.012398	3.37551
14		7.441250	-2.038043	0	2.292799
15		7.460925	-3.948709	-4.012398	-2.165421
16		7.791700	2.483865	3.948709	6.942086
17		7.925500	0.382133	2.674932	3.566576
18		8.660975	9.871773	11.018173	11.591372
19		8.744100	0.636889	2.101732	3.630265
20		8.883725	0.191067	1.337466	3.120754
21		9.026025	-5.540931	-3.630265	0.636889

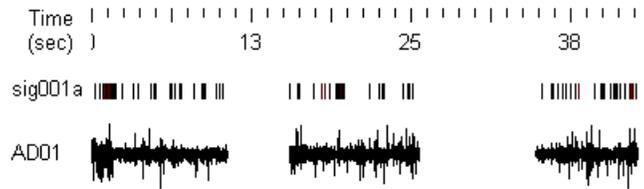
To select a different waveform variable, click in the cell located in the row 1 of the Variable column (the cell shows sig001a\_wf in the figure above).

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

### 3.2.7. Continuously Recorded Data

The continuous data type is used in NeuroExplorer to store the data that has been continuously recorded (digitized).

The following figure shows the continuous variable AD01 together with the spike train sig001a:



Continuous data may not be imported by default from the data files created by the data acquisition systems. Use **View | Data Import Options** menu command to specify which data types NeuroExplorer will import from the data files.

Continuous variables can be used in the following analyses:

- Rate Histograms
- Rasters
- Perievent Histograms (instead of PST, NeuroExplorer calculates the spike-triggered average for a continuous variable)
- Perievent Rasters (NeuroExplorer shows the values of continuous variable using color)
- Spectral Densities
- Correlations with Continuous Variable
- Coherence Analysis
- Spectrogram Analysis

### Limitations

Internally, each data point of the continuous variable is stored as a 2-byte integer. The maximum number of values (in each continuous variable) in NeuroExplorer is 2,147,483,647. In reality, the limiting factor is the amount of virtual memory on your machine, since all the values should fit into virtual memory.

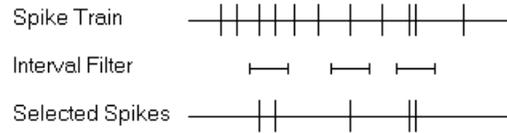
### Viewers

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

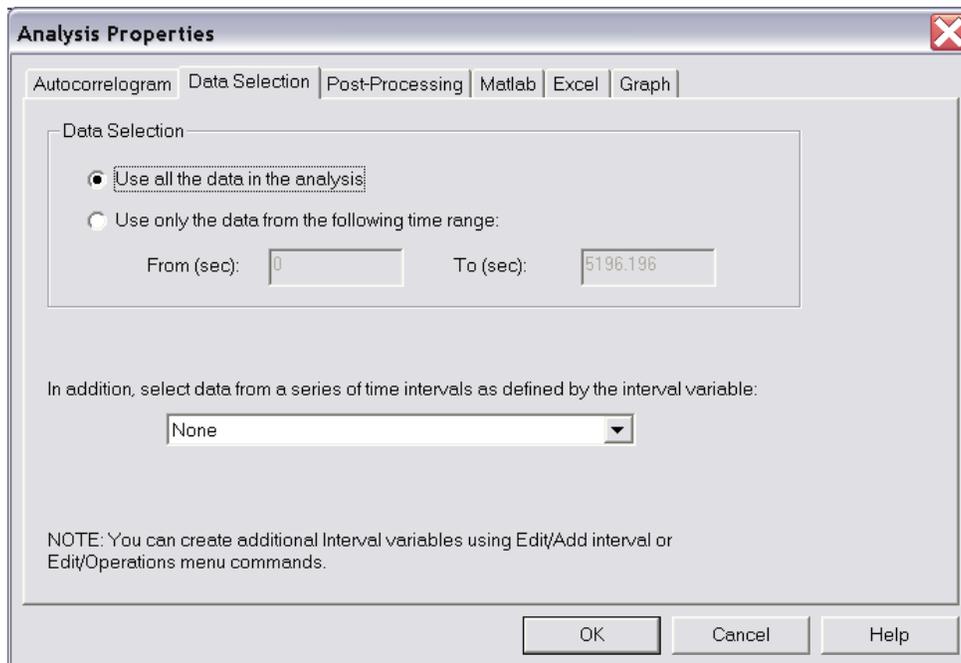
Numerical values of the continuous variables are shown in the **Continuous** sheet of the Data view.

### 3.3. 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:



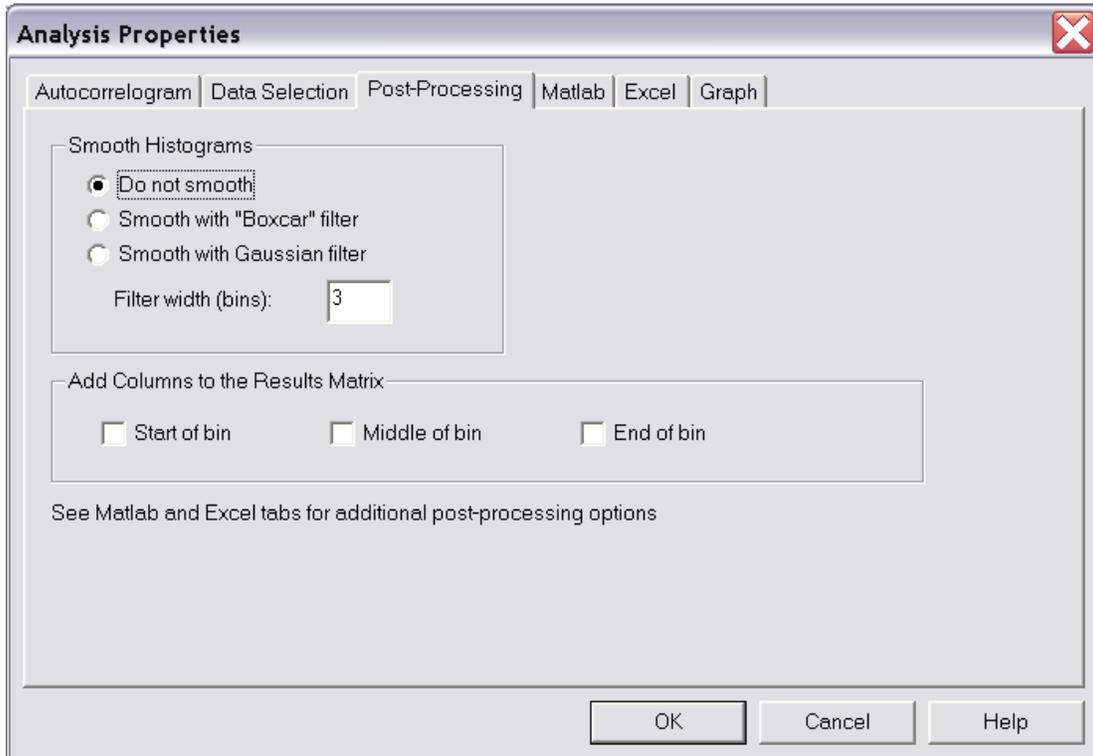
You can also choose an interval variable (at the bottom of the Data Selection page) as an additional data filter. NeuroExplorer then will select data from one or more time intervals as specified by the interval variable.

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.

### 3.4. Post-Processing Options

For the histogram-style analyses, NeuroExplorer has an optional **Post-processing** analysis step. You can select post-processing options using Post-processing tab in the Analysis Parameters dialog.

(double-click in the Graph window and select **Analysis Parameters** menu item):



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

$$f[i] = 1/w \text{ for all } i$$

Thus for the filter 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 Gaussian filter is calculated using the following formula:

$$f[i] = \exp(-i*i/sigma)/norm, i \text{ from } -2*d \text{ to } 2*d$$

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, filter width can be non-integer (for example, 3.5).

See [Matlab Options](#) and [Excel Options](#) for more information on NeuroExplorer post-processing capabilities.

### 3.5. 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. You need to use Matlab 5.0 or later to be able to use these features of NeuroExplorer.

Use the Matlab tab in the Analysis parameters dialog to specify the matrix name and the Matlab command string.

### 3.6. 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.

You need to use Excel 97 or later to be able to use this feature of NeuroExplorer.

### 3.7. Confidence Limits for Perievent Histograms

If the total time interval (experimental session) is T (seconds) and we have N spikes in the interval, then the neuron frequency is:

$$F = N/T$$

Then 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:

$$C = P*N_{Ref}, \text{ where } N_{Ref} \text{ is the number of the reference events.}$$

The value C is used for drawing the Mean Frequency in the Perievent Histograms and Cross- and Autocorrelograms.

The confidence limits for C are calculated using the assumption that C has a Poisson distribution. Assume that a random variable S has a Poisson distribution with parameter C. Then the 99% confidence limits are calculated as follows:

$$\begin{aligned} \text{Low Conf.} &= x \text{ such that } \text{Prob}(S < x) = 0.005 \\ \text{High Conf.} &= y \text{ such that } \text{Prob}(S > y) = 0.005 \end{aligned}$$

If  $C < 30$ , NeuroExplorer uses the actual Poisson distribution

$$\text{Prob}(S = K) = \exp(-C)*C^K/K!$$

to calculate the confidence limits.

If  $C \geq 30$ , the Gaussian approximation is used:

$$\begin{aligned} \text{Low Conf.} &= C - 2.58*\text{sqrt}(C); \\ \text{High Conf.} &= C + 2.58*\text{sqrt}(C); \end{aligned}$$

#### Reference

Abeles M. Quantification, smoothing, and confidence limits for single-units' histograms. Journal of Neuroscience Methods. 5(4):317-25, 1982

### 3.8. Cumulative Sum Graphs

Here is the algorithm that is used to draw optional cumulative sum graphs above the histograms.

Suppose we have a histogram with bin counts  $bc[i]$ ,  $i=1,\dots,N$ . 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:

- $A$  equals to average of all  $bc[i]$  if you select "Use all histogram" option
- $A$  equals to average of all  $bc[i]$  for bins that are before zero on time scale if you select "Use preref as base" option.

If you use "Use all histogram" option, the value of the cumulative sum for the last bin is always zero:  $sc[N] = bc[1]+bc[2]+\dots+bc[N] - A*N$ , where  $A = (bc[1]+bc[2]+\dots+bc[N])/N$ .

NeuroExplorer displays 99% confidence limits for Cumulative Sum Graphs. Confidence limits for "Use preref as base" option are proportional to square root of bin number.

Calculations of confidence limits for "Use all histogram" option are not trivial. These calculations are based on the formulas developed by the author of NeuroExplorer, Alexander Kirillov.

## 3.9. Rate Histograms

### Parameters

**XMin** - time axis minimum in seconds

**XMax** - time axis maximum in seconds.

**Bin** - bin size in seconds.

**Normalization** - histogram units (**Counts/Bin** or **Spikes/Second**).

**Select Data, Select Data From, Select Data To, Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### 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.

For each bin, the number of events 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**.

## 3.10. Interspike Interval Histograms

### Parameters

**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**).

**Select Data, Select Data From, Select Data To, Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### Algorithm

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.

## 3.11. Autocorrelograms

### Parameters

**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**).

**Select Data, Select Data From, Select Data To, Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Draw 99% confidence limits** - option to draw two horizontal lines representing the confidence limits for the autocorrelogram. See [Confidence Limits](#) for details.

