# Table of Contents

## 1. Getting Started 7

1.1. Getting Started with NeuroExplorer 8  
1.2. Working with Sentinel Keys 9  
1.3. NeuroExplorer Screen Elements 11  
1.4. Opening Files and Importing Data 15  
1.5. Importing Files Created by Data Acquisition Systems 16  
1.6. Importing Data from Text Files 18  
1.7. Importing Data from Spreadsheets 19  
1.8. Importing Data from Matlab 21  
1.9. Reading and Writing NeuroExplorer Data Files 22  
1.10. 1D Data Viewer 23  
1.11. Analyzing Data 24  
1.12. Selecting Variables for Analysis 25  
1.13. How to Select Variables for Existing Window 26  
1.14. How to Select Variables in Variables Panel 28  
1.15. Adjusting Analysis Properties 29  
1.16. Analysis Templates 30  
1.17. Numerical Results 31  
1.18. Post-processing 32  
1.19. Saving Results as Power Point Slides 32  
1.20. Working with Matlab 34  
1.21. Working with Excel 35  
1.22. Saving Graphics 36  

## 2. Analysis Reference 37

2.1. Data Types 38  
2.1.1. Spike Trains 38  
2.1.2. Events 39  
2.1.3. Intervals 39  
2.1.4. Markers 41  
2.1.5. Population Vectors 42  
2.1.6. Waveforms 43  
2.1.7. Continuously Recorded Data 43  
2.2. Data Selection Options 45  
2.3. Post-Processing Options 46  
2.4. Matlab Options 47  
2.5. Excel Options 47  
2.6. Confidence Limits for Perievent Histograms 48  
2.7. Cumulative Sum Graphs 49  
2.8. Rate Histograms 50  
2.9. Interspike Interval Histograms 52  
2.10. Autocorrelograms 55  
2.11. Perievent Histograms 58  
2.12. Crosscorrelograms 63  
2.13. Shift-Predictor for Crosscorrelograms 68  
2.14. Rasters 70  
2.15. Perievent Rasters 71  
2.16. Joint PSTH 74  
2.17. Cumulative Activity Graphs 76  
2.18. Instant Frequency 77  
2.19. Interspike Intervals vs. Time 78  
2.20. Poincare Maps 79
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.21. Synchrony vs. Time</td>
<td>80</td>
</tr>
<tr>
<td>2.22. Trial Bin Counts</td>
<td>82</td>
</tr>
<tr>
<td>2.23. Power Spectral Densities</td>
<td>84</td>
</tr>
<tr>
<td>2.24. Burst Analysis</td>
<td>87</td>
</tr>
<tr>
<td>2.25. Principal Component Analysis</td>
<td>90</td>
</tr>
<tr>
<td>2.26. PSTH Versus Time</td>
<td>92</td>
</tr>
<tr>
<td>2.27. Correlations with Continuous Variable</td>
<td>94</td>
</tr>
<tr>
<td>2.28. Regularity Analysis</td>
<td>96</td>
</tr>
<tr>
<td>2.29. Place Cell Analysis</td>
<td>98</td>
</tr>
<tr>
<td>2.30. Reverse Correlation</td>
<td>102</td>
</tr>
<tr>
<td>2.31. Epoch Counts</td>
<td>105</td>
</tr>
<tr>
<td>2.32. Coherence Analysis</td>
<td>107</td>
</tr>
<tr>
<td>2.33. Spectrogram Analysis</td>
<td>110</td>
</tr>
<tr>
<td>2.34. Perievent Spectrograms</td>
<td>112</td>
</tr>
<tr>
<td>2.35. Joint ISI Distribution</td>
<td>114</td>
</tr>
<tr>
<td>2.36. Autocorrelograms Versus Time</td>
<td>116</td>
</tr>
<tr>
<td><strong>3. Working with Graphics</strong></td>
<td>118</td>
</tr>
<tr>
<td>3.1. NeuroExplorer Graphics</td>
<td>118</td>
</tr>
<tr>
<td>3.2. Graphics Modes</td>
<td>118</td>
</tr>
<tr>
<td>3.3. Positioning the Graphics Objects</td>
<td>120</td>
</tr>
<tr>
<td>3.4. Text Labels</td>
<td>121</td>
</tr>
<tr>
<td>3.5. Lines</td>
<td>122</td>
</tr>
<tr>
<td>3.6. Rectangles</td>
<td>123</td>
</tr>
<tr>
<td><strong>4. Working with 3D Graphics</strong></td>
<td>124</td>
</tr>
<tr>
<td>4.1. Viewing Multiple Histograms in 3D</td>
<td>125</td>
</tr>
<tr>
<td>4.2. 3D Graphics Parameters</td>
<td>126</td>
</tr>
<tr>
<td>4.3. Viewing the Neuronal Activity &quot;Movie&quot;</td>
<td>127</td>
</tr>
<tr>
<td>4.4. Activity Animation Parameters</td>
<td>129</td>
</tr>
<tr>
<td><strong>5. Programming with NexScript</strong></td>
<td>130</td>
</tr>
<tr>
<td>5.1. Script Variables</td>
<td>132</td>
</tr>
<tr>
<td>5.2. File Variables</td>
<td>133</td>
</tr>
<tr>
<td>5.3. Expressions</td>
<td>135</td>
</tr>
<tr>
<td>5.4. Flow Control</td>
<td>136</td>
</tr>
<tr>
<td>5.5. Functions</td>
<td>138</td>
</tr>
<tr>
<td>5.5.1. File Read and Write Functions</td>
<td>139</td>
</tr>
<tr>
<td>5.5.1.1. GetFileCount Function</td>
<td>140</td>
</tr>
<tr>
<td>5.5.1.2. GetFileName Function</td>
<td>141</td>
</tr>
<tr>
<td>5.5.1.3. OpenFile Function</td>
<td>142</td>
</tr>
<tr>
<td>5.5.1.4. CloseFile Function</td>
<td>143</td>
</tr>
<tr>
<td>5.5.1.5. ReadLine Function</td>
<td>144</td>
</tr>
<tr>
<td>5.5.1.6. WriteLine Function</td>
<td>145</td>
</tr>
<tr>
<td>5.5.1.7. OpenDocument Function</td>
<td>146</td>
</tr>
<tr>
<td>5.5.1.8. NewDocument Function</td>
<td>147</td>
</tr>
<tr>
<td>5.5.1.9. CloseDocument Function</td>
<td>148</td>
</tr>
<tr>
<td>5.5.1.10. SaveDocument Function</td>
<td>149</td>
</tr>
<tr>
<td>5.5.1.11. SaveDocumentAs Function</td>
<td>150</td>
</tr>
<tr>
<td>5.5.1.12. SaveNumResults Function</td>
<td>151</td>
</tr>
<tr>
<td>5.5.1.13. SaveNumSummary Function</td>
<td>152</td>
</tr>
<tr>
<td>5.5.1.14. SaveAsTextFile Function</td>
<td>153</td>
</tr>
<tr>
<td>5.5.1.15. MergeFiles Function</td>
<td>154</td>
</tr>
<tr>
<td>5.5.1.16. ReadBinary Function</td>
<td>155</td>
</tr>
<tr>
<td>5.5.1.17. FileSeek Function</td>
<td>156</td>
</tr>
<tr>
<td>5.5.1.18. SelectFile Function</td>
<td>157</td>
</tr>
<tr>
<td>5.5.2. Document Properties Functions</td>
<td>158</td>
</tr>
</tbody>
</table>
5.5.2.1. GetDocPath Function 159
5.5.2.2. GetDocTitle Function 160
5.5.2.3. GetTimestampFrequency Function 161
5.5.2.4. GetDocEndTime Function 162
5.5.2.5. SetDocEndTime Function 163
5.5.2.6. GetDocComment Function 164
5.5.3. Document Variables Functions 165
5.5.3.1. GetVarCount Function 166
5.5.3.2. GetVarName Function 167
5.5.3.3. GetVarSpikeCount Function 168
5.5.3.4. GetVar Function 169
5.5.3.5. DeleteVar Function 170
5.5.3.6. Delete Function 171
5.5.3.7. GetVarByName Function 172
5.5.3.8. NewEvent Function 173
5.5.3.9. NewIntEvent Function 174
5.5.3.10. NewPopVector Function 175
5.5.3.11. GetContNumDataPoints Function 176
5.5.3.12. NewContVar Function 177
5.5.3.13. CopySelectedVarsToAnotherFile Function 178
5.5.4. Variable Selection Functions 179
5.5.4.1. IsSelected Function 180
5.5.4.2. Select Function 181
5.5.4.3. Deselect Function 182
5.5.4.4. SelectVar Function 183
5.5.4.5. DeselectVar Function 184
5.5.4.6. Select Function 185
5.5.4.7. Deselect Function 186
5.5.4.8. SelectAll Function 187
5.5.4.9. DeselectAll Function 188
5.5.4.10. SelectAllNeurons Function 189
5.5.4.11. SelectAllEvents Function 190
5.5.4.12. EnableRecalcOnSelChange Function 191
5.5.4.13. DisableRecalcOnSelChange Function 192
5.5.5. Properties of Variables Functions 193
5.5.5.1. GetName Function 194
5.5.5.2. GetSpikeCount Function 195
5.5.5.3. AddTimestamp Function 196
5.5.5.4. SetNeuronType Function 197
5.5.5.5. AddContValue Function 198
5.5.5.6. AddInterval Function 199
5.5.6. Analysis Functions 200
5.5.6.1. ApplyTemplate Function 201
5.5.6.2. ApplyTemplateToWindow Function 202
5.5.6.3. PrintGraphics Function 203
5.5.6.4. Dialog Function 204
5.5.6.5. ModifyTemplate Function 206
5.5.6.6. RecalculateAnalysisInWindow Function 208
5.5.6.7. EnableRecalcOnSelChange Function 209
5.5.6.8. DisableRecalcOnSelChange Function 210
5.5.6.9. SendGraphicsToPowerPoint Function 211
5.5.6.10. SaveGraphics Function 212
5.5.7. Numerical Results Functions 213
5.5.7.1. GetNumRes Function 214
5.5.7.2. GetNumResNCols Function 215
5.5.7.3. GetNumResNRows Function 216
5.5.7.4. GetNumResColumnName Function 217
5.5.7.5. SendResultsToExcel Function 218
5.5.7.6. GetNumResSummaryNCols Function 219
5.5.7.7. GetNumResSummaryNRows Function 220
5.5.7.8. GetNumResSummaryColumnName Function 221
5.5.7.9. GetNumResSummaryData Function 222
5.5.7.10. SendResultsSummaryToExcel Function 223
5.5.7.11. SaveNumResults Function 224
5.5.7.12. SaveNumSummary Function 225