**Draw mean firing rate** - option to draw a horizontal line representing the expected histogram value for the Poisson spike train. See [Confidence Limits](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### 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  $ts[k]$ :

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

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

2) for each  $i$  except  $i$  equal  $k$ :

if  $d[i]$  is inside the first bin, increment the bin counter for the first bin:  
if  $d[i] \geq XMin$  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  $d[i] \geq XMin+Bin$  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  $d[i] \geq XMin$  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.

## 3.12. Perievent Histograms

### Parameters

**Reference Type** - a choice between single and multiple reference events.

**Reference** - reference neuron or event.

**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**).

**No Selfcount** - option not to count reference events if the target event is the same as the reference event.

**Select Data**, **Select Data From**, **Select Data To**, **Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth**, **Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Draw 99% confidence limit** - option to draw two horizontal lines representing the confidence limits for the perievent histogram. See [Confidence Limits](#) for details.

**Draw mean firing rate** - option to draw a horizontal line representing the expected histogram value for the Poisson spike train. See [Confidence Limits](#) for details.

**Draw Cumulative Sum** - option to draw a cumulative sum graph above the histogram. See [Cumulative Sum Graphs](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Background, Peak/Trough Width, Left Shoulder, Right Shoulder** – parameters of the peak and trough analysis described in the Algorithm section of the Crosscorrelogram analysis options.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

## Algorithm

In general, the Perievent Histogram shows the conditional probability of a spike in the spike train at time  $t$  on the condition that there is a reference event (or reference 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  $ref[i]$  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  $d[i] \geq XMin$  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  $d[i] \geq XMin+Bin$  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 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  $d[i] \geq XMin$  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.

### 3.13. Crosscorrelograms

#### Parameters

**Reference Type** - a choice between single and multiple reference events.

**Reference** - reference neuron or event.

**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**).

**Count Bins In Filter** – an option to normalize each bin individually if the interval filter is used. See Algorithm section below for detailed discussion.

**No Selfcount** - option not to count reference events if the target event is the same as the reference event.

**Shift Predictor** - option to calculate the shift-predictor crosscorrelogram. Use the Shift-Predictor page of the Analysis Properties dialog to specify the shift-predictor options.

See [Using Shift-Predictor](#) for details.

See [Data Selection Options](#) for details on using the interval filter.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Draw 99% confidence limit** - option to draw two horizontal lines representing the confidence limits for the crosscorrelogram. See [Confidence Limits](#) for details.

**Draw mean firing rate** - option to draw a horizontal line representing the expected histogram value for the Poisson spike train. See [Confidence Limits](#) for details.

**Draw Cumulative Sum** - option to draw a cumulative sum graph above the histogram. See [Cumulative Sum Graphs](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Background, Peak/Trough Width, Left Shoulder, Right Shoulder** – parameters of the peak and trough analysis described below.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

## Algorithm

In general, the Crosscorrelogram shows the conditional probability of a spike in the spike train at time  $t$  on the condition that there is a reference event (or reference 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  $ref[i]$  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  $d[i] \geq XMin$  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  $d[i] \geq XMin+Bin$  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 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  $ts[k]$  above there are many  $d[i]$  values such that  $d[i] \geq XMin$  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 **NumRefEvents\*Bin**, where NumRefEvents is the number of reference events.

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 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:
  - o If **Background** parameter is set as Bins outside peak and trough, bins outside the peak and trough (i.e., bins that are more than  $\text{PeakWidth}/2$  away from the bin with the histogram maximum and the bin with the histogram minimum) are used to calculate **M** and **S**
  - o If **Background** parameter is set as Shoulders, bins that are to the left of 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  $(\text{HistogramMaximum} - \text{M})/\text{S}$  is shown as **Peak Z-score**

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

## 3.14. 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:

- click on the *New Events!* button (or 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, click on the Crosscorrelogram button, select Shift-predictor page and select *Trials* as an

interval filter and specify other shift-predictor parameters.

### 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.) (press "New Intervals!" button to create a new interval variable).

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.

## 3.15. Rasters

### Parameters

**XMin** - time axis minimum in seconds

**XMax** - time axis maximum in seconds.

**Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

### Algorithm

The raster display shows the spikes as vertical lines positioned according to the spikes' timestamps.

## 3.16. Perievent Rasters

### Parameters

**Reference Type** - a choice between single and multiple reference events.

**Reference** - reference neuron or event.

**XMin** - time axis minimum in seconds.

**XMax** - time axis maximum in seconds.

**Markers** - you can select several markers to be displayed in the raster.

**Sort Trials** - you can sort the trials in the perievent raster with respect to the latency between a reference event and some other event (**Sort ref.** below).

**Sort ref.** - event to be used to sort the trials.

**Histogram** - an option to include the perievent histogram in the graph below the raster.

**Bin** - histogram bin size in seconds. This parameter is also used in the display of the continuous variables (see **Continuous Variables** below).

**Normalization** - histogram units (**Counts/Bin, Probability** or **Spikes/Second**).

**Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

## Algorithm

For the perievent histogram calculation, see [Perievent Histograms](#).

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  $dist[k]$ .

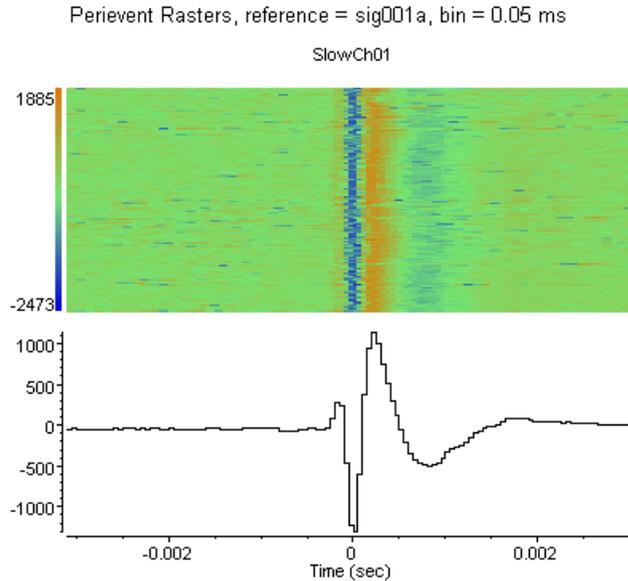
**Continuous variables** can also be used in this analysis.

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.



### 3.17. Joint PSTH

#### Parameters

**XMin** - time axis minimum in seconds.

**XMax** - time axis maximum in seconds.

**Bin** - bin size in seconds.

**Reference** - reference neuron or event.

**Bottom Neuron** - neuron of event shown along the horizontal axis.

**Main Diagonal Width** - the width of the area in the scatter matrix around its main diagonal used to calculate the main diagonal histogram.

**Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Font size** – the size of the font (in percent of the height of the main Joint PSTH matrix) to be used to draw labels in Joint PSTH graphs.

#### 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.

See the following paper for details on Joint PSTH analysis:

A. 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.

### 3.18. Cumulative Activity Graphs

#### Parameters

**XMin** - time axis minimum in seconds

**XMax** - time axis maximum in seconds.

#### Algorithm

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

### 3.19. Instant Frequency

#### Parameters

**XMin** - time axis minimum in seconds

**XMax** - time axis maximum in seconds.

#### Algorithm

The instant frequency display shows the instantaneous frequency of the spike train. For each spike that occurred at time  $t[i]$  it draws the 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.

### 3.20. Interspike Intervals vs. Time

#### Parameters

**XMin** - time axis minimum in seconds

**XMax** - time axis maximum in seconds.

#### 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.

### 3.21. Poincare Maps

#### Parameters

**Min Interval** - axis minimum in seconds

**Max Interval** - axis maximum in seconds.

#### 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 Y coordinate of the point is the preceding interspike interval.

### 3.22. Spike Distance vs. Time

#### Parameters

**XMin** - time axis minimum in seconds

**XMax** - time axis maximum in seconds.

**Reference** - reference neuron or event.

**Distance** - distance type (linear or inverted).

### 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  $(t[i], 1/dist)$ .

## 3.23. Trial Bin Counts

### Parameters

**Reference** - reference neuron or event.

**XMin** - time axis minimum in seconds.

**XMax** - time axis maximum in seconds.

**Bin** - bin size in seconds.

**Normalization** - numerical results units (**Counts/Bin**, or **Spikes/Second**).

**No Selfcount** - option not to count reference events if the target event is the same as the reference event.

**Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### 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.

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) Set all bin counts to zero:

```
bincount[i] = 0 for all i
```

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

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

3) 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**.

## 3.24. Power Spectral Densities

### Parameters

**Maximum Frequency** - maximum frequency for the spectrum.

**Number of Frequency Values** - number of values in the spectrum.

**Show Frequencies From/To** - options to show a subset of frequencies in the spectrum.

**Normalization** - option that specifies how the spectrum is normalized.

**Select Data, Select Data From, Select Data To, Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

## Algorithm

### *Spectral Densities for Spike Trains and Events*

NeuroExplorer uses rate histograms to estimate the power spectra of the spike trains. The parameters of rate histograms are calculated using the following formulas:

$$\text{Bin} = 1./(2.*\mathbf{Maximum\_Frequency})$$
$$\text{Number\_of\_Bins} = 2*\mathbf{Number\_of\_Frequency\_Values}$$

The time axis is divided into intervals of length  $\text{Bin}*\text{NumberOfBins}$ . Then the power spectrum for each interval is calculated. The final raw power spectrum is the average of all the spectra for the separate intervals.

For each interval:

- 1) The rate histogram is calculated.
- 2) The histogram is detrended (its linear regression is subtracted)
- 3) The Hann window

$$w[i] = (1 - \cos(2*\text{PI}*i/(\text{NumBer\_Of\_Bins}+1)))/2$$

is applied to the histogram.

- 4) FFT of the result is calculated.
- 5) Raw power spectrum is formed from the FFT.

### *Spectral Densities for Continuous Variables*

If **Maximum\_Frequency** = M and **Number\_of\_Frequency\_Values** = N, we need to calculate the spectrum for frequency values 0, M/N, 2\*M/N, ..., M.

For a continuous variable with digitizing frequency F, a standard power spectrum with K values in the FFT transform would produce the spectrum values for frequencies 0, F/K, 2\*F/K, ..., F/2.

If a continuous variable digitizing frequency F equals to **Maximum\_Frequency**\*2, we can set  $K = \mathbf{Number\_of\_Frequency\_Values}*2$  and calculate the standard power spectrum to get the spectrum values for the desired frequencies. That is, in the algorithm described above, instead of rate histogram we simply use continuous variable values in millivolts.

If a continuous variable digitizing frequency is not equal to **Maximum\_Frequency**\*2, we cannot directly calculate the spectrum for the specified frequencies. Instead, we find a value K

such that the frequency step for continuous variable spectrum is less than the specified step (i.e.  $F/K < M/N$ , so that we should get a more detailed spectrum than specified) and calculate the spectrum for this value of K. Then, we resample the resulting spectrum to get the spectrum values for the specified frequencies.

### ***Normalization***

If **Normalization** is Raw PSD, the power spectrum is normalized so that the sum of all the spectrum values equals to the mean squared value of the rate histogram.

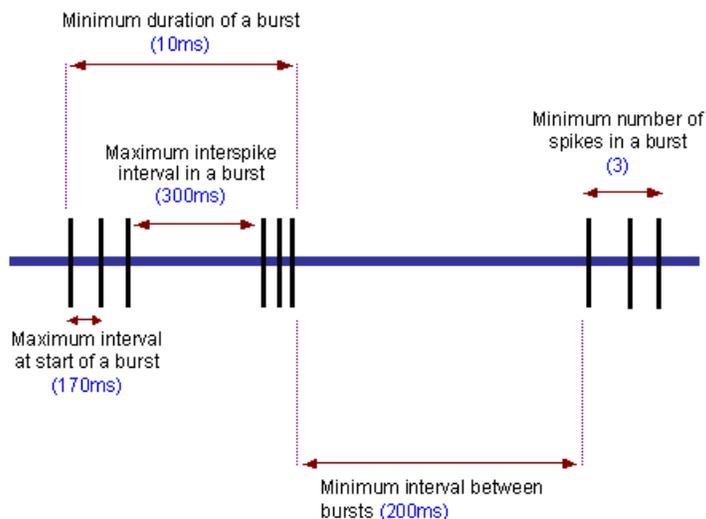
If **Normalization** is % of Total PSD, the power spectrum is normalized so that the sum of all the spectrum values equals 100.

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

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

## 3.25. Burst Analysis

### Parameters



**Algorithm** - MaxInterval of Surprise.

**Max. Interval** - maximum interspike interval to start the burst (MaxInterval method).

**Max. End Interval** - maximum interspike interval to end the burst (MaxInterval method).

**Min. Interval Between Bursts** - minimum interval between bursts (MaxInterval method).

**Min. Duration of Burst** - minimum burst duration (MaxInterval method).

**Min. Number of Spikes** - minimum number of spikes in the burst (MaxInterval method).

**Min. Surprise** - minimum surprise of the burst (Surprise method).

**Burst Rate Histogram Bin** - the value of the bin for the burst rate histogram.

**Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### Algorithm

For each spike train, NeuroExplorer creates a new Interval event and stores in this event all the burst intervals.

#### MaxInterval method

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

### Surprise method

- A.** First, the mean firing rate (**Freq**) and mean interspike interval (**MeanISI**) of the neuron are calculated.

$$\text{Freq} = \text{NumberOfSpikes} / (\text{FileEndTime} - \text{FileStartTime})$$

$$\text{MeanISI} = 1 / \text{Freq}$$

- B.** NeuroExplorer scans the spike train until it finds two sequential ISI's so that each of those ISI's is less than **MeanISI / 2**. 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**. If 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:

$$\text{S} = -\log_{10} (\text{Probability that } \mathbf{P} \text{ has at least } \mathbf{N} \text{ points in a time interval of length } \mathbf{T})$$

- C.** NeuroExplorer adds the spikes to the end of the burst until the first ISI that is more than **MeanISI**. The surprise is calculated for each of the bursts (with 3 initial spikes, 4 spikes, 5 spikes, etc.) The burst with maximum surprise **Smax** is then selected.
- D.** NeuroExplorer removes the spikes from the beginning of the burst and calculates the surprise for each of the shortened bursts. The burst with maximum surprise **Smax** is then selected.
- E.** If **Smax** is more than **MinSurprise** and the number of spikes in the burst is more than 3, NeuroExplorer adds the burst to the result.

### Reference

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

## 3.26. Principal Component Analysis

### Parameters

**Bin** - bin size in seconds.

**Vector Prefix** - a string specifying how the population vector names will be generated. For example, if prefix is **pca1**, the vector names would be **pca1\_01**, **pca1\_02**, etc.

**Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### Numerical Results

The **Results** sheet shows for each neuron the weights this neuron has in all the eigenvectors. The **Summary** sheet shows the eigenvectors (principal components), eigenvalues and the matrix of correlations between neurons ( $c[t, s]$ , see Algorithm below).

### Algorithm

Step 1.

Rate histograms are calculated for each of the selected neurons. The time axis is divided into bins. The first bin is **[0, Bin)**. The second bin is **[Bin, 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 >= 0 and ts < Bin
```

Bin counts are calculated in such a way for all the selected variables resulting in a matrix

```
bin_count[i, j],
```

where  $i$  is the neuron number,  $j$  is the bin number.

Step 2.

The matrix of correlations 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.

## 3.27. 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

**Reference** - reference neuron or event.

**XMinPSTH** - PSTH time axis minimum in seconds.

**XMaxPSTH** - PSTH time axis maximum in seconds.

**BinPSTH** - PSTH bin size in seconds.

**Start** - start of the first window.

**Duration** - duration of each sliding window.

**Shift** - how much sliding window is shifted each time.

**Number of Shifts** - total number of sliding windows.

**Normalization** - histogram units (**Counts/Bin**, **Probability** or **Spikes/Second**).

**No Selfcount** - option not to count reference events if the target event is the same as the reference event.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

### Algorithm

NeuroExplorer calculates here 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  
...
```

### 3.28. 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

**Reference** - reference continuous variable.

**XMin** - time axis minimum in seconds.

**XMax** - time axis maximum in seconds.

**Bin Type** - if Bin Type is set to auto, NeuroExplorer will use the bin equal to the A/D sampling interval of the reference variable.

**Bin** - bin size in seconds.

#### 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  $x[i]$ ,  $i=1, \dots, N$  is the reference variable and  $y[i]$ ,  $i=1, \dots, N$  is another continuous variable or spike train rate histogram, then

$c[t]$  = Pearson correlation between vectors  $\{ x[1], x[2], \dots, x[N-t] \}$  and  $\{ y[t+1], y[t+2], \dots, y[N] \}$ .

## 3.29. Regularity Analysis

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

### Parameters

**Reference** - reference event or spike train.

**XMin** - time axis minimum in seconds.

**XMax** - time axis maximum in seconds.

**Bin** - bin size in seconds.

**ISI Graph** - this parameter determines how the mean ISI values will be displayed in the graph.

**SD Graph** - this parameter determines how the standard deviations of ISI will be displayed in the graph.

**CV Graph** - this parameter determines how the CV (coefficient of variation) values will be 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).

### 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)

1) 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 (  $d[i] \geq XMin$  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 (  $d[i] \geq X_{Min} + Bin$  and  $d[i] < X_{Min} + Bin * 2$  )  
and  $d[i] + isi[i] < X_{Max}$ ,  
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.

## 3.30. 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

**X Position** - the variable that describes X position of the animal.

**Y Position** - the variable that describes X position of the animal.

**Fix Positions** - this option specifies whether to 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.

**Fix Threshold** - fix option parameter. See **Algorithm** section below for details.

**XMin** - X axis minimum in raw x1\_pos units.

**XMax** - X axis maximum in raw x1\_pos units.

**YMin** - Y axis minimum in raw y1\_pos units.

**YMax** - Y axis maximum in raw y1\_pos 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)).

**Min Time Spent, Min Visits** - these two parameters are used only 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.

**Smooth Matrix** - option to smooth the matrix after the calculation.

**Smooth Radius** - radius of the smoothing filter (in cells).

## 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:  
$$\text{aver} = (x[t-2]+x[t-1]+x[t+1]+x[t+2])/4$$
- if  $\text{abs}(x[t] - \text{aver}) > \text{FixThreshold}$ , then assign  $x[t]$  the average of its neighbors:  
$$x[t] = \text{aver}$$

The following steps are then performed:

1. The position space is divided into cells with width  $(X_{\text{Max}} - X_{\text{Min}})/\text{Number of Bins (X)}$  and height  $(Y_{\text{Max}} - Y_{\text{Min}})/\text{Number of Bins (Y)}$ .
2. For each cell, the number of visits to this cell and the time spent in the cell are calculated.
3. 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.
4. For each cell, the number of times the neuron fired in this cell is calculated.
5. 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.

## 3.31. 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

**Reference** - the variable that contains timestamps of the stimuli.

**Stim. Sequence File** - path of the text file that contains the sequence of images used for stimulation. See **Algorithm** below for file format description.

**Char. per Pixel** - number of characters per pixel in image file.

**Pixels in Row** - number of pixels in a row for each stimulus image.

**Rows in Image** - number of rows of pixels in each stimulus image.

The following 4 parameters specify the axes to be used in Reverse Correlation display.  
**XMin (degrees), XMax (degrees)** - minimum and maximum for X axis.  
**YMin (degrees) , YMax (degrees)** - minimum and maximum for Y axis.

The following 2 parameters specify the time range (before the spike) to be used in Reverse Correlation display.

**Time Min (sec)** - time minimum.

**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** - the option specifies the slice that will be shown. See **Algorithm** below.

**Z Min** and **Z Max** specify the limits to be used for color scale.

## 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 (16 pixels per row, 12 rows per image):

```

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 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 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

```

The computational algorithm works in 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). 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.

## 3.32. Epoch Counts

This analysis is very similar to Perievent Histograms. The only distinction is that Epochs analysis can calculate bin counts for bins (epochs) of any size. Epochs can be of different lengths and can overlap.