5.5.8. Operations on Variables Functions 226

5.5.8.1. Rename Function 228
5.5.8.2. Join Function 229
5.5.8.3. Sync Function 230
5.5.8.4. NotSync Function 231
5.5.8.5. FirstAfter Function 232
5.5.8.6. FirstNAfter Function 233
5.5.8.7. LastBefore Function 234
5.5.8.8. IntervalFilter Function 235
5.5.8.9. SelectTrials Function 236
5.5.8.10. SelectRandom Function 237
5.5.8.11. SelectEven Function 238
5.5.8.12. SelectOdd Function 239
5.5.8.13. ISIFilter Function 240
5.5.8.14. FirstInInterval Function 241
5.5.8.15. LastInInterval Function 242
5.5.8.16. StartOfInterval Function 243
5.5.8.17. EndOfInterval Function 244
5.5.8.18. MakeIntervals Function 245
5.5.8.19. MakeIntFromStart Function 246
5.5.8.20. MakeIntFromEnd Function 247
5.5.8.21. IntOpposite Function 248
5.5.8.22. IntOr Function 249
5.5.8.23. IntAnd Function 250
5.5.8.24. IntSize Function 251
5.5.8.25. IntFind Function 252
5.5.8.26. MarkerExtract Function 253
5.5.8.27. Shift Function 254
5.5.8.28. NthAfter Function 255
5.5.8.29. PositionSpeed Function 256
5.5.8.30. FilterContinuousVariable Function 257
5.5.8.31. LinearCombinationOfContVars Function 258
5.5.8.32. DecimateContVar Function 259

5.5.9. Matlab Functions 260

5.5.9.1. SendSelectedVarsToMatlab Function 261
5.5.9.2. ExecuteMatlabCommand Function 262
5.5.9.3. GetVarFromMatlab Function 263
5.5.9.4. GetContVarFromMatlab Function 264
5.5.9.5. GetContVarWithTimestampsFromMatlab Function 265
5.5.9.6. GetIntervalVarFromMatlab Function 266

5.5.10. Excel Functions 267

5.5.10.1. SetExcelCell Function 268
5.5.10.2. CloseExcelFile Function 269

5.5.11. Power Point Functions 270

5.5.11.1. SendGraphicsToPowerPoint Function 271
5.5.11.2. ClosePowerPointFile Function 272

5.5.12. Running Script Functions 273

5.5.12.1. RunScript Function 274
5.5.12.2. Sleep Function 275

5.5.13. Math Functions 276

5.5.13.1. seed Function 278
5.5.13.2. expo Function 279
5.5.13.3. floor Function 280
5.5.13.4. ceil Function  281
5.5.13.5. round Function  282
5.5.13.6. abs Function  283
5.5.13.7. sqrt Function  284
5.5.13.8. pow Function  285
5.5.13.9. exp Function  286
5.5.13.10. min Function  287
5.5.13.11. max Function  288
5.5.13.12. log Function  289
5.5.13.13. sin Function  290
5.5.13.14. cos Function  291
5.5.13.15. tan Function  292
5.5.13.16. acos Function  293
5.5.13.17. asin Function  294
5.5.13.18. atan Function  295
5.5.13.19. RoundToTS Function  296
5.5.13.20. GetFirstGE Function  297
5.5.13.21. GetFirstGT Function  298
5.5.13.22. GetBinCount Function  299
5.5.13.23. BitwiseAnd Function  300
5.5.13.24. BitwiseOr Function  301
5.5.13.25. GetBit Function  302
5.5.14. String Functions  303
5.5.14.1. Left Function  304
5.5.14.2. Mid Function  305
5.5.14.3. Right Function  306
5.5.14.4. Find Function  307
5.5.14.5. StrLength Function  308
5.5.14.6. NumToStr Function  309
5.5.14.7. StrToNum Function  310
5.5.14.8. GetNumFields Function  311
5.5.14.9. GetField Function  312
5.5.14.10. CharToNum Function  313
5.5.14.11. NumToChar Function  314
5.5.15. Debug Functions  315
5.5.15.1. Trace Function  316
5.5.15.2. MsgBox Function  317

6. COM/ActiveX Interfaces  318
6.1. Application  319
6.1.1. ActiveDocument Property  320
6.1.2. DocumentCount Property  321
6.1.3. Version Property  322
6.1.4. Visible Property  323
6.1.5. OpenDocument Method  324
6.1.6. Document Method  325
6.1.7. Sleep Method  326
6.1.8. RunNexScript Method  327
6.1.9. RunNexScriptCommands Method  328
6.2. Document  329
6.2.1. Path Property  331
6.2.2. FileName Property  332
6.2.3. Comment Property  333
6.2.4. TimestampFrequency Property  334
6.2.5. StartTime Property  335
6.2.6. EndTime Property  336
6.2.7. VariableCount Property  337
6.2.8. NeuronCount Property  338

Page 5
6.2.9. EventCount Property  339
6.2.10. IntervalCount Property  340
6.2.11. MarkerCount Property  341
6.2.12. WaveCount Property  342
6.2.13. ContinuousCount Property  343
6.2.14. Variable Method  344
6.2.15. Neuron Method  345
6.2.16. Event Method  346
6.2.17. Interval Method  347
6.2.18. Marker Method  348
6.2.19. Wave Method  349
6.2.20. Continuous Method  350
6.2.21. DeselectAll Method  351
6.2.22. SelectAllNeurons Method  352
6.2.23. SelectAllContinuous Method  353
6.2.24. ApplyTemplate Method  354
6.2.25. GetNumericalResults Method  355

6.3. Variable  357
  6.3.1. Name Property  358
  6.3.2. TimestampCount Property  359
  6.3.3. Timestamps Method  360
  6.3.4. IntervalStarts Method  361
  6.3.5. IntervalEnds Method  362
  6.3.6. FragmentTimestamps Method  363
  6.3.7. FragmentCounts Method  364
  6.3.8. ContinuousValues Method  365
  6.3.9. MarkerValues Method  366
  6.3.10. WaveformValues Method  367
  6.3.11. SamplingRate Property  368
  6.3.12. Select Method  369
  6.3.13. Deselect Method  370
1. Getting Started

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, please exit all currently running applications. Please follow the following steps to install NeuroExplorer:

• Insert NeuroExplorer Setup USB flash drive into your computer USB port
• Navigate to NeuroExplorer4Setup.exe file on the flash drive and double-click on the file. NeuroExplorer Version 4 setup screen appears
• Follow the prompts of 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
• Reboot your computer

NeuroExplorer Setup will create the following directory structure:

• The main NeuroExplorer directory: C:\Program Files\Nex Technologies\NeuroExplorer
• SentinelDrivers subdirectory: C:\Program Files\Nex Technologies\NeuroExplorer\SentinelDrivers. If you need to reinstall Sentinel Drivers, you can run SentinelSetup.7.5.8.exe program in this directory.
• Additional NeuroExplorer files are copied to Windows Application Data directory (C:\Documents and Settings\All Users\Application Data\Nex Technologies\NeuroExplorer under Windows XP, or C:\Application Data\Nex Technologies\NeuroExplorer under Vista or Windows 7). This directory also contains Scripts and Templates folders where NeuroExplorer scripts and analysis templates are stored.

Installing Hardware Key
Before you can use NeuroExplorer, you need to install provided Sentinel Hardware Key on your computer.

To install the USB key:

• Make sure to run the NeuroExplorer setup and reboot the computer after installing NeuroExplorer
• Attach the key to the available USB port
• If the New Hardware Found wizard is shown, accept the defaults in the wizard
1.1. Getting Started with NeuroExplorer

This section, *Getting Started with NeuroExplorer*, describes the basics of using NeuroExplorer:

- Working with Sentinel keys
- 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 sections:

- Working with Graphics
- NeuroExplorer Analysis Reference
- Programming with NexScript

**NeuroExplorer Technical Support**

NeuroExplorer users can get help via e-mail: support@neuroexplorer.com. Please visit NeuroExplorer Web site [http://www.neuroexplorer.com](http://www.neuroexplorer.com) for program updates and latest information about NeuroExplorer.

**NeuroExplorer Updates**


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. Working with Sentinel Keys

General

NeuroExplorer requires a Sentinel key to operate. Sentinel drivers need to be installed so that NeuroExplorer can communicate with the Sentinel keys. These drivers are installed when you run NeuroExplorer setup (NeuroExplorer4Setup.exe).