### Parameters

**Reference Type** - a choice between single and multiple reference events.

**Reference** - reference neuron or event.

**Epochs** - a table of epochs in seconds.

**Select Data, Select Data From, Select Data To, Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

### 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  $d[i] \geq EpochStart[1]$  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  $d[i] \geq EpochStart[2]$  and  $d[i] < EpochEnd[2]$ 
then  $epochcount[2] = epochcount[2] + 1$ 
```

and so on... .

### 3.33. Coherence Analysis

#### Parameters

**Reference** - reference neuron, event or continuous variable.

**Maximum Frequency** - maximum frequency for the coherence analysis.

**Number of Frequency Values** - number of values in the FFT spectrum used for coherence analysis.

**Show Frequencies From/To** - options to show a subset of frequencies in the spectrum.

**Select Data, Select Data From, Select Data To, Interval Filter** - select spikes from time intervals. See [Data Selection Options](#) for details.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

#### Algorithm

NeuroExplorer uses rate histograms to estimate the power spectra of the spike trains. The parameters of rate histograms are calculated using the following formulas:

$$\text{Bin} = 1./(2.*\text{Maximum\_Frequency})$$
$$\text{Number\_of\_Bins} = 2*\text{Number\_of\_Frequency\_Values}$$

The time axis is divided into intervals of length  $\text{Bin}*\text{NumberOfBins}$ . Then the power spectrum for each interval is calculated as described in Power Spectral Densities analysis reference.

For two variables X and Y, NeuroExplorer averages the squares of the spectra of the X to form Pxx, averages the squares of the spectra of the Y sections to form Pyy, and averages the products of the spectra of the X and Y to form Pxy. It calculates coherence Cxy by the following formula.

$$C_{xy} = (P_{xy}*P_{xy})/(P_{xx}.*P_{yy})$$

### 3.34. Spectrogram Analysis

#### Parameters

**Maximum Frequency** - maximum frequency for the spectrograms. If at least one continuous variable is selected, the value of Maximum Frequency parameter is set as  $0.5*\text{Digitizing}$

frequency of the first selected continuous variable. All continuous variables selected for this analysis should have the same digitizing rate.

**Number of Frequency Values** - number of frequency values in spectrograms.

**Normalization** – normalization of the spectra (**Raw PSD (Power Spectral Density), Log of PSD**).

**Start** - start of the first window.

**Shift** - how much sliding window is shifted each time.

**Number of Shifts** - total number of sliding windows.

**Smooth, Smooth Filter Width** - options to smooth the histogram after calculation. See [Post-Processing Options](#) for details.

**Add to Results** - options to add additional vectors to the matrix of numerical results.

**Send to Matlab, Matrix Name, Matlab Command** - Matlab-related options. See [Matlab Options](#) for details.

**Send to Excel, Sheet Name, Topleft Cell** - Excel-related options. See [Excel Options](#) for details.

## 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 in to the Signal array. The following parameters are used:

Histogram\_Start = **Start + Shift** \* (Window\_Number – 1)  
Bin = 1./(2.\***Maximum\_Frequency**)  
Number\_of\_Bins = 2\***Number\_of\_Frequency\_Values**

2. For a continuous variable, the values of the continuous variable are copied to Signal array.
3. Signal values are multiplied by the coefficients of Hann window:

$Hann[j] = 0.5 * (1 - \cos(2 * \pi * j / \text{Number\_of\_Bins}))$

4. Discrete FFT of the result is calculated.
5. 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, the power spectrum is normalized so that the sum of all the spectrum values equals to the mean squared value of the Signal.

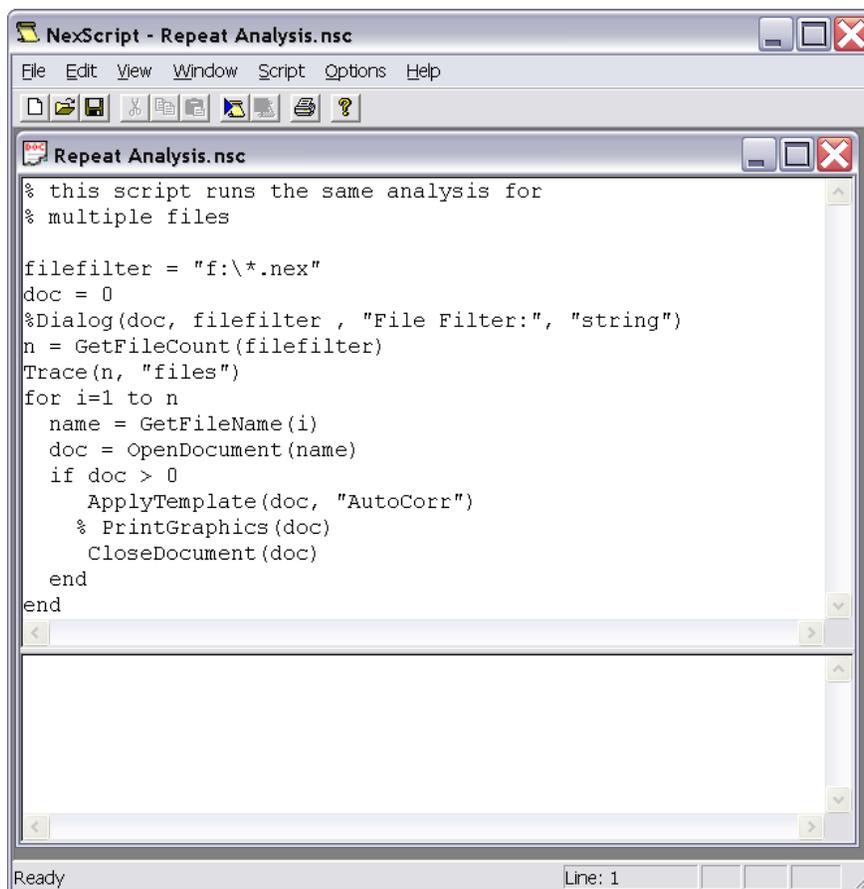
If **Normalization** is Log of PSD, the power spectrum is calculated using the formula:

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

## 4. Programming with NexScript

### 4.1. Introduction to NexScript Programming

NeuroExplorer has a powerful scripting language (NexScript) resembling the Matlab language. To create a script, select the **Script | New Script** menu command. To open existing script, double-click at the script name in the Scripts View. NeuroExplorer will open NexScript editor:



Each window in NexScript editor has two parts. The top part contains the text of the script. The bottom part displays compiler messages and debug output (using **Trace** function)

#### Scripts

Each script is a text file that has an extension .nsc. Scripts are usually saved in the directory <NexHome>\Scripts.

Note that in NeuroExplorer version 3 you can create subfolders in the scripts folder. The scripts tree shown in the Scripts View is a copy of the directory <NeuroExplorer Home>\Scripts, where <NeuroExplorer Home> is your NeuroExplorer installation directory (usually, C:\Program Files\Nex Technologies\Nex). You can create subfolders in your scripts directory and then NeuroExplorer will allow you to navigate through <NeuroExplorer Home>\Scripts tree within the Scripts View.

## Lines and Comments

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:\Plexon\data\mydata\*.plx", "Filter", "string")
```

you can write:

```
Dialog(doc, "D:\Plexon\data\mydata\*.plx",\  
        "Filter", "string")
```

The percent symbol (`%`) marks the beginning of a comment, for example:

```
% this is a comment line  
x = 0 % this also a comment
```

## Variables and Expressions

NexScript supports numeric variables and standard expressions:

```
xmean = (xmax - xmin)/2.
```

strings:

```
name = "DSP" + "01"
```

and timestamped events:

```
doc.DSP01a[3] = 0.001
```

See [Variables](#) and [Expressions](#) to learn more about the basics of NexScript.

## Flow Control

`for` loops, `while` loops and `if ... else` constructs can be used for flow control:

```
imax = 0  
doc = GetActiveDocument()  
for i = 2 to n  
    interval = doc.DSP01a[i] - doc.DSP01a[i-1]  
    if interval > imax  
        imax = interval  
    end  
end
```

See [Flow Control](#) for more information.

## Functions

NexScript offers more than 100 functions that allow you to edit data, open and close files, perform analyses and print the results. See [Functions](#) for more information.

## 4.2. Variables

### Variable Names

The variable name in NeuroExplorer should begin with a letter and contain only letters, digits and the underscore sign.

The following names are valid:

```
Neuron01   Bar_press   Nr_1a   DSP02b
```

These names cannot be used in NeuroExplorer:

```
2Neuron   Bar-press
```

The variables from the opened data file should have the prefix specifying the document. For example, the spike train DSP01a from the active document should be addressed as:

```
doc.DSP01a
```

where `doc` is a reference to the document:

```
doc = GetActiveDocument()
```

### Variable Types

NexScript supports the numeric variables:

```
xmean = (xmax - xmin)/2.
```

strings:

```
name = "DSP" + "01"
```

and timestamped events:

```
doc.DSP01a[3] = 0.001
```

A variable may be also a reference to the existing variable in the file:

```
neuron1 = GetVar(doc, 1, "neuron")
```

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 = "DSP" % 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.

## Modifying Timestamped Variables

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 DSP01a, you simply write:

```
doc = GetActiveDocument()
doc.DSP01a[3] = 0.5
```

## Modifying Interval Variables

`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, 2)

doc.MyInterval[1,1] = 0.
doc.MyInterval[1,2] = 100.

doc.MyInterval[2,1] = 200.
doc.MyInterval[2,2] = 300.
```

## 4.3. Expressions

Standard algebraic expressions are supported:

```
xmean = (xmax - xmin)/2.
```

Addition operation can also be applied to the strings:

```
name = "DSP" + "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
```

## 4.4. Flow Control

### 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
```

## 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

if GetVarCount(doc, i, "neuron") > 0
  SelectVar(doc, i, "neuron")
else
  DeselectVar(doc, i, "neuron")
end
```

## 4.5. Functions

### 4.5.1. Function categories

The following function types are available in NexScript:

- [Math Functions](#)
- [String Functions](#)
- [File Access Functions](#)
- [Access to the File Variables](#)
- [Modifying Existing Variables](#)
- [Creating and Deleting Variables](#)
- [Analysis Functions](#)
- [User Interface](#)
- [Debugging Functions](#)

### 4.5.2. Math Functions

**seed(number)** - seeds random number generator, returns nothing.

All other functions in this group return a number.

**rand()** - random number uniformly distributed between 0 and 1.

**expo(mean)** - exponentially distributed random number with specified mean.

**sqrt(x)** - square root function

**log(x), exp(x)** - logarithm and exponential functions

**pow(x, y)** - calculates x raised to the power of y

**sin(x), cos(x)** - standard trigonometric functions

**abs(x)** - calculates the absolute value

**min(x,y)** - calculates minimum of two numbers

**max(x,y)** - calculates maximum of two numbers

**floor(x)** - returns a number representing the largest integer that is less than or equal to x

**ceil(x)** - returns a number representing the smallest integer that is greater than or equal to x.

**round(x)** - returns a number representing the integer that is closest x.

**BitwiseAnd(x, y)** – converts x and y to unsigned integers and performs bitwise AND operation on these integers. Returns the result of bitwise AND operation.

**BitwiseOr(x, y)** – converts x and y to unsigned integers and performs bitwise OR operation on these integers. Returns the result of bitwise OR operation.

**GetBit(x, n)** – converts x to unsigned integer and returns the value of the n-th bit of this integer. n is 1-based -- the least significant bit is 1, the most significant bit is 32.

### 4.5.3. String Functions

**Left(string, nchar)** - extracts left **nchar** characters from **string**, returns string.

```
name = "DSP01a"  
prefix = Left(name, 3) % prefix is "DSP"
```

**Mid(string, nstartchar, nchar)** - extracts **nchar** characters from **string** starting with character number **nstartchar**, returns string.

```
name = "DSP01a"  
chanstring = Mid(name, 4, 2) % chanstring is "01"
```

**Right(string, nchar)** - extracts right **nchar** characters from **string**, returns string.

```
name = "DSP01a"  
unitstring = Right(name, 1) % unitstring is "a"
```

**Find(string1, string2)** - looks for a string **string2** inside the string **string1**, returns a number - the position of the first character of **string2** in the **string1**. Returns zero if string **string2** is not found.

```
% select only units from channel 5  
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
```

**StrToNum(string)** - converts **string** to number, returns number.

```
chanstring = "01"  
channelnumber = StrToNum(chanstring)
```

**NumToStr(number, formatstring)** - converts **number** to string using optional **formatstring**, returns string. **formatstring** is a standard C-style format specifier.

For example, to generate strings:

```
Event001, Event002, ..., Event016
```

use the following loop

```
for i=1 to 16  
    str = "Event0" + NumToStr(i, "%02.0f")  
end
```

**NumToChar(number)** - converts **number** to a one-character string containing the character with the ASCII code equal to **number**. The number should be an ASCII code for a printable symbol. For example, NumToChar(69) is "D".

**CharToNum(string)** - converts a one-character string to a number (a character's ASCII code), returns string.

**StrLength(string)** - calculates the number of characters in the **string**, returns number.

**GetNumFields(string)** - returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas. For example,

```
GetNumFields("One two 3 4") returns 4.
```

**GetField(string, fieldnumber)** - returns the field with the specified number. For example,

```
GetField("One two 3 4", 2) returns "two".
```

## 4.5.4. File Access Functions

### 4.5.4.1. File Access Functions

The following categories of file access functions are available:

- [Opening and Saving Data Files](#)
- [Opening Multiple Data Files](#)
- [Saving Numerical Results](#)
- [Accessing Document Parameters](#)
- [Reading and Writing Text Files in Script](#)

### 4.5.4.2. Opening and Saving Data Files

**OpenDocument(name)** - opens the file with specified file name, returns reference to the opened document. The function returns zero if OpenDocument operation failed.

```
doc = OpenDocument("c:\data\file1.nex")
```

This function can also be used to open all supported data file formats. For example:

```
doc = OpenDocument("c:\data\file1.plx")
doc = OpenDocument("c:\data\file1.map")
doc = OpenDocument("c:\data\file1.mcd")
doc = OpenDocument("c:\data\file1.txt")
```

Before opening a text file, specify text file options using [View/Options](#) menu command.

**NewDocument(freq)** - creates a new document with the specified timestamp frequency **freq** (in Hertz) returns reference to the opened document.