Error Messages and Troubleshooting

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Description and Probable Cause</th>
<th>Troubleshooting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to initialize Sentinel library</td>
<td>Sentinel drivers are not installed or driver version is not supported.</td>
<td>Make sure that Sentinel drivers version 7.5.8 or later are installed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Open Control Panel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If an older version (prior to 7.5.8) of Sentinel Driver installer is listed, remove this version and reboot the computer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- To install the 7.5.8 drivers, run the driver installer:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C:\Program Files\Nex Technologies\NeuroExplorer\SentinelDrivers\SentinelSetup.7.5.8.exe</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Reboot the computer after installing the drivers.</td>
</tr>
<tr>
<td>Unable to find Sentinel key</td>
<td>NeuroExplorer Sentinel key is not attached to the computer or there is a problem communicating with the key.</td>
<td>1) Verify that you are using a NeuroExplorer key. The code SRB10491 should be printed on the key. The newer keys also have Neuroexplorer printed on them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2) Make sure that Sentinel drivers version 7.5.8 or later are installed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Open Control Panel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If an older version (prior to 7.5.8) of Sentinel Driver installer is listed, remove this version and reboot the computer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- To install the 7.5.8 drivers, run the driver installer:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C:\Program Files\Nex Technologies\NeuroExplorer\SentinelDrivers\SentinelSetup.7.5.8.exe</td>
</tr>
<tr>
<td>Unable to read Sentinel key data</td>
<td>Malfunctioning or damaged Sentinel key.</td>
<td>Please contact NeuroExplorer technical support (<a href="mailto:support@neuroexplorer.com">mailto:support@neuroexplorer.com</a>) if you still have problems with the Sentinel key.</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Sentinel key does not have version 4 license</td>
<td>You are using NeuroExplorer version 3 Sentinel key.</td>
<td>If you purchased NeuroExplorer version 4, please contact NeuroExplorer technical support (<a href="mailto:support@neuroexplorer.com">mailto:support@neuroexplorer.com</a>) If you would like to upgrade to NeuroExplorer version 4, please contact NeuroExplorer support (<a href="mailto:support@neuroexplorer.com">support@neuroexplorer.com</a>) or one of the NeuroExplorer resellers listed in the Purchase page of the NeuroExplorer web site: <a href="http://www.neuroexplorer.com/purchase.htm">http://www.neuroexplorer.com/purchase.htm</a></td>
</tr>
</tbody>
</table>
1.3. NeuroExplorer Screen Elements

NeuroExplorer user interface consists of the main window surrounded by several panels. These panels allow you to select files, select analyses and specify analysis properties. The figure below shows the default layout of NeuroExplorer window. Files, Properties, Analyses, Variables and Explore panels are visible and Templates and Scripts panels are hidden. To view a hidden panel, click at the panel tab or use View menu command.

You can reposition panels by dragging them with your mouse and you can save and restore window layouts using Window menu commands.

**Files Panel**

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 Properties panel. To open the
data file, simply double-click the file name.

**Analyses Panel**

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

**Properties Panel**

The left column of the Properties Panel 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 Panel**

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

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

You can analyze all the variables in your data file, a subset of the variables, or may be just one variable. Variables Panel 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 Panel

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

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

Explore Panel
Explore panel allows you to quickly explore analysis parameter ranges. Select a parameter you want to explore from a list and drag a slider. NeuroExplorer will keep recalculating analysis with the new parameter values while you drag the slider, thus creating an “analysis animation”.

One of the best uses for this panel is to explore analysis properties over a sliding window within your file:

- Specify Use Time Range using Analysis | Edit menu command, then Data Selection tab
- Select ‘Select Data From’ and ‘Select Data To’ parameters in the list boxes of the Explore panel
- Click on Link Sliders check box

Now, when you drag one of the sliders, the other one will follow allowing you to visualize analysis results over a sliding window in time.
1.4. Opening Files and Importing Data

NeuroExplorer can read native data files created by popular data acquisition systems (Alpha Omega, CED Spike-2, Cortex, Blackrock Microsystems, DataWave, Multi Channel Systems, Neuralynx, Plexon, RC Electronics. See Importing Files Created By Data Acquisition Systems for more information).

NeuroExplorer can also import data from text files (see Importing Data from Text Files ).

You can import the data from spreadsheets using the clipboard (see Importing Data from Spreadsheets ).

NeuroExplorer has its own data format and by default saves the data in the binary file with the extension .nex.

To open a NeuroExplorer data file,

- press File Open toolbar button , or
- select File | Open... menu command.

To import data in any of the supported file formats, select the corresponding File | Import command:

You can also paste data directly into Data View. See Importing Data from the Text Files and Importing Data from Spreadsheets for more information.
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, Blackrock Microsystems, Multi Channel Systems, Neuralynx, Plexon, 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).

Blackrock Microsystems Files

NeuroExplorer can import Blackrock Microsystems NEV files. All the spike trains and DIO events Analog channels can be imported as an option. The waveforms are imported when using Neuroshare DLL to import NEV files.

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 are not imported. 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

You can 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:

\[
\text{Internal\_Timestamp} = \text{Timestamp\_In\_Seconds} \times \text{Sampling\_Frequency}
\]

NeuroExplorer can import timestamped 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:

<table>
<thead>
<tr>
<th>Neuron01</th>
<th>Neuron02</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.001</td>
</tr>
<tr>
<td>0.3</td>
<td>0.05</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
</tr>
</tbody>
</table>

   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 that 64 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  |

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:

<table>
<thead>
<tr>
<th>ContChannel1</th>
<th>ContChannel2</th>
</tr>
</thead>
<tbody>
<tr>
<td>114.74609</td>
<td>-63.47656</td>
</tr>
<tr>
<td>56.15234</td>
<td>-358.88672</td>
</tr>
<tr>
<td>-187.98828</td>
<td>-63.47656</td>
</tr>
<tr>
<td>-48.82813</td>
<td>388.18359</td>
</tr>
<tr>
<td>-26.85547</td>
<td>285.64453</td>
</tr>
</tbody>
</table>

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

data. For data row N (N = 1, 2, …) in the text file, the timestamp is calculated using the following

formula:

DataRowTimestamp = FirstDataPoint + (N-1)*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:

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 (use File | Save or File | SaveAs menu command) 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 **Matlab | Get Data From Matlab... | Open Matlab As Engine** menu command. NeuroExplorer will open Matlab as engine.
- In opened Matlab, run your Matlab scripts and create (or load from file) spike train variables or continuous variables.
- To import timestamp variables, in NeuroExplorer, select **Matlab | Get Data 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 **Matlab | Get Data 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
...

Each variable header contains the size of the array that stores the variable data as well as the location of this array in the file.

The pseudo-code for reading NeuroExplorer data file looks like this:

```cpp
open file in binary mode

read file header

for each variable
    read variable header
end for

for each variable header
    seek to the file offset specified in the variable header
    read variable data
end for
```

The C++ source code of the program that reads and writes NeuroExplorer data files is available at NeuroExplorer web site:

http://www.neuroexplorer.com/downloads/HowToReadAndWriteNexFiles.zip

There are also Matlab scripts to read and write NeuroExplorer data files:

http://www.neuroexplorer.com/downloads/HowToReadAndWriteNexFilesInMatlab.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 Properties Panel).

Mouse wheel can be used to quickly navigate in 1D View:
- Click in 1D View so set 1D View as an active window in NeuroExplorer
- Press Shift and rotate mouse wheel to shift 1D View horizontally
- Press Ctrl and rotate mouse wheel to increase time range (zoom out) and decrease time range (zoom in)
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 clicking at the corresponding analysis in the Analyses Panel (you can also use Analysis | Select Analysis menu commands):

After you click at one of the analysis items, 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 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 using the Variable Panel (see How to select variables using Variables Panel)
• Select the variables using Analysis | Select Variables menu command. This command will invoke the Variable Selection dialog similar to Variables Panel.
• Select the variables directly in the Variables Window by using the check boxes located next to the variable names. Please note that variable selection in Variables Window is applied only when a new Graph Window or 1D View is created. To change the list of selected variables in existing Graph Window, use Variables Panel or Analysis | Select Variables menu command. See How to Select Variables for Existing Analysis Window for details.
1.13. How to Select Variables for Existing Window

Variables selected in the Variables spreadsheet of data window are used to populate variables lists of new analysis windows.

If you would like to change the list of variables for existing analysis window, please follow these steps:

- Click anywhere in analysis window to activate it
- If Variables panel is visible, make changes in selected variables list (how?) and press Apply button:

  ![Image showing how to select variables](image.png)

- If Variables panel is hidden, click at panel tab to open it:
• Make changes in selected variables list (how to make changes?) and press Apply button
• If Variables panel or its tab is not visible, use View | Variables Panel menu command
1.14. How to Select Variables in Variables Panel

Select variables for analysis by moving them from the left (available) to the right (selected) list box. In the figure below, 6 variables (Neuron04a, Neuron05b, ..., Neuron07a) are selected for analysis:

To move variables from one listbox to another:

- Select variables using the mouse (in the figure above two variables in Available listbox are selected -- ContChannel01 and ContChannel02)
- Press one of four buttons just above the listboxes

The drop-down box above the buttons can be used to specify what kind of variables are shown in the available window:
1.15. 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. Double-click (or right-click) anywhere in the Graph Window to invoke this menu:

![Analysis Parameters Menu](image)

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 analysis configuration as a new template.

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

![Properties Panel](image)
1.16. Analysis Templates

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

- Analysis type (Rate Histograms, I/IH, 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 Window (see NeuroExplorer Screen Elements). You can apply the template to the currently opened file by double-clicking the template name in the Templates Window.