```
doc = NewDocument(25000.)
```

**SaveDocument(doc)** - saves the specified document in NeuroExplorer format. If the file name extension of the document is not ".nex", the file name extension is replaced with ".nex". For example, if you open the file "Test.txt", then the command SaveDocument(doc) will save the document in the file "Test.nex".

**SaveDocumentAs(doc, fileName)** - saves the document in the file with the specified file name.

**SaveAsTextFile(doc, fileName)** - saves the document in the text file with the specified file name. This function uses options that were specified last time the menu command **File | Export Data | As Text File** was executed.

**CloseDocument(doc)** - closes the document **doc**.

**MergeFiles(name\_list, gap)** - merges the specified files, returns the reference to the merged file. **name\_list** is the list of files separated by "+", for example:

```
doc = MergeFiles("c:\data\file1.plx+c:\data\file2.plx+c:\data\file3.plx", 1.)
```

**gap** is the time interval between the files.

#### 4.5.4.3. Opening Multiple Data Files

**GetFileCount(filefilter)** - returns the number of files found for the given filter.

```
n = GetFileCount("c:\data\*.nex")
```

**GetFileName(number)** - returns the file name after the **GetFileCount(...)** was called.

Here is a script that opens all \*.nex files in the directory and analyses them:

```
filefilter = "c:\data\*.nex"
n = GetFileCount(filefilter)
for i=1 to n
    name = GetFileName(i)
    doc = OpenDocument(name)
    if doc > 0
        % run the analysis, print results and close the file
        ApplyTemplate(doc, "Interspike Interval Histograms")
        PrintGraphics(doc)
        CloseDocument(doc)
    end
end
```

#### 4.5.4.4. Saving Numerical Results

**SaveNumResults(doc, filename)** - saves the numerical results to a text file with the specified name.

**SaveNumSummary(doc, filename)** - saves the summary of numerical results to a text file with

the specified name.

#### 4.5.4.5. Accessing Document Parameters

**GetTimestampFrequency(doc)** - returns the frequency used in the specified file to time the timestamps.

**GetDocEndTime(doc)** - returns the maximum timestamp value (in seconds) for all the document variables.

**SetDocEndTime(doc, endtime)** - sets the length of experimental session (in seconds) for the document.

**GetDocTitle(doc)** - returns the title of the document. For example, if the document has the path "C:\Data\data1.nex", this function will return "data1.nex".

**GetDocPath(doc)** - returns the full path of the document.

**GetDocComment(doc)** - returns the document comment string.

#### 4.5.4.6. Reading and Writing Text Files in NexScript

**OpenFile(name, mode)** - opens the text file that can be used by the script (*not the data file!*), returns the reference to the opened file.

**mode** should be "rt" to open the file in read text mode, or "wt" to open file in write text mode. If the returned value is zero, OpenFile failed.

Use this function to read text files directly in NexScript or to write text to a file directly from the script.

```
% the following command opens a file in a text read mode
fileid = OpenFile("C:\out.txt", "rt")
```

**CloseFile(fileid)** - closes the file **fileid**.

**ReadLine(fileid, linestr)** - reads a line of text from the specified file and puts the line into **linestr** string variable. Returns 1 if succeeded or 0 if failed.

```
linestr = " " % make linestr a string variable
fileid = OpenFile("C:\parameters.txt", "rt")
% read all the lines in the file
if fileid > 0
    while ReadLine(fileid, linestr) > 0
        Trace(fileid, linestr)
    end
    CloseFile(file)
end
```

**WriteLine(fileid, linestr)** - writes a line of text to the specified file. The file should be opened with "wt" mode.

The following string functions can be used to extract the string fields from the text files:

**GetNumFields(string)** - returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas. For example,

```
GetNumFields("One two 3 4") returns 4.
```

**GetField(string, fieldnumber)** - returns the field with the specified number. For example,

```
GetField("One two 3 4", 2) returns "two".
```

## 4.5.5. Access to the File Variables

### Direct Access to the Variables

Since NeuroExplorer can open several documents (data files), before accessing the variables in the file, you need to obtain a reference to an active document:

```
doc = GetActiveDocument()
```

The variables from the opened data file should have the prefix specifying the document. For example, the spike train DSP01a from the active document should be addressed as:

```
doc.DSP01a
```

### Spike Trains and Events

The timestamps can be accessed using brackets []. For example, to print the value of the 3rd timestamp of DSP01a, you can use the following line:

```
Trace(doc.DSP01a[3])
```

### Interval Variables

**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. See [Modifying Existing Variables](#) for details.

### Marker Variables

**Marker[i,1]** gives you the access to the timestamp of the i-th marker, **Marker[i,2]** returns the string representation of the first marker value (the DIO value in Plexon files).

For example, the following script will extract the timestamps corresponding to the DIO values between 0 and 10 and will create the new event with these timestamps:

```
doc = GetActiveDocument()
n = GetSpikeCount(doc.Strobed)
doc.n10 = NewEvent(doc, 0)
for i=1 to n
    timestamp = doc.Strobed[i, 1]
    dio_string = doc.Strobed[i, 2]
```

```

        dio_value = StrToNum(dio_string)
        Trace(i, timestamp, dio_string, dio_value)
        if (dio_value >= 0) & (dio_value <= 10)
            AddTimestamp(doc.n10, timestamp)
        end
    end
end

```

Here `doc.Strobed[i, 2]` returns a string corresponding to the i-th DIO value. To get the numerical DIO value, you need to use the function `NumToStr()`.

### Continuous Variables

`Cont[i, 1]` returns the timestamp of the i-th value of the continuous variable (in seconds), `Cont[i, 2]` returns the i-th value of the continuous variable in millivolts.

For example, the following script will print the timestamps and values of the first continuous variable:

```

doc = GetActiveDocument()
contvar = GetVar(doc, 1, "continuous")
n = GetContNumDataPoints(contvar)
for i=1 to n
    timestamp = contvar[i, 1]
    value = contvar[i, 2]
    Trace(i, timestamp, value)
end

```

### Accessing Arrays of Variables

NeuroExplorer maintains several arrays of variables. The following functions may be used to access the elements of those arrays.

**GetVarCount(doc, string)** - returns the number of variables of the specified type in the file. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "popvector", "continuous" or "all".

**GetVarName(doc, number, string)** - returns a string -- the name of the variable of the specified type in the file. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "popvector", "continuous" or "all".

**GetName(var)** - returns the name of the variable.

**GetVarSpikeCount(doc, number, string)** - returns the number of timestamps in the variable. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "continuous" or "all".

**GetSpikeCount(var)** - returns the number of timestamps in the variable.

**GetContNumDataPoints(var)** - returns the number of data points in the continuous variable.

**GetVar(doc, number, string)** - returns the reference to the specified variable. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "popvector", "continuous" or "all".

**GetVarByName(doc, strname)** - returns the reference to the variable which has the name **strname**, or zero, if the document **doc** does not have a variable with such a name.

**SetNeuronType(doc, var, type)** - changes the type of a variable **var**. If **type** > 0, the new type is "neuron", if **type** <= 0, the new type is "event".

## Example

The following example script selects all neurons that have timestamps:

```
doc = GetActiveDocument()
n = GetVarCount(doc, "neuron")
DeselectAll()
for i = 1 to n
    k = GetVarSpikeCount(doc, i, "neuron")
    if k > 0
        SelectVar(doc, i, "neuron")
    end
end
```

## 4.5.6. Selecting Variables

**SelectVar(doc, number, string)** - selects the specified variable for analysis. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "popvector", "continuous" or "all".

**DeselectVar(doc, number, string)** - deselects the specified variable. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "popvector", "continuous" or "all".

**Select(doc, var)** - selects the specified variable for analysis. For example:

```
doc = GetActiveDocument()
Select(doc, doc.DSP01a)
```

**IsSelected(var)** - returns 1 if the variable **var** is selected, 0 otherwise..

**Deselect(doc, var)** - deselects the specified variable.

**SelectAll(doc)** - selects all variables for analysis.

**DeselectAll(doc)** - deselects all variables.

**SelectAllNeurons(doc)** - selects all neuron type variables for analysis.

**SelectAllEvents(doc)** - selects all event type variables for analysis.

**DisableRecalcOnSelChange ()** – disables recalculation of analyses when the list of selected variables changes.

**EnableRecalcOnSelChange ()** – enables recalculation of analyses when the list of selected variables changes.

**RecalculateAnalysisInWindow(doc, graph\_window\_number)** – forces recalculation of analysis in the specified graph window.

## Example

The following example script selects all neurons that have timestamps:

```
doc = GetActiveDocument()
n = GetVarCount(doc, "neuron")

DeselectAll(doc)

for i = 1 to n
    k = GetVarSpikeCount(doc, i, "neuron")
    if k > 0
        SelectVar(doc, i, "neuron")
    end
end
```

## 4.5.7. Modifying Existing Variables

### Timestamped Variables

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 dsp01a, you simply write:

```
doc = GetActiveDocument()
doc.DSP01a[3] = 0.5
```

You can also add timestamps to a variable using AddTimestamp function.

**AddTimestamp(var, timestamp)** - adds a new timestamp to the variable **var**. The **timestamp** value should be specified in seconds.

For example, if **DSP01a** has 100 timestamps, after the following script line

```
AddTimestamp(doc.DSP01a, 500.)
```

**DSP01a** will have 101 timestamps with the last timestamp being equal to 500.

### Interval Variables

**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.

**AddInterval(var, interval\_start, interval\_end)** – adds a new interval to the specified interval variable **var**. **interval\_start** and **interval\_end** should be specified in seconds.

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, 2)

doc.MyInterval[1,1] = 0.
doc.MyInterval[1,2] = 100.

doc.MyInterval[2,1] = 200.
doc.MyInterval[2,2] = 300.
```

An alternative way is to use **AddInterval** function:

```
doc = GetActiveDocument()
doc.MyInterval = NewIntEvent(doc, 0)

AddInterval(doc.MyInterval, 0., 100.)
AddInterval(doc.MyInterval, 200., 300.)
```

## 4.5.8. Creating and Deleting Variables

**NewEvent(doc, count)** - creates a new timestamped variable, returns a reference to the new variable. **count** specifies the initial number of timestamps.

### Example

```
doc = GetActiveDocument()
temp = NewEvent(doc, 10) % creates a script-only variable
doc.NewVar = NewEvent(doc, 0) % creates a new variable in the file
```

**NewIntEvent(doc, count)** - creates a new interval variable, returns a reference to the new variable. **count** specifies the initial number of intervals.