NeuroExplorer stores template files in Windows Application Data directory:

*C:\Documents and Settings\All Users\Application Data\Nex Technologies\NeuroExplorer\Templates* under Windows XP, or
*C:\Application Data\Nex Technologies\NeuroExplorer\Templates* under Vista or Windows 7.
1.17. 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 Panel:

NeuroExplorer will open Numerical Results Window:

Note that the window has two Excel-style sheets -- the Results sheet and the Summary sheet. The Results sheet usually contains the 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.
1.18. 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 Options for details.

See Saving Results as Power Point Slides, Working with Matlab and Working with Excel for more information on NeuroExplorer post-processing options.

1.19. 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 notebook where you can save your NeuroExplorer analysis results:
1.20. Working with Matlab

NeuroExplorer can interact with Matlab via COM interface using Matlab as a powerful post-processing engine. Matlab can also control NeuroExplorer and exchange data with NeuroExplorer via NeuroExplorer COM interfaces. See COM/ActiveX Interfaces for details.

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.

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 Matlab | 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** - vector (or matrix with 2 columns) of all A/D values in milliVolts. If “Optimize transfer for small amounts of data” is selected in View | Options | Matlab, ContVar1 is a matrix with 2 columns: the first column contains A/D values in millivolts, the second column contains timestamps in seconds.
- **ContVar1_ts** - vector of fragment 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.
- **ContVar1_ts_step** - digitizing step in seconds.

You can generate all the timestamps for continuous variable using this Matlab script:

```matlab
function [ ts ] = MakeTs( fragmentInd, fragmentTs, numValues, step )
% MakeTs: makes timestamps for continuous variable based on the fragment information
% INPUT: fragmentInd - vector of fragment indexes
% fragmentTs - vector of fragment timestamps
% numValues - total number of values in all fragments
% step - digitizing step of continuous variable
%
% Example: NeuroExplorer sent continuous variable FP01 via file transfer
% The following values were sent:
% FP01 - vector of continuous values
% FP01_ind - fragment indexes
% FP01_ts - fragment timestamps
% FP01_ts_step - digitizing step
% to generate all timestamps for FP01, use:
% ts = MakeTs(FP01_ind, FP01_ts, size(FP01,1), FP01_ts_step);
%
% ts = [];
numFr = size(fragmentTs);
% add fake index for the last fragment
fragmentInd = [fragmentInd; numValues+1];
for i=1:numFr
    valuesInFragment = fragmentInd(i+1)-fragmentInd(i);
    ts = [ts; fragmentTs(i)+valuesInFragment*step];
end
```

You can generate all the timestamps for continuous variable using this Matlab script:
You can also import timestamped and continuous variables directly from Matlab. See Importing Data From Matlab for details.

See Also
Importing Data From Matlab
COM/ActiveX Interfaces

1.21. 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 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.
1.22. 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. Analysis Reference

This section, *NeuroExplorer Analysis Reference*, describes general pre-processing and post-processing analysis options, analysis parameters and computational algorithms used in analyses.

- Data Types
- 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
2.1. Data Types

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

2.1.1. 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 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 have the timestamp equal to 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

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

NeuroExplorer also requires that

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

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 SPK01a, you can use this script:

```
doc = GetActiveDocument()
doc.SPK01a[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:

![Image of NeuroExplorer interface](image)

2.1.2. 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.

2.1.3. 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:

```nexscript
doc = GetActiveDocument()
doc.MyInterval = NewIntEvent(doc)
AddInterval(doc.MyInterval, 0., 100.)
AddInterval(doc.MyInterval, 200., 300.)
```

Limitations

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

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:
2.1.4. 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.

NeuroExplorer creates Marker variables when it imports strobed events from Plexon data files, when it imports Trial Descriptor Records from the Datawave files and when DIO events with flags are loaded from Alpha Omega files.

Viewers

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

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:
2.1.5. 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:

$$\text{histogram	extunderscore of	extunderscore neuron	extunderscore 1} \times \text{weight	extunderscore 1} + \text{histogram	extunderscore of	extunderscore neuron	extunderscore 2} \times \text{weight	extunderscore 2} \ldots$$

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.

2.1.6. 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.

To select a different waveform variable, click in the cell located in the row 1 of the Variable column (the cell shows sig001i_wf in the figure above).

Waveform values are shown in columns labeled 1, 2, etc.
2.1.7. 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 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
- Spectrograms
- Perievent Spectrograms

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 in your computer 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.
2.2. 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.
2.3. 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 of width 3, the boxcar filter is

$$f[-1] = 0.33333, f[0] = 0.33333, f[1] = 0.33333.$$

and the smoothed histogram $sh[i]$ is calculated as

$$sh[i] = f[-1]*h[i-1] + f[0]*h[i] + f[1]*h[i+1]$$

where $h[i]$ is the original histogram.

The Gaussian filter is calculated using the following formula:

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

where

$$d = \frac{(\text{int})w + 1}{2}$$

$$\text{sigma} = -w * w * 0.25 / \log(0.5)$$

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

The parameters of the filter are such that the width of the Gaussian curve at half the height equals to the specified filter width. Please note that for the Gaussian filter, width of the filter ($w$) can be a non-integer. For example, $w$ can be 3.5.

See **Matlab Options** and **Excel Options** for more information on NeuroExplorer post-processing capabilities.
2.4. Matlab Options

NeuroExplorer can interact with Matlab via COM interface using Matlab as a powerful post-processing engine. Immediately after calculating histograms, NeuroExplorer can send the resulting matrix of histograms to Matlab and then ask Matlab to execute any series of Matlab commands. Use the Matlab tab in the Analysis parameters dialog to specify the matrix name and the Matlab command string.

2.5. 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.
2.6. 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:

\[
F = \frac{N}{T}
\]

Several options how to calculate neuron frequency \( F \) are available. See Options below.

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

\[
P = F \times b
\]

The expected bin count for the perievent histogram is then:

\[
C = P \times N_{\text{Ref}}, \text{where } N_{\text{Ref}} \text{ is the number of the reference events.}
\]

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:

Low Conf. = \( x \) such that \( \text{Prob}(S < x) = 0.005 \)

High Conf. = \( y \) such that \( \text{Prob}(S > y) = 0.005 \)

If \( C < 30 \), NeuroExplorer uses the actual Poisson distribution

\[
\text{Prob}(S = K) = \exp(-C) \times \frac{(C^K)}{K!}, \text{where } C^K \text{ is } C \text{ to the power of } K,
\]

to calculate the confidence limits.

If \( C \geq 30 \), the Gaussian approximation is used. For example, for 99% confidence limits:

Low Conf. = \( C - 2.58 \times \sqrt{C} \);

High Conf. = \( C + 2.58 \times \sqrt{C} \);

Reference


Options

The following options to calculate neuron frequency \( F \) are available:

- Use all file data. Here \( T \) is the total length of experimental session, \( N \) is the total number of spikes for a given neuron.
- Use selected time range and interval filter. \( T \) is the length of all the time intervals used in analysis, \( N \) is the number of spikes within these intervals.
- Use time intervals corresponding to prereference bins. This option only works for a stimulation-type data. For example, if you stimulate every second and calculate PSTH with \( X_{\text{Min}}=-0.2 \), \( X_{\text{Max}}=0.2 \), NeuroExplorer can easily distinguish the spikes before and after the stimulus. However, if you stimulate every 200 ms, the spikes before the second stimulus are also the spikes after the first stimulus, so you cannot distinguish the spikes that should be used to calculate the mean firing rate. The algorithm: for each reference event timestamp \( r \), a time interval \( (r+X_{\text{Min}}, r) \) is created (where \( X_{\text{Min}} \) is PSTH or crosscorrelogram time axis minimum parameter; \( X_{\text{Min}} \) should be negative). \( T \) is the length of all \( (r+X_{\text{Min}}, r) \) intervals that do not overlap, \( N \) is the number of spikes in these intervals. If more than 5% of intervals overlap, \( F \) is set to zero.
2.7. 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 $bc[i]$, $i=1,...,N$. Cumulative Sum Graph displays the following values $cs[i]$:

for bin 1: $cs[1] = bc[1] - A$


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:


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 Kirilov.
### 2.8. Rate Histograms

Rate histogram displays firing rate versus time.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin/XMax type</td>
<td>An option on how XMin and XMax values are specified.</td>
</tr>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds</td>
</tr>
<tr>
<td>Bin</td>
<td>Histogram bin size in seconds.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability or Spikes/Second). See Algorithm</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by</td>
</tr>
<tr>
<td></td>
<td>Select Data From and Select Data To parameters) time range will be used</td>
</tr>
<tr>
<td></td>
<td>in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Smooth histogram</td>
<td>An option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also</td>
</tr>
<tr>
<td></td>
<td>Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results</td>
</tr>
<tr>
<td></td>
<td>are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to</td>
</tr>
<tr>
<td></td>
<td>Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form</td>
</tr>
<tr>
<td></td>
<td>CR where C is Excel column name, R is the row number. For example, A1</td>
</tr>
<tr>
<td></td>
<td>is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>

#### Summary of Numerical Results

The following information is available in the Summary of Numerical Results

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

**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

\[
\text{bin\_count} = \text{number of timestamps (ts) such that } ts \geq XMin \text{ and } ts < XMin + Bin
\]

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

If **Normalization** is Spikes/Sec, bin counts are divided by Bin.
2.9. Interspike Interval Histograms

Interspike interval histogram shows the conditional probability of a first spike at time \( t_0 + t \) after a spike at time \( t_0 \).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Interval</td>
<td>Minimum interspike interval in seconds.</td>
</tr>
<tr>
<td>Max Interval</td>
<td>Maximum interspike interval in seconds.</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability or Spikes/Second). See Algorithm below.</td>
</tr>
<tr>
<td>Log Bins and X Axis</td>
<td>An option to use Log10 interval scale (see Algorithm below for details).</td>
</tr>
<tr>
<td>Bins per decade</td>
<td>A bin size option if Log option is selected (see Algorithm below for details).</td>
</tr>
<tr>
<td>Y Axis Limit (%)</td>
<td>An option to “zoom in” small histogram values. This parameter specifies the maximum of Y axis. If it is 100%, the Y axis maximum equals to the maximum bin value. If it is, for example, 10%, Y axis maximum is set to 10% of the bin max and the small bin values have better visibility.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Overlay Graphs</td>
<td>An option to draw several histograms in each graph. This option requires that Int. filter type specifies that multiple interval filters will be used (either Table (row) or Table (col)).</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
</tbody>
</table>
### Add to Results / Bin

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

### Summary of Numerical Results

The following information is available in the Summary of Numerical Results:

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

### Algorithm

**If Use Log Bins and X Axis option is not selected:**

The time axis is divided into bins. The first bin is \([\text{IntMin}, \text{IntMin+Bin})\). The second bin is \([\text{IntMin+Bin}, \text{Intmin+Bin*2})\), 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

\[
\text{bin\_count} = \text{number of interspike intervals (isi)} \\
\text{such that isi} \geq \text{IntMin and isi} < \text{IntMin} + \text{Bin}
\]

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

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

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

If Use Log Bins and X Axis option is selected:

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

2.10. Autocorrelograms

Autocorrelogram shows the conditional probability of a spike at time $t_0 + t$ on the condition that there is a spike at time $t_0$.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds.</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability or Spikes/Second). See Algorithm below.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Overlay Graphs</td>
<td>An option to draw several histograms in each graph. This option requires that Int. filter type specifies that multiple interval filters will be used (either Table (row) or Table (col)).</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Draw confidence limits</td>
<td>An option to draw the confidence limits.</td>
</tr>
<tr>
<td>Confidence (%)</td>
<td>Confidence level (percent). See Confidence Limits for details.</td>
</tr>
<tr>
<td>Conf. mean calculation</td>
<td>An option that specifies how the mean firing rate (that is used in the derivation of the confidence limits) is calculated. There are 2 options: Use data selection and Use all file. See Confidence Limits for details.</td>
</tr>
<tr>
<td>Conf. display</td>
<td>An option to draw confidence limits either as horizontal lines or as a...</td>
</tr>
<tr>
<td><strong>Conf. line style</strong></td>
<td>Line style for drawing confidence limits (used when Conf. display is <strong>Lines</strong>).</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Conf. background color</strong></td>
<td>Background color for drawing confidence limits (used when Conf. display is <strong>Colored Background</strong>).</td>
</tr>
<tr>
<td><strong>Draw mean freq.</strong></td>
<td>An option to draw a horizontal line representing the expected histogram value for a Poisson spike train. See <strong>Confidence Limits</strong> for details.</td>
</tr>
<tr>
<td><strong>Mean line style</strong></td>
<td>Line style for drawing mean frequency.</td>
</tr>
<tr>
<td><strong>Add to Results / Bin left</strong></td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td><strong>Add to Results / Bin middle</strong></td>
<td>An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td><strong>Add to Results / Bin right</strong></td>
<td>An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td><strong>Send to Matlab</strong></td>
<td>An option to send the matrix of numerical results to Matlab. See also <strong>Matlab Options</strong>.</td>
</tr>
<tr>
<td><strong>Matrix Name</strong></td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td><strong>Matlab command</strong></td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td><strong>Send to Excel</strong></td>
<td>An option to send numerical results or summary of numerical results to Excel. See also <strong>Excel Options</strong>.</td>
</tr>
<tr>
<td><strong>Sheet Name</strong></td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td><strong>TopLeft</strong></td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>

**Summary of Numerical Results**

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Histogram minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Histogram maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
<tr>
<td>Filter Length</td>
<td>The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).</td>
</tr>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate (Spikes/Filter Length).</td>
</tr>
<tr>
<td>Mean Hist.</td>
<td>The mean of the histogram bin values.</td>
</tr>
<tr>
<td>St. Dev. Hist.</td>
<td>The standard deviation of the histogram bin values.</td>
</tr>
<tr>
<td>Conf. Low</td>
<td>Lower confidence level.</td>
</tr>
<tr>
<td>Conf. High</td>
<td>Upper confidence level.</td>