### 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, 2)

doc.MyInterval[1,1] = 0.
doc.MyInterval[1,2] = 100.

doc.MyInterval[2,1] = 200.
doc.MyInterval[2,2] = 300.
```

An alternative way is to use **AddInterval** function:

```
doc = GetActiveDocument()
doc.MyInterval = NewIntEvent(doc, 0)

AddInterval(doc.MyInterval, 0., 100.)
AddInterval(doc.MyInterval, 200., 300.)
```

**NewPopVector(doc, type)** - creates a new population vector, returns a reference to the new vector.

- 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.

You can also use any of the functions available in the **New Variables** dialog (**Edit | Operations on Variables** menu command). See help window in the New Variables dialog for detailed explanations on what each function is doing.

Additional functions that perform operations on data variables are listed below.

**MarkerExtract(doc, MarkerVariableName, ExtractString)** – this function creates a new event variable based on existing marker variable. **doc** is a reference to the document; **MarkerVariableName** is a string with the name of a marker variable, **ExtractString** is a string that specifies what timestamps should be included in the new event.

**ExtractString** contains a list of items separated by commas. Items AND or OR should be placed between the conditions for the sample 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")
```

**NthAfter(var1, var2, N)** – this function creates a new event using the following algorithm:  
 If variable **var1** has timestamps  $t1[i]$  and variable **var2** has timestamps  $t2[j]$ ,  
 - for each timestamp  $t2[j]$ , find **Nth** timestamp  $t1[j]$ , such that  $t1[j] > t2[j]$  and  $t1[j] < t2[j+1]$ .

**DeleteVar(doc, number, string)** - deletes the specified variable. **string** should be one of: "neuron", "neuronorevent", "event", "interval", "wave", "popvector", "continuous" or "all". All analysis windows are closed.

**Delete(doc, var)** - deletes the specified variable from the file. All analysis windows are closed.

## 4.5.9. Analysis Functions

**RunScript(scriptname)** - runs the script with the specified name.

**ApplyTemplate(doc, templatename)** - applies the template to the specified file.

**ApplyTemplateToWindow(doc, templatename, windownumber)** - applies the template to the specified file and draws the result in the specified Graph window. 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.

**ModifyTemplate(doc, templatename, paramname, newparvalue)** - modify one of the template parameters. **templatename, paramname, newparvalue** are strings.

For example, to set the new bin value in the *Rate Histograms* template, you need to write:

```
ModifyTemplate(doc, "Rate Histograms", "Bin (sec)", "5.0")
```

Note that parameter name should be specified **exactly** as it is shown in the left column of the *Parameters View* (e.g. not "**Bin**", but "**Bin (sec)**" as in the example above). You can select the parameter name in the left column of Parameters View and press Ctrl+C to copy the parameter name to the clipboard. You can then paste the name of the parameter into your script.

ModifyTemplate can be used to specify multiple references in Perievent Histograms, Crosscorrelograms and Perievent Rasters:

```
doc = GetActiveDocument()
ModifyTemplate(doc, "Peri", "Ref. type", "Table (row)")
ModifyTemplate(doc, "Peri", "Reference", "Event04+Event05+Event06")
ApplyTemplate(doc, "Peri")
```

You can also use "+" to specify multiple interval filters in Perievent Histograms, Crosscorrelograms and Perievent Rasters.

ModifyTemplate can be used to specify graphics parameters (NeuroExplorer 2.41 and later versions). To change the Graph parameter, you need to add **Graph|** before the parameter name:

```
doc = GetActiveDocument()
ModifyTemplate(doc, "Peri", "Graph|Graph Style", "Histogram")
ModifyTemplate(doc, "Peri", "Graph|Line color", "1")
ModifyTemplate(doc, "Peri", "Graph|Fill under line", "0")
```

To change the Y Axis parameter, you need to add **YAxis|** before the parameter name:

```
ModifyTemplate(doc, "Peri", "YAxis|Max Type", "Fixed")
ModifyTemplate(doc, "Peri", "YAxis|Fixed Max", "50.")
```

To change the X Axis parameter, you need to add **XAxis|** before the parameter name.

**SaveNumResults(doc, filename)** - saves the numerical results to a text file with the specified name.

**SaveNumSummary(doc, filename)** - saves the summary of numerical results to a text file with the specified name.

**PrintGraphics(doc)** - prints the contents of the first graphical window of the document.

**GetNumResNRows(doc)** - returns the number of rows of the Numerical Results Window of the

first graphical view of the document.

**GetNumResNCols(doc)** - returns the number of columns of the Numerical Results Window of the first graphical view of the document.

**GetNumRes(doc, row, col)** - returns the value of the cell (**row, col**) in the Numerical Results Window of the first graphical view of the document.

**GetNumResColumnName(doc, col)** - returns the name of the column of the Numerical Results Window of the first graphical view of the document.

**DisableRecalcOnSelChange()** – disables recalculation of analyses when the list of selected variables changes.

**EnableRecalcOnSelChange()** – enables recalculation of analyses when the list of selected variables changes.

**RecalculateAnalysisInWindow(doc, graph\_window\_number)** – forces recalculation of analysis in the specified graph window.

**GetNumResSummaryNRows(doc)** - returns the number of rows of the Numerical Results Summary Window of the first graphical view of the document.

**GetNumResSummaryNCols(doc)** - returns the number of columns of the Numerical Results Summary Window of the first graphical view of the document.

**GetNumResSummaryColumnName(doc, col)** - returns the name of the column of the Numerical Results Summary Window of the first graphical view of the document.

**GetNumResSummaryData(doc, row, col)** - returns the value of the cell (**row, col**) in the Numerical Results Summary Window of the first graphical view of the document.

**SendGraphicsToPowerPoint (doc, presentationPath, slideTitle, comment, addParameters)** - sends the contents of the first graphical window of the document to the specified Power Point presentation. **presentationPath, slideTitle** and **comment** parameters are strings, **addParameters** is a numeric parameter (0 or 1).

**SendResultsToExcel(doc, fileName, worksheetName, useFirstEmptyRow, cellName, includeHeader, includeFileName)** – sends numerical results (of the first graphics window of the document) to Excel. **doc** - reference to the document, **fileName** - string with Excel file name, **worksheetName** - string with the name of the worksheet, **useFirstEmptyRow** - number (0 or 1) (if 1, NeuroExplorer will ignore cellName parameter and add the results to the first row where the cell in column A is empty; if 0, NeuroExplorer will paste data starting with the cell specified in **cellName**); **includeHeader** - (0 or 1) option whether to send the column names to Excel.

**SendResultsSummaryToExcel (doc, fileName, worksheetName, useFirstEmptyRow, cellName, includeHeader, includeFileName)** – sends summary of numerical results (of the first graphics window of the document) to Excel. **doc** - reference to the document, **fileName** - string with Excel file name, **worksheetName** - string with the name of the worksheet, **useFirstEmptyRow** - number (0 or 1) (if 1, NeuroExplorer will ignore cellName parameter and add the results to the first row where the cell in column A is empty; if 0, NeuroExplorer will paste data starting with the cell specified in **cellName**); **includeHeader** - (0 or 1) option whether to send the column names to Excel.

## 4.5.10. Matlab Functions

**SendSelectedVarsToMatlab(doc)** - sends the selected variables (as vectors of timestamps in seconds) to Matlab.

**ExecuteMatlabCommand(command)** - sends the string **command** to Matlab and executes the command in Matlab.

**GetVarFromMatlab(doc, varname, isneuron)** - gets the specified variable from Matlab. If **isneuron** is 1, the variable is added to the list of Neurons. Otherwise, the variable is added to the list of events. The variable in Matlab should be a double matrix with either one row or one column of data containing timestamps in seconds.

**GetContVarFromMatlab(doc, MatrixName, Timestamp, TimeStep)** – imports the specified matrix from Matlab. Each column of the matrix is imported as a continuous variable. **Timestamp** specifies the time of the first value of each continuous variable. **TimeStep** identifies digitizing time step of the imported variables.

## 4.5.11. Excel Functions

**SetExcelCell(sheet, cell, text)** - sets the text value of the specified cell in Excel. **sheet** is the worksheet name, **cell** is the cell address (for example, A1), **text** is the cell value.

**SendResultsToExcel(doc, fileName, worksheetName, useFirstEmptyRow, cellName, includeHeader, includeFileName)** – sends numerical results (of the first graphics window of the document) to Excel. **doc** - reference to the document, **fileName** - string with Excel file name, **worksheetName** - string with the name of the worksheet, **useFirstEmptyRow** - number (0 or 1) (if 1, NeuroExplorer will ignore **cellName** parameter and add the results to the first row where the cell in column A is empty; if 0, NeuroExplorer will paste data starting with the cell specified in **cellName**); **includeHeader** - (0 or 1) option whether to send the column names to Excel.

**SendResultsSummaryToExcel (doc, fileName, worksheetName, useFirstEmptyRow, cellName, includeHeader, includeFileName)** – sends summary of numerical results (of the first graphics window of the document) to Excel. **doc** - reference to the document, **fileName** - string with Excel file name, **worksheetName** - string with the name of the worksheet, **useFirstEmptyRow** - number (0 or 1) (if 1, NeuroExplorer will ignore **cellName** parameter and add the results to the first row where the cell in column A is empty; if 0, NeuroExplorer will paste data starting with the cell specified in **cellName**); **includeHeader** - (0 or 1) option whether to send the column names to Excel.

## 4.5.12. User Interface Functions

**Dialog(doc, par1, name1, type1, par2, name2, type2, ...)** - shows a dialog that can be used to specify the script parameters.

**doc** is a reference to the document; could be zero if all **type** values are "number" or "string"

**par** is a variable name (either numerical or string variable). The variable should be created before the Dialog function is called (see example below)

**name** is a string that will be shown in a dialog,

**type** is a string that specifies parameter type. It should be one of: "number", "string", "neuron", "neuronorevent", "event", "interval", "wave", or "all".

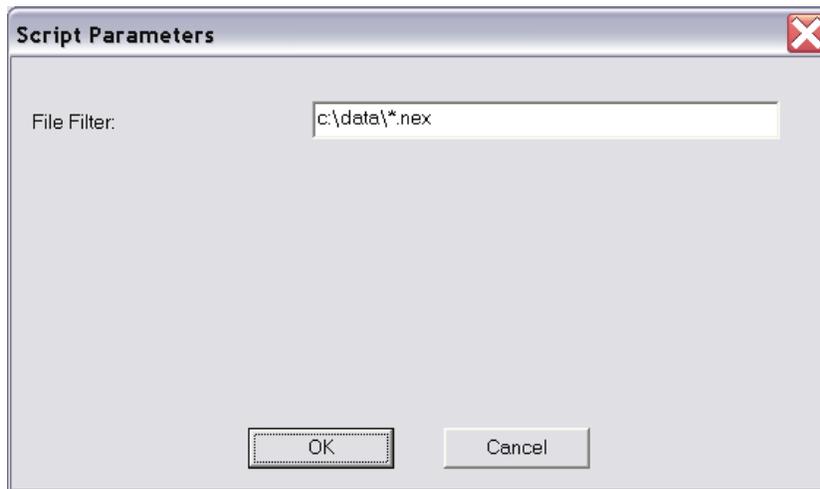
Dialog function returns 1 if user pressed OK button or 0, if user pressed Cancel button.