</tr>
<tr>
<td>Mean</td>
<td>Expected mean value of the histogram.</td>
</tr>
</tbody>
</table>
**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 \([X_{\text{Min}}, X_{\text{Min}} + \text{Bin})\). The second bin is \([X_{\text{Min}} + \text{Bin}, X_{\text{Min}} + \text{Bin}*2)\), 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 to \( k \):
   - if \( d[i] \) is inside the first bin, increment the bin counter for the first bin:
     \[ \text{if } d[i] \geq X_{\text{Min}} \text{ and } d[i] < X_{\text{Min}} + \text{Bin} \]
     \[ \text{then } \text{bincount}[1] = \text{bincount}[1] +1 \]
   - if \( d[i] \) is inside the second bin, increment the bin counter for the second bin:
     \[ \text{if } d[i] \geq X_{\text{Min}} + \text{Bin} \text{ and } d[i] < X_{\text{Min}} + \text{Bin}*2 \]
     \[ \text{then } \text{bincount}[2] = \text{bincount}[2] +1 \]
   - 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 \( d[i] \geq X_{\text{Min}} \) and \( d[i] < X_{\text{Min}} + \text{Bin} \), the bin count for the first bin can exceed the number of spikes in the spike train. Then, the probability value \( (\text{bincount}[1]/\text{number}\_\_\text{of}\_\text{spikes}) \) could be larger than 1.

If **Normalization** is **Spikes/Sec**, bin counts are divided by \( \text{NumSpikes}*\text{Bin} \), where NumSpikes is the number of spikes in the spike train.
## 2.11. Perievent Histograms

Perievent Histogram shows the conditional probability of a spike at time \( t_0 + t \) on the condition that there is a reference event at time \( t_0 \).

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Type</td>
<td>Specifies if the analysis will use a single or multiple reference events.</td>
</tr>
<tr>
<td>Reference</td>
<td>Specifies a reference neuron or event (or a group of reference neurons or events).</td>
</tr>
<tr>
<td>XMin</td>
<td>Histogram time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Histogram time axis maximum in seconds</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability, Spikes/Second or Z-score). See Algorithm below.</td>
</tr>
<tr>
<td>No Selfcount</td>
<td>An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Overlay Graphs</td>
<td>An option to draw several histograms in each graph. This option requires that Int. filter type specifies that multiple interval filters will be used (either Table (row) or Table (col)).</td>
</tr>
<tr>
<td>Overlay Options</td>
<td>Specifies line colors and line styles for overlaid graphs.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Draw confidence limits</td>
<td>An option to draw the confidence limits.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Confidence (%)</td>
<td>Confidence level (percent). See Confidence Limits for details.</td>
</tr>
<tr>
<td>Conf. mean calculation</td>
<td>An option that specifies how the mean firing rate (that is used in the derivation of the confidence limits) is calculated. There are 3 options: Use data selection, Use all file and Use pre-ref data. See Confidence Limits for details. Note that Use pre-ref data option can only be used for a stimulation type data, i.e. when the distance between any two consecutive reference events is larger than XMax-XMin. See Confidence Limits for details.</td>
</tr>
<tr>
<td>Conf. display</td>
<td>An option to draw confidence limits either as horizontal lines or as a colored background.</td>
</tr>
<tr>
<td>Conf. line style</td>
<td>Line style for drawing confidence limits (used when Conf. display is Lines).</td>
</tr>
<tr>
<td>Conf. background color</td>
<td>Background color for drawing confidence limits (used when Conf. display is Colored Background).</td>
</tr>
<tr>
<td>Draw mean freq.</td>
<td>An option to draw a horizontal line representing the expected histogram value for a Poisson spike train. See Confidence Limits for details.</td>
</tr>
<tr>
<td>Mean line style</td>
<td>Line style for drawing mean frequency.</td>
</tr>
<tr>
<td>Draw Cusum</td>
<td>An option to draw a cumulative sum graph above the histogram. See Cumulative Sum Graphs for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Background</td>
<td>An option on how to calculate the histogram background for the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Peak width</td>
<td>Peak width (the number of bins in peak) in the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Left shoulder</td>
<td>Specifies the left shoulder value (in seconds) in the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Right shoulder</td>
<td>Specifies the right shoulder value (in seconds) in the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>
Summary of Numerical Results

The following information is available in the Summary of Numerical Results

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

In general, the Perievent Histogram shows the conditional probability of a spike in the spike train at time $t_0 + t$ on the condition that there is a reference event (or reference spike) at time $t_0$.

The time axis is divided into bins. The first bin is $[X_{\text{Min}}, X_{\text{Min}} + \text{Bin}]$. The second bin is $[X_{\text{Min}} + \text{Bin}, X_{\text{Min}} + \text{Bin} \times 2]$, etc. The left end is included in each bin, the right end is excluded from the bin.

Let $\text{ref}[i]$ be the array of timestamps of the reference event, $\text{ts}[i]$ be the spike train (each $\text{ts}$ is the timestamp)

For each timestamp $\text{ref}[k]$:

1) calculate the distances from this event (or spike) to all the spikes in the spike train:
   $$d[i] = \text{ts}[i] - \text{ref}[k]$$

2) for each $i$:
   - if $d[i]$ is inside the first bin, increment the bin counter for the first bin:
     $$\text{if } d[i] \geq X_{\text{Min}} \text{ and } d[i] < X_{\text{Min}} + \text{Bin} \text{ then } \text{bincount}[1] = \text{bincount}[1] + 1$$
   - if $d[i]$ is inside the second bin, increment the bin counter for the second bin:
     $$\text{if } d[i] \geq X_{\text{Min}} + \text{Bin} \text{ and } d[i] < X_{\text{Min}} + \text{Bin} \times 2 \text{ then } \text{bincount}[2] = \text{bincount}[2] + 1$$
   and so on...

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

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

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

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

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

Peak and Trough Statistics
NeuroExplorer calculates the histogram peak statistics the following way:

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

Histogram trough statistics are calculated in a similar way. The only difference is that histogram minimum instead of histogram maximum is analyzed.
2.12. Crosscorrelograms
Crosscorrelogram shows the conditional probability of a spike at time \( t_0 + t \) on the condition that there is a reference event at time \( t_0 \).
Compared to PSTH (that calculates the same probabilities), crosscorrelogram allows to use shift-predictors.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Type</td>
<td>Specifies if the analysis will use a single or multiple reference events.</td>
</tr>
<tr>
<td>Reference</td>
<td>Specifies a reference neuron or event (or a group of reference neurons or events).</td>
</tr>
<tr>
<td>XMin</td>
<td>Histogram time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Histogram time axis maximum in seconds</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability, Spikes/Second or Z-score). See Algorithm below.</td>
</tr>
<tr>
<td>No Selfcount</td>
<td>An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating crosscorrelogram versus itself).</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is <em>From Time Range</em>, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Count bins in filter</td>
<td>An option to normalize each bin individually if the interval filter is used. See Algorithm section below for detailed discussion.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Overlay Graphs</td>
<td>An option to draw several histograms in each graph. This option requires that Int. filter type specifies that multiple interval filters will be used (either Table (row) or Table (col)).</td>
</tr>
<tr>
<td>Overlay Options</td>
<td>Specifies line colors and line styles for overlaid graphs.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Shift-predictor</td>
<td>An option to use shift-predictor. Shift-predictor requires that an interval filter is used in the analysis. See Using Shift-Predictor for details.</td>
</tr>
<tr>
<td>Shift times</td>
<td>How many times to calculate shift predictor. See Using Shift-Predictor for details.</td>
</tr>
<tr>
<td>Subtract predictor</td>
<td>An option to subtract shift-predictor from crosscorrelogram. See Using Shift-Predictor for details.</td>
</tr>
<tr>
<td>Predictor line style</td>
<td>Specifies the line used to draw shift-predictor (if subtract predictor is not selected).</td>
</tr>
<tr>
<td>Draw confidence limits</td>
<td>An option to draw the confidence limits.</td>
</tr>
<tr>
<td>Confidence (%)</td>
<td>Confidence level (percent). See Confidence Limits for details.</td>
</tr>
<tr>
<td>Conf. mean calculation</td>
<td>An option that specifies how the mean firing rate (that is used in the derivation of the confidence limits) is calculated. There are 3 options: Use data selection, Use all file and Use pre-ref data. See Confidence Limits for details. Note that Use pre-ref data option can only be used for a stimulation type data, i.e. when the distance between any two consecutive reference events is larger than XMax-XMin. See Confidence Limits for details.</td>
</tr>
<tr>
<td>Conf. display</td>
<td>An option to draw confidence limits either as horizontal lines or as a colored background.</td>
</tr>
<tr>
<td>Conf. line style</td>
<td>Line style for drawing confidence limits (used when Conf. display is Lines).</td>
</tr>
<tr>
<td>Conf. background color</td>
<td>Background color for drawing confidence limits (used when Conf. display is Colored Background).</td>
</tr>
<tr>
<td>Draw mean freq.</td>
<td>An option to draw a horizontal line representing the expected histogram value for a Poisson spike train. See Confidence Limits for details.</td>
</tr>
<tr>
<td>Mean line style</td>
<td>Line style for drawing mean frequency.</td>
</tr>
<tr>
<td>Draw Cusum</td>
<td>An option to draw a cumulative sum graph above the histogram. See Cumulative Sum Graphs for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Background</td>
<td>An option on how to calculate the histogram background for the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Peak width</td>
<td>Peak width (the number of bins in peak) in the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Left shoulder</td>
<td>Specifies the left shoulder value (in seconds) in the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
<tr>
<td>Right shoulder</td>
<td>Specifies the right shoulder value (in seconds) in the peak and through analysis. See Peak and Trough Statistics below.</td>
</tr>
</tbody>
</table>
Send to Matlab

An option to send the matrix of numerical results to Matlab. See also Matlab Options.

Matrix Name

Specifies the name of the results matrix in Matlab workspace.

Matlab command

Specifies a Matlab command that is executed after the numerical results are sent to Matlab.

Send to Excel

An option to send numerical results or summary of numerical results to Excel. See also Excel Options.

Sheet Name

The name of the worksheet in Excel where to copy the numerical results.

TopLeft

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

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

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

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

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

Let \( \text{ref}[i] \) be the array of timestamps of the reference event,

\( \text{ts}[i] \) be the spike train (each \( ts \) is the timestamp).

For each timestamp \( \text{ref}[k] \):

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

\[
d[i] = \text{ts}[i] - \text{ref}[k]
\]

2) for each \( i \):

   if \( d[i] \) is inside the first bin, increment the bin counter for the first bin:

   \[
   \text{if } d[i] \geq XMin \text{ and } d[i] < XMin + Bin \text{ then } \text{bincount}[1] = \text{bincount}[1] + 1
   \]

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

   \[
   \text{if } d[i] \geq XMin+Bin \text{ and } d[i] < XMin + Bin*2 \text{ then } \text{bincount}[2] = \text{bincount}[2] + 1
   \]

and so on...

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

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

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

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

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

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

**Peak and Trough Statistics**

NeuroExplorer calculates histogram peak statistics the following way:

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

Histogram trough statistics are calculated in a similar way. The only difference is that histogram 
minimum instead of histogram maximum is analyzed.
2.13. 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.

Shift-Predictor Algorithm

Suppose we have n trial intervals:

[t1a, t1b], [t2a, t2b], ..., [tna, tnb]