### Example

```
% create a string variable
filefilter = "c:\data\*.nex"

% show the dialog to the user
res = Dialog(0., filefilter , "File Filter:", "string")
```

The following dialog will be shown:



Now 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)
```

## 4.5.13. Debugging Functions

**Trace(par1, par2, ... )** - prints values of parameters to the debug window, returns nothing.

**Sleep(nms)** - pauses the script execution for **nms** milliseconds.

## 5. Working with 3D Graphics

### 5.1. Introduction

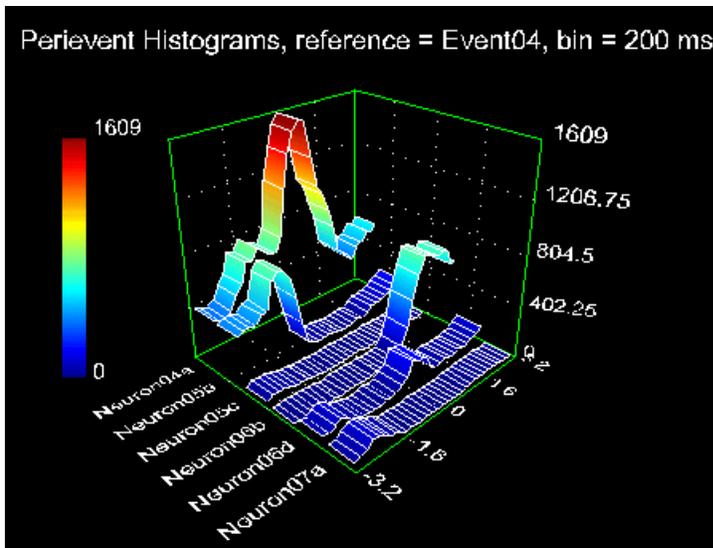
NeuroExplorer 3D Graphics Module (included in NeuroExplorer version 2.0 or later) adds high quality 3D graphics to NeuroExplorer. It was always possible before to export the analysis results from NeuroExplorer to Matlab and then create the 3D graphs in Matlab. However, it would require a considerable amount of time and effort to create the properly labeled 3D graph in Matlab. With the NeuroExplorer 3D Module installed, a single click of a button will display in NeuroExplorer a highly customizable and properly labeled 3D graph.

You can easily change dozens of graph parameters and options using the Properties Window or dialogs. You can even rotate the graph with the mouse in all 3 dimensions.

As with the NeuroExplorer Analysis Templates, any set of 3D graph parameters can be saved as a template. For example, you may save one set of color preferences for slide presentations and another set of colors for printed materials.

NeuroExplorer 3D Module also enables you to display a 'movie' of the concurrent activity of a neuronal network. You can specify a position of each recorded neuron in a plane and then display the animation of the activity of the neurons over time.

Here is an example of a 3D graph in NeuroExplorer. Multiple perievent histograms are shown side-by-side in 3 dimensions:

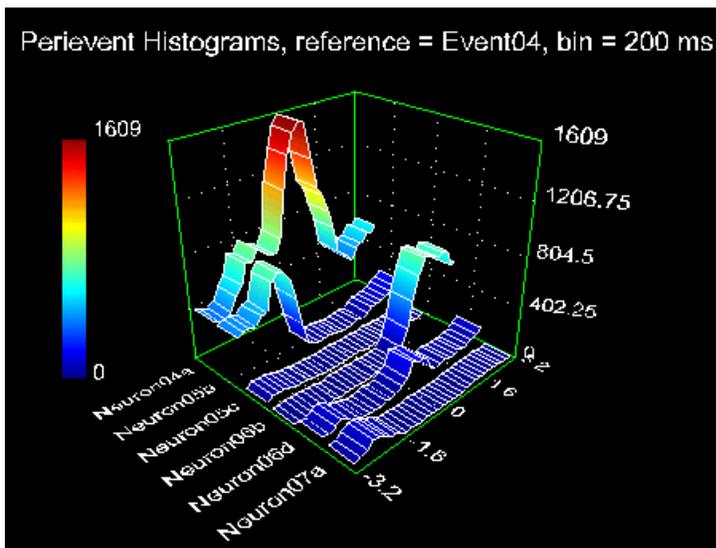


## 5.2. Viewing Multiple Histograms in 3D

To view a set of histograms in 3 dimensions:

1. Calculate the histograms by applying any of the histogram analyses (Rate Histograms, Interspike Interval Histograms, Auto- and Cross-correlograms, Perievent Histograms, etc.)
2. Select **3D | View Histograms in 3D** menu command.

NeuroExplorer will open the 3D Viewer window and all the histograms in the NeuroExplorer graphics view will be shown in a 3D graph:

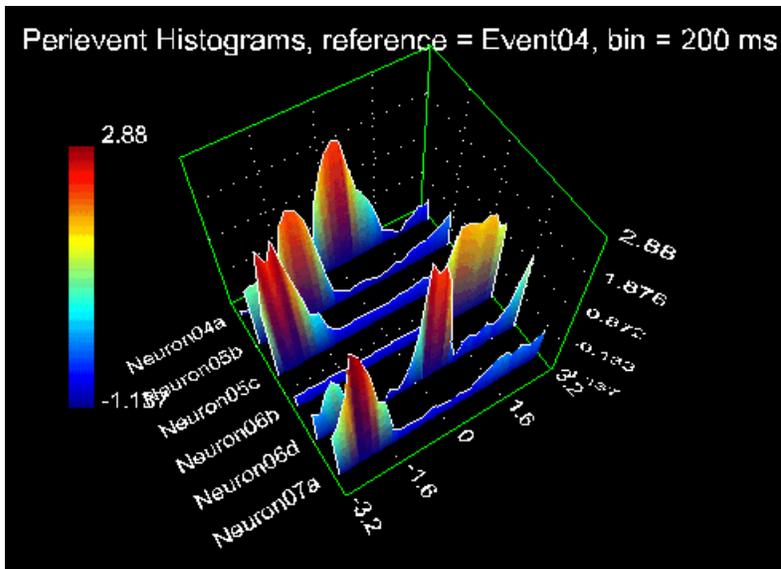


Note that all the histograms are shown in the same scale. However, neurons often have very different firing rates so that the histograms for the slow firing neurons may be displayed as having almost zero values.

If you would like to view the variations of the histograms around their respective means, you can use a separate menu command, **3D | View Histogram Variations in 3D**. This option will allow you to view the z-scores of histograms instead of the raw histogram values. z-score is calculated as:

$$z = (\text{histogram\_value} - \text{histogram\_mean}) / \text{histogram\_standard\_deviation}$$

Here is the same set of histograms as above shown using **3D | View Histogram Variations in 3D** menu command (the graph type is also changed from 'Stripes' to 'Walls'):



For more information on 3D Viewer options and parameters, see [3D Graphics Parameters](#).

### 5.3. 3D Graphics Parameters

**3D Template** - allows you to choose the 3D Template. When you start using 3D Module, only one 3D Template, **Default** is available. You can add templates by saving any set of 3D graphics parameters as a new template (to do this, use **3D | Save As New 3D Template** menu command).

**Graph Type** - allows you to choose the graph type.

**Draw Lines** - an option to draw mesh or contour lines.

**Z Color Scale** - an option to use color scale in the graph.

**Light** - an option to use lighting effects in the graph.

**Smooth** - an option to smooth the graph using a 2D Gaussian filter.

**Smooth Radius** - smooth filter radius in mesh points.

**Z Min, Z Max** - overwritable minimum and maximum of the Z Axis.

**Distance From Camera** - this parameter determines the size of the graph in the window. The distance should be between 1 and 10.

**Draw Labels** - an option to draw axes labels.

**Title** - 3D graph title.

**Title Font Size, Label Font Size** - font sizes relative to the 3D graph 'cube' size.

**Light X, Y, Z** - position of the light source.

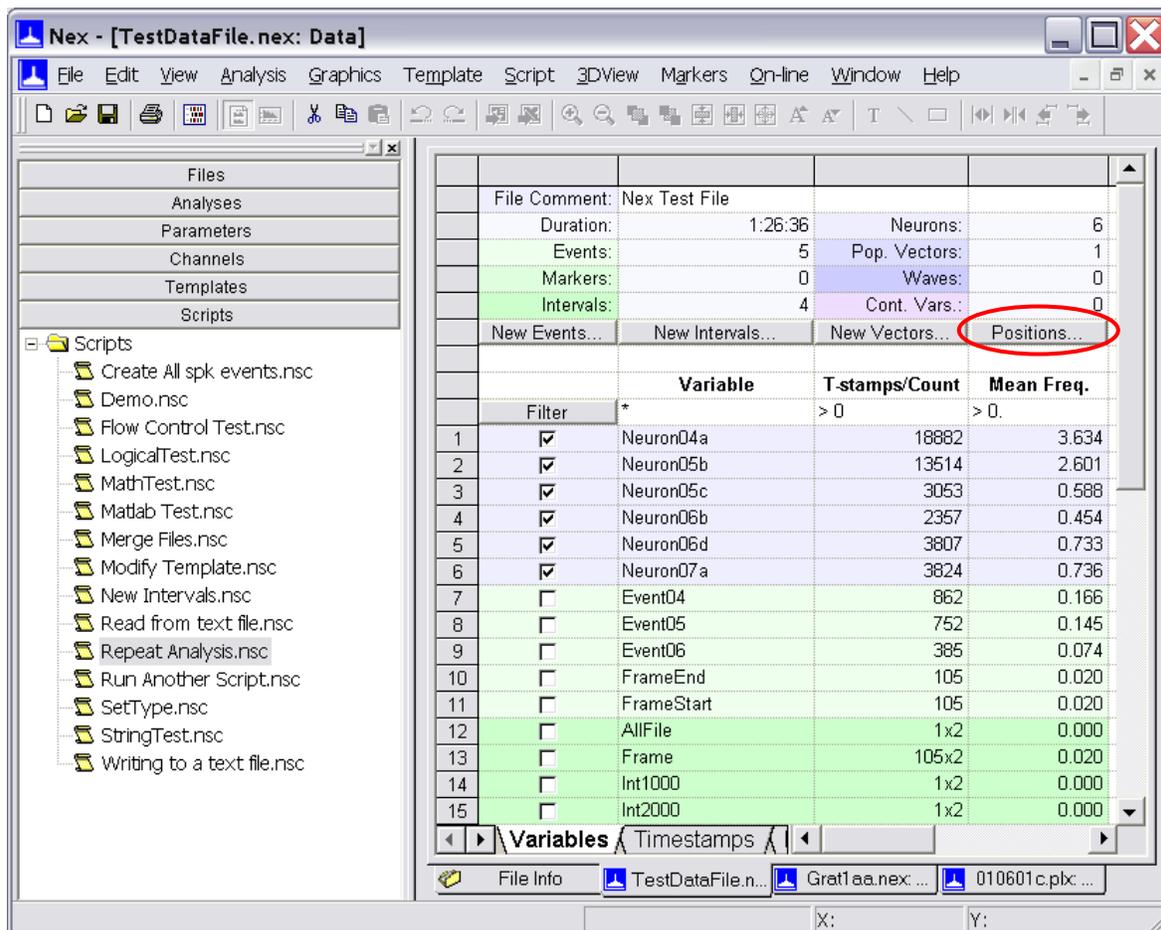
## 5.4. Viewing the Neuronal Activity "Movie"

To view a 'movie' of neuronal activity:

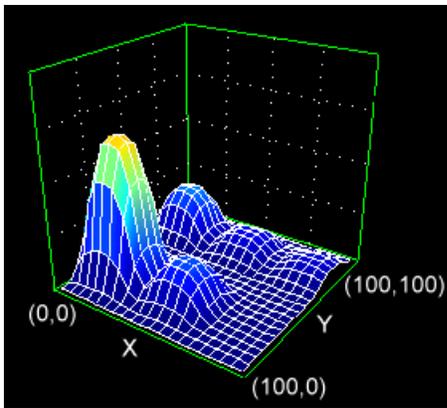
1. Open the data file.

Next, you need to specify the positions of neurons. Neuron positions are saved in a NeuroExplorer data file, so you only need to do this once.

2. Press Positions button in the Variables sheet of the Data View:



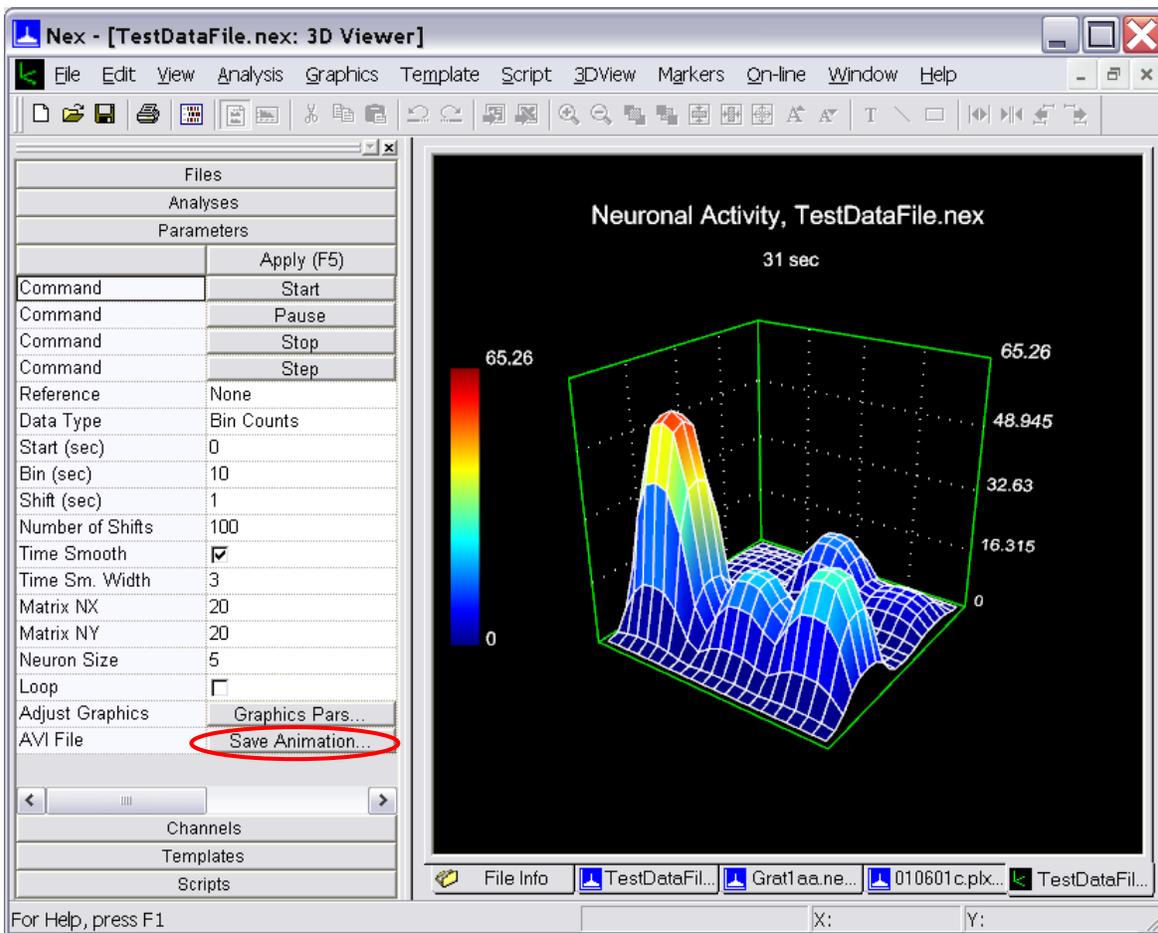
3. Specify the positions of neurons in the Positions Dialog Box. NeuroExplorer assumes that all the neurons are located on a plane with X and Y coordinates ranging from 0 to 100:



4. Select **3D | Activity Animation** menu command.

5. Use the buttons in the Properties window to start and stop the animation.

You can save animation to an AVI file. To do this, press "Save Animation" button in the control panel:



You may want to change the 3D graph properties to get the smooth surface shown above. To change the graph properties, right-click in the 3D Viewer window and select **Graphics Parameters** menu command. Select Surface plot type and enable Gaussian Smoothing.

For more information on 3D Viewer options and parameters, see [3D Graphics Parameters](#).

For more information on Activity Animation options and parameters, see [Activity Animation Parameters](#).