and we have 2 spike trains:
reference spike train with N timestamps r1, r2, ... , rN;
and target spike train with M timestamps s1, s2, ... , sM.

Shift-predictor for shift = 1 is calculated like this:

Step 1: find all the timestamps ri that are inside the first trial interval [t1a, t1b]
Step 2: find all the timestamps sj that are inside the second trial interval [t2a, t2b]
Step 3: shift all the timestamps ri so that thay align with the second trial:
calculate new timestamps pi = ri + (t2a-t1a)
Step 4: calculate a crosscorrelogram between pi and sj

Repeat Steps 1 through 4 with interval [t2a, t2b] in Step 1 and interval [t3a, t3b] in Step 2.
...
Repeat Steps 1 through 4 with interval [tna, tnb] in Step 1 and interval [t1a, t1b] in Step 2. Note that we wrap around the trials: the last trial is correlated with the first one.
Then, calculate shift-predictors for shift = 2:
Repeat Steps 1 through 4 with interval \([t_{1a}, t_{1b}]\) in Step 1 and interval \([t_{3a}, t_{3b}]\) in Step 2. That is, shift trials by 2.

... 
And, finally, calculate shift-predictors for shift = Number_of_shifts specified in the shift-predictor options.

**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.
2.14. Rasters

The raster display shows the spikes as vertical lines (tick) positioned according to the spikes timestamps. This analysis does not generate numerical results since the raster lines (ticks) are drawn directly from the data. Some generic analysis statistics are available in the Summary of Numerical Results (see below).

Parameters

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

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

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

Algorithm

The raster display shows the spikes as vertical lines positioned according to the spikes timestamps.
2.15. Perievent Rasters

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Type</td>
<td>Specifies if the analysis will use a single or multiple reference events.</td>
</tr>
<tr>
<td>Reference</td>
<td>Specifies a reference neuron or event (or a group of reference neurons or events).</td>
</tr>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds.</td>
</tr>
<tr>
<td>Markers</td>
<td>An option to display additional events (as markers - triangles, squares, etc.) in the perievent raster.</td>
</tr>
<tr>
<td>Background Intervals</td>
<td>An option to draw intervals in the raster background.</td>
</tr>
<tr>
<td>Trial Order</td>
<td>This option allows you to specify how the raster lines are drawn. The raster lines (corresponding to reference events) can be drawn in the order of reference events (with the first event on top, the last at the bottom of the perievent raster), or in the reverse order (the first event at the bottom of the raster display, the last event at the top). This option is ignored if Sort trials parameter is not None.</td>
</tr>
<tr>
<td>Sort trials</td>
<td>An option to sort raster lines. By default, the raster lines (corresponding to reference events) are drawn in the order specified by the Trial Order parameter. This option allows you to display reference events in a different order. For example, in behavioral experiments, you can sort the trials according to the latency of the response.</td>
</tr>
<tr>
<td>Sort event</td>
<td>Specifies how to identify the timestamps (in Sort ref. variable) that will be used in sorting.</td>
</tr>
<tr>
<td>Sort ref.</td>
<td>Specifies event variable that will be used to sort trials.</td>
</tr>
<tr>
<td>Histogram</td>
<td>An option to draw a histogram.</td>
</tr>
<tr>
<td>Bin</td>
<td>Histogram bin size in seconds.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability or Spikes/Second). See Algorithm below.</td>
</tr>
<tr>
<td>No Selfcount</td>
<td>An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
</tbody>
</table>
Create filter on-the-fly
Specifies if a temporary interval filter needs to be created (and used to preselect data).

Create filter around
Specifies an event that will be used to create a temporary filter.

Start offset
Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.

End offset
Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.

Fix overlaps
An option to automatically merge the overlapping intervals in the temporary filter.

Smooth histogram
An option to smooth the histogram after the calculation. See Post-Processing Options for details.

Smooth Filter Width
The width of the smooth filter. See Post-Processing Options for details.

### Summary of Numerical Results
The following information is available in the Summary of Numerical Results

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

### 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:
   \[ \text{dist}[k] = \text{sortref}[i] - \text{ref}[k] \]
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:

\[ \text{ref}[k] + X\text{Min}, \text{ref}[k] + X\text{Min} + \text{Bin} \]

the second bin is \[ \text{ref}[k] + X\text{Min} + \text{Bin}, \text{ref}[k] + X\text{Min} + \text{Bin} + \text{Bin} \], 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.
## 2.16. Joint PSTH

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

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds.</td>
</tr>
<tr>
<td>Reference</td>
<td>Specifies a reference neuron or event.</td>
</tr>
<tr>
<td>Bottom Neuron</td>
<td>The neuron of event shown along the horizontal axis (the vertical axis shows a neuron selected for analysis).</td>
</tr>
<tr>
<td>Diag</td>
<td>The width (in seconds) of the area in the scatter matrix around its main diagonal used to calculate the main diagonal histogram.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Scatter matrix normalization.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is <em>From Time Range</em>, only the data from the specified (by <em>Select Data From</em> and <em>Select Data To</em> parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in <em>Create filter around</em> parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in <em>Create filter around</em> parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histograms (not the scatter matrix) after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel.</td>
</tr>
</tbody>
</table>
Excel. See also Excel Options.

<table>
<thead>
<tr>
<th>Sheet Name</th>
<th>The name of the worksheet in Excel where to copy the numerical results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
<tr>
<td>Matrix Scale</td>
<td>An option on how to draw the scatter matrix (color or black and white).</td>
</tr>
<tr>
<td>Font size (% matrix)</td>
<td>Font size in percent of scatter matrix height.</td>
</tr>
</tbody>
</table>

**Summary of Numerical Results**

The following information is available in the Summary of Numerical Results

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

**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:

2.17. Cumulative Activity Graphs
The cumulative activity display shows a stepwise function which makes a jump at the moment of spike and stays constant between the spikes.
This analysis does not generate numerical results since the stepwise function is drawn directly from the data. Some generic analysis statistics are available in the Summary of Numerical Results (see below).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds</td>
</tr>
</tbody>
</table>

**Summary of Numerical Results**
The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate.</td>
</tr>
</tbody>
</table>

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

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

This analysis does not generate numerical results since the graph is drawn directly from the data. Some generic analysis statistics are available in the Summary of Numerical Results (see below).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds</td>
</tr>
<tr>
<td>Graph Style</td>
<td>An option to draw vertical lines or points.</td>
</tr>
</tbody>
</table>

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate.</td>
</tr>
</tbody>
</table>

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.
2.19. Interspike Intervals vs. Time

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

This analysis does not generate numerical results since the graph is drawn directly from the data. Some generic analysis statistics are available in the Summary of Numerical Results (see below).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds</td>
</tr>
</tbody>
</table>

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate.</td>
</tr>
</tbody>
</table>

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.
2.20. Poincare Maps

For each spike, a point is displayed where the X coordinate of the point is the current interspike interval and Y coordinate of the point is the preceding interspike interval. This analysis does not generate numerical results since the graph is drawn directly from the data. Some generic analysis statistics are available in the Summary of Numerical Results (see below).

Parameters

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

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate.</td>
</tr>
</tbody>
</table>

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.
2.21. Synchrony vs. Time

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

This analysis does not generate numerical results since the graph is drawn directly from the data. Some generic analysis statistics are available in the Summary of Numerical Results (see below).

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

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

![Graph showing distance to nearest spike](image)

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference event.</td>
</tr>
<tr>
<td>Distance</td>
<td>An option to draw linear or inverted distance.</td>
</tr>
</tbody>
</table>

### Summary of Numerical Results

The following information is available in the Summary of Numerical Results:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
</tbody>
</table>
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:

$$\text{dist} = \min(\text{abs}(t[i] - \text{ref}[j])),$$

where $\text{ref}[j]$ is a timestamp of the reference event.

If **Distance** is **Linear**, the vertical line is drawn from point $(t[i], 0.)$ to point $(t[i], \text{dist})$.

If **Distance** is **Inverted**, the vertical line is drawn from point $(t[i], 0.)$ to point $(t[i], 1/\text{dist})$. 

<table>
<thead>
<tr>
<th>YMin</th>
<th>Y axis minimum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate.</td>
</tr>
</tbody>
</table>
2.22. Trial Bin Counts

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

Parameters

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

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
</tbody>
</table>
Algorithm

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

The time axis is divided into bins. The first bin is \([X_{\text{Min}}, X_{\text{Min}}+\text{Bin})\). The second bin is \([X_{\text{Min}}+\text{Bin}, X_{\text{Min}}+\text{Bin}^*2)\), etc. The left end is included in each bin, the right end is excluded from the bin.

Let \(\text{ref}[i]\) be the array of timestamps of the reference event,

\(\text{ts}[i]\) be the spike train (each \(\text{ts}\) is the timestamp)

For each timestamp \(\text{ref}[k]\):

1. Set all bin counts to zero:
   \[
   \text{bincount}[i] = 0 \quad \text{for all } i
   \]

2. Calculate the distances from this event (or spike) to all the spikes in the spike train:
   \[
   d[i] = \text{ts}[i] - \text{ref}[k]
   \]
   for each \(i\):
   
   - if \(d[i]\) is inside the first bin, increment the bin counter for the first bin:
     
     \[
     \text{if } d[i] \geq X_{\text{Min}} \text{ and } d[i] < X_{\text{Min}} + \text{Bin} \\
     \text{then } \text{bincount}[1] = \text{bincount}[1] + 1
     \]
   
   - if \(d[i]\) is inside the second bin, increment the bin counter for the second bin:
     
     \[
     \text{if } d[i] \geq X_{\text{Min}}+\text{Bin} \text{ and } d[i] < X_{\text{Min}} + \text{Bin}^*2 \\
     \text{then } \text{bincount}[2] = \text{bincount}[2] + 1
     \]
   
   and so on...

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

If **Normalization** is **Spikes/Sec**, bin counts are divided by **Bin**.
### 2.23. Power Spectral Densities

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

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Frequency</td>
<td>Maximum frequency of the spectrum in Hz</td>
</tr>
<tr>
<td>Number of values</td>
<td>Number of values in the spectrum.</td>
</tr>
<tr>
<td>Show Frequency From</td>
<td>Minimum frequency to be shown in the spectrum graph.</td>
</tr>
<tr>
<td>Show Frequency To</td>
<td>Maximum frequency to be shown in the spectrum graph.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Spectrum normalization. See Algorithm below.</td>
</tr>
<tr>
<td>Log Frequency Scale</td>
<td>An option show Log10 frequency scale.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the spectrum after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Overlay Graphs</td>
<td>An option to draw several histograms in each graph. This option requires that Int. filter type specifies that multiple interval filters will be used (either Table (row) or Table (col)).</td>
</tr>
<tr>
<td>Overlay Options</td>