## 5.5. Activity Animation Parameters

**Animation Template** - allows you to choose the Animation Template. When you start using 3D Module, only one Animation Template, **Default** is available. You can add templates by saving any set of Animation parameters as a new template (to do this, use **3D | Save As New Animation Template** menu command).

**Reference** - if this parameter is **None**, NeuroExplorer shows the rate histograms-based animation (that is, neuronal activity in real time). If this parameter is not None, NeuroExplorer calculates the perievent histograms for the specified reference and shows neural network activity around the specified event.

NeuroExplorer uses a sliding window to calculate the firing rates of the neurons. The window has the width equal to the **Bin**. The first window begins at **Start** time. The second window begins at **Start+Shift** and ends at **Start+Shift+Bin**, etc. For each of the windows, the number of spikes that each neuron has inside this window is calculated. Then for each neuron the point (X\_position, Y\_position, Spike\_Count) is shown in a 3D graph.

**Start** - beginning of the first sliding window.

**Bin** - width of the sliding window.

**Shift** - sliding window shift.

**Number of shifts** - number of windows (frames).

**Time Smooth** - allows you to enable or disable time smoothing of the spike counts.

**Time Smooth Width** - sigma of the Gaussian filter used for smoothing over time. See [Post-Processing Options](#) for more information about smoothing the histograms.

**Matrix NX** - the number of points in the mesh in X direction.

**Matrix NY** - the number of points in the mesh in Y direction.

**Neuron Size** - the size of the square (in mesh points) that represents a single neuron.

**Loop** - an option to show the animation in a 'loop' mode (after all the frames are shown, animation goes back to the first frame and starts again).

# Index

## 1

1D Data Viewer, 18, 36, 39, 43, 44

## 3

3D Graphics, 36, 100, 101, 102, 103, 104, 105

## A

Algebraic expressions, 82  
Alpha Omega, 11, 12  
Analysis Parameters, 21, 24, 25, 26, 27, 45, 46, 47  
Animation, 104, 105  
Arrays of Variables in NexScript, 91  
Autocorrelograms, 35, 48, 52, 58

## B

Boxcar Filter, 24, 33, 46  
Burst Analysis, 35, 38, 67, 68

## C

CED Spike-2 Files, 11, 12  
Coherence, 35, 44, 77  
Conditional Operators, 84  
Confidence Limits, 35, 48, 52, 53, 55  
Continuously Recorded Data, 12, 43, 44, 59, 71, 77, 91, 92, 95  
Correlations with Continuous Variable, 4, 44, 71  
Counts/Bin, 50, 51, 52, 53, 54, 55, 56, 59, 63, 64, 70  
Crosscorrelograms, 28, 35, 55, 57, 96  
Cumulative Activity Display, 35, 61  
Cyberkinetics, 11, 12

## D

Data Acquisition Systems, 11, 12  
Data Import Options, 10, 13, 43, 44  
Data Selection, 35, 38, 45, 50, 51, 52, 53, 55, 58, 59, 60, 63, 64, 67, 69, 76, 77  
Data Types, 36  
DataWave Technologies, 11, 12

## E

Epoch Counts, 4, 35, 76  
Epochs, 76  
Excel, 6, 16, 24, 27, 35, 47, 50, 51, 52, 54, 56, 63, 65, 67, 69, 70, 77, 98  
Expressions, 80, 82

## F

File Access Functions in NexScript, 85, 87  
File Info Window, 8, 10  
File Viewer, 10  
Flow control, 80, 83  
**for** loop, 80, 83

## G

Gaussian, 24, 33, 46, 47, 48, 102, 104, 105  
Global variables, 82  
Graph Mode, 29, 30  
Graphics, 6, 7, 25, 27, 28, 29, 30, 32, 34, 100, 102, 104, 105

## I

**if** operator, 84  
Importing Data, 6, 10, 11, 12, 13, 15, 16, 26, 38, 43, 44  
Importing Data from Spreadsheets, 11, 15, 38  
Instant Frequency, 35, 61  
Interspike Interval Histograms, 35, 51, 88, 101  
Interspike Intervals vs. Time, 62  
Interval Filter, 33, 50, 51, 52, 53, 58, 59, 60, 63, 64, 67, 69, 76, 77  
Interval Variable, 35, 38, 39, 58, 62, 82, 90, 93

## J

Joint PSTH, 35, 60, 61

## L

Lines, 34, 80, 102  
Logical expressions, 83

## M

Markers, 40, 58  
Matlab, 6, 16, 24, 26, 35, 47, 50, 51, 52, 54, 55, 60, 63, 65, 69, 70, 77, 79, 98, 100  
Merge Files, 88  
Multichannel Systems, 11, 12

## N

Neuralynx, 11, 12, 13  
NexScript, 6, 36, 38, 79, 80, 81, 83, 85, 89  
Normalization, 50, 51, 52, 53, 54, 55, 56, 59, 63, 64, 66, 70, 78  
Numerical Results, 6, 23, 27, 47, 69, 87, 88, 96, 97

## O

Opening and Saving Data Files, 87  
Opening Files, 6, 11

## P

Page Mode, 29, 30  
Perievent Histogram, 35, 41, 43, 44, 45, 48, 53, 54, 59, 70, 76, 96, 101  
Place Cell Analysis, 35, 73  
Plexon, 11, 12, 36, 80, 90  
Poincare Maps, 35, 62  
Population Vectors, 41, 42  
Positioning, 30, 32, 34  
Post-processing, 6, 24, 35, 46, 50, 51, 52, 53, 55, 59, 60, 64, 70, 77, 105  
Power Point, 24, 25, 27  
Principal Component Analysis, 35, 42, 69  
Probability, 51, 52, 53, 54, 55, 56, 59, 70  
Properties Window, 21, 23, 100  
PSTH versus Time, 4, 70

## R

Rasters, 35, 43, 44, 45, 58, 96  
Rate Histograms, 22, 35, 41, 43, 44, 50, 96, 101  
RC Electronics, 11, 12  
Rectangles, 34  
Regularity Analysis, 35, 72, 73  
Reverse Correlation, 35, 74, 75

## S

Saving, 6, 24, 25, 27, 87, 88  
Selecting Variables, 6, 20, 92  
Shift Predictor, 55, 57  
Smooth, 33, 50, 51, 52, 53, 55, 59, 60, 64, 70, 74, 77, 102, 105

Spectral Densities, 35, 44, 64, 65, 77  
Spectrogram Analysis, 4, 35, 44, 77  
Spike Distance vs. Time, 62  
Spike Trains, 36, 37, 39, 65, 90  
Spikes/Second, 50, 51, 52, 53, 55, 59, 63, 70  
Spreadsheets, 6, 11, 15, 38  
String Functions in NexScript, 85, 86

## T

Template, 6, 8, 9, 19, 22, 32, 100, 102, 105  
Text Files, Importing Data XE "Importing Data" from, 6, 11  
Text Files, Reading and Writing, 87, 89  
Text Import, 13  
Text Labels, 32  
Trial Bin Counts, 35, 63

## U

User Interface Functions, 98

## V

Variable Names in NexScript, 81  
Variable Types in NexScript, 81  
Variables Window, 20

## W

**while** loop, 80, 83, 84

## X

X Axis, 30, 73, 75, 96  
X Range, 18

## Y

Y Axis, 31, 73, 75, 96