<td>Specifies line colors and line styles for overlaid graphs.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each bin)</td>
</tr>
</tbody>
</table>
right to the matrix of numerical results.

Send to Matlab
An option to send the matrix of numerical results to Matlab. See also Matlab Options.

Matrix Name
Specifies the name of the results matrix in Matlab workspace.

Matlab command
Specifies a Matlab command that is executed after the numerical results are sent to Matlab.

Send to Excel
An option to send numerical results or summary of numerical results to Excel. See also Excel Options.

Sheet Name
The name of the worksheet in Excel where to copy the numerical results.

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

Summary of Numerical Results
The following information is available in the Summary of Numerical Results

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

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:

\[
\text{Bin} = 1 / (2 \cdot \text{Maximum Frequency})
\]

Number_of_Bins = \(2 \cdot \text{Number_of_Frequency Values}\)

The time axis is divided into intervals of length Bin\(^*\)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:

- The rate histogram is calculated.
- The histogram is detrended (its linear regression is subtracted)
- The Hann window \( w[i] = \frac{1 - \cos(2\pi i/(\text{Number\_Of\_Bins}-1))}{2} \)

is applied to the histogram.

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

**Spectral Densities for Continuous Variables**

If \( \text{Maximum\_Frequency} = M \) and \( \text{Number\_of\_Frequency\_Values} = N \), we need to calculate the spectrum for frequency values \( 0, \frac{M}{N}, \frac{2M}{N}, ..., M \).

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 \( 0, \frac{F}{K}, \frac{2F}{K}, ..., \frac{F}{2} \).

If a continuous variable digitizing frequency \( F \) equals to \( \text{Maximum\_Frequency} \times 2 \), we can set \( K = \text{Number\_of\_Frequency\_Values} \times 2 \) 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 \( \text{Maximum\_Frequency} \times 2 \), 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. \( \frac{F}{K} < \frac{M}{N} \), 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 \( \text{Normalization} \) 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 \( \text{Normalization} \) is % of Total PSD, the power spectrum is normalized so that the sum of all the spectrum values equals 100.

If \( \text{Normalization} \) is Log of PSD, the power spectrum is calculated using the formula:

\[
\text{power\_spectrum}[i] = 10.0 \times \log_{10}(\text{raw\_spectrum}[i])
\]
2.24. Burst Analysis

This analysis identifies bursts in spike trains. For each spike train (neuron variable), a new interval variable is created that contains the time intervals corresponding to bursts.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Selection of MaxInterval or Surprise algorithms.</td>
</tr>
<tr>
<td>Max Interval</td>
<td>Maximum interspike interval to start the burst (in seconds, MaxInterval method).</td>
</tr>
<tr>
<td>Max End Interval</td>
<td>Maximum interspike interval to end the burst (in seconds, MaxInterval method).</td>
</tr>
<tr>
<td>Min Interburst Interval</td>
<td>Minimum interval between bursts (in seconds, MaxInterval method).</td>
</tr>
<tr>
<td>Min Burst Duration</td>
<td>Minimum burst duration (in seconds, MaxInterval method)</td>
</tr>
<tr>
<td>Min Number of Spikes</td>
<td>Minimum number of spikes in the burst (MaxInterval method).</td>
</tr>
<tr>
<td>Min Surprise</td>
<td>Minimum surprise of the burst (Surprise method).</td>
</tr>
<tr>
<td>Rate Histogram Bin</td>
<td>The value of the bin for the burst rate histogram (in seconds).</td>
</tr>
<tr>
<td>Rate Histogram Units</td>
<td>Histogram units (Counts/Bin, Probability or Spikes/Second). See Rate Histograms for details.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>
### Summary of Numerical Results

The following information is available in the Summary of Numerical Results

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

MaxInterval method
Find all the bursts using the following algorithm:

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

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

Freq = NumberOfSpikes / (FileEndTime – FileStartTime)

MeanISI = 1 / Freq

2) NeuroExplorer scans the spike train until it finds two sequential ISI's so that each of those ISIs 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 and we also assume that the burst has N spikes and the distance from the first to the last spike of the burst is T, then the surprise of the burst is:

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

3) NeuroExplorer adds the spikes to the end of the burst until the first ISI that is more than MeanISI and calculates surprise for each of the bursts (with 3 initial spikes, 4 spikes, 5 spikes, etc.) The burst with maximum surprise Smax is then selected.

4) NeuroExplorer removes the spikes from the beginning of the burst and calculates the surprise for each of the reduced bursts. The burst with maximum surprise Smax is then selected.

5) If Smax is more than MinSurprise and the number of spikes in the burst is more than 3, NeuroExplorer adds the burst to the result.

Reference
2.25. Principal Component Analysis

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

**Parameters**

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

**Summary of Numerical Results**

The following information is available in the Summary of Numerical Results

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

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

**Algorithm**

Step 1.

Rate histograms are calculated for each of the selected neurons.

The time axis is divided into bins. The first bin is \([0, \text{Bin})\). The second bin is \([\text{Bin}, \text{Bin}^*2)\), 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

\[
\text{bin\_count} = \text{number of timestamps (ts) such that ts} \geq 0 \text{ and ts} < \text{Bin}
\]

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

\[
\text{bin\_count}[i, j],
\]

where \(i\) is the neuron number, \(j\) is the bin number.

Step 2.

The matrix of correlations between neurons \(c[t, s]\) is calculated:

\[
c[t, s] = \text{correlation between vectors bin\_count}[t, *] \text{ and bin\_count}[s, *], \ s, t = 1, ..., \text{number of selected neurons}.
\]

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.
2.26. 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Specifies a reference neuron or event.</td>
</tr>
<tr>
<td>XMinPSTH</td>
<td>Histogram time minimum in seconds.</td>
</tr>
<tr>
<td>XMaxPSTH</td>
<td>Histogram time maximum in seconds.</td>
</tr>
<tr>
<td>BinPSTH</td>
<td>Histogram bin size in seconds.</td>
</tr>
<tr>
<td>Start</td>
<td>Start of the first sliding window in seconds.</td>
</tr>
<tr>
<td>Duration</td>
<td>Duration of the sliding window in seconds.</td>
</tr>
<tr>
<td>Shift</td>
<td>How much sliding window is shifted each time.</td>
</tr>
<tr>
<td>Number of shifts</td>
<td>The number of sliding windows to be used.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Histogram units (Counts/Bin, Probability or Spikes/Second). See Algorithm below.</td>
</tr>
<tr>
<td>No Selfcount</td>
<td>An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).</td>
</tr>
<tr>
<td>Sliding window axis direction</td>
<td>Direction of the sliding window axis (horizontal or vertical).</td>
</tr>
<tr>
<td>Sliding window axis alignment</td>
<td>Allows to specify what values are shown in the sliding window axis. There are two options: Start of Window and Center of Window. Suppose your data has the strongest correlation at 50 seconds. This means that the window that has 50 seconds as its center will show the highest PST values. If the window width is 20 seconds, it will be the window [40,60]. If you are using 'Start of window' option, you will see the peak at 40 instead of 50 seconds. However, with 'Center of window' option, the peak will be at 50 seconds.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
</tbody>
</table>
TopLeft Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>PSTH minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>PSTH maximum.</td>
</tr>
<tr>
<td>Color Scale Min</td>
<td>PSTH minimum.</td>
</tr>
<tr>
<td>Color Scale Max</td>
<td>PSTH maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes in the variable.</td>
</tr>
</tbody>
</table>

Algorithm

NeuroExplorer calculates multiple PST histograms (see Perievent Histogram for more information on how each PSTH is calculated). Each histogram is calculated for a different interval (window) in time: [window_start[i], window_end[i]], i = 1, ..., Number of Shifts.

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

The following rules are used to calculate the windows:

window_start[1] = Start
window_end[1] = Start + Duration
window_start[2] = Start + Shift
window_end[2] = Start + Shift + Duration
...

Page 93
2.27. 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Specifies a reference continuous variable.</td>
</tr>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds.</td>
</tr>
<tr>
<td>Bin Type</td>
<td>An option to specify how the bin value is determined. If this option is Auto, NeuroExplorer will use the bin equal to the A/D sampling interval of the reference variable.</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds. Used if Bin Type is set to User defined.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the histogram after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>
Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Mean Hist.</td>
<td>Mean of all the correlation values for this variable.</td>
</tr>
<tr>
<td>St. Dev. Hist</td>
<td>Standard deviation of all the correlation values for this variable.</td>
</tr>
<tr>
<td>St. Eff. Mean Hist</td>
<td>Standard error of mean of all the correlation values for this variable.</td>
</tr>
</tbody>
</table>

Algorithm

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

If x[i], i=1,...,N is the reference variable and y[i], i=1,...,N is another continuous variable or spike train rate histogram, then

c[t] = Pearson correlation between vectors

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

and 

{ y[t+1], y[t+2], ..., y[N] }
2.28. Regularity Analysis
This analysis estimates the regularity of neuronal firing after the stimulation.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Specifies a reference event or spike train.</td>
</tr>
<tr>
<td>XMin</td>
<td>Time axis minimum in seconds.</td>
</tr>
<tr>
<td>XMax</td>
<td>Time axis maximum in seconds.</td>
</tr>
<tr>
<td>Bin</td>
<td>Bin size in seconds. Used if Bin Type is set to User defined.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>ISI Graph</td>
<td>This parameter determines how the mean ISI values is displayed in the graph.</td>
</tr>
<tr>
<td>SD Graph</td>
<td>This parameter determines how the standard deviation of ISI is displayed in the graph.</td>
</tr>
<tr>
<td>CV Graph</td>
<td>This parameter determines how the CV (coefficient of variation) values are displayed in the graph.</td>
</tr>
<tr>
<td>CV Graph Max</td>
<td>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).</td>
</tr>
</tbody>
</table>

Add to Results / Bin left An option to add an additional vector (containing a left edge of each bin) to the matrix of numerical results.
Add to Results / Bin middle An option to add an additional vector (containing a middle point of each bin) to the matrix of numerical results.
Add to Results / Bin right An option to add an additional vector (containing a right edge of each bin) to the matrix of numerical results.

Summary of Numerical Results
The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>NumRefEvents</td>
<td>Number of reference events.</td>
</tr>
<tr>
<td>YMin</td>
<td>Histogram minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Histogram maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
</tbody>
</table>

Page 96
<table>
<thead>
<tr>
<th>Filter Length</th>
<th>The length of all the intervals of the interval filter (if a filter was used) or the length or the recording session (in seconds).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Freq.</td>
<td>Mean firing rate (Spikes/Filter_Length).</td>
</tr>
<tr>
<td>Mean Hist.</td>
<td>The mean of the histogram bin values (ISI values).</td>
</tr>
<tr>
<td>St. Dev. Hist.</td>
<td>The standard deviation of the histogram bin values.</td>
</tr>
<tr>
<td>St. Dev. ISI</td>
<td>The mean value of the SD ISI graph.</td>
</tr>
<tr>
<td>CV</td>
<td>The mean value of the CV ISI graph.</td>
</tr>
</tbody>
</table>

**Algorithm**

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

Let ref\([i]\) be the array of timestamps of the reference event,

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

For each timestamp ref\([k]\):

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

\[ d\([i]\) = ts\([i]\) - ref\([k]\) \]

1b) calculate the interspike interval

\[ isi\([i]\) = ts\([i+1]\) - ts\([i]\) \]

1c) for each i:

if \( d\([i]\) \) is inside the first bin ( \( d\([i]\) >= XMin \) and \( d\([i]\) < XMin + Bin \) )

and \( d\([i]\) + isi\([i]\) < XMax, 

add isi\([i]\) to the series of intervals for the first bin

if \( d\([i]\) \) is inside the second bin ( \( d\([i]\) >= XMin+Bin \) and \( d\([i]\) < XMin + Bin*2 \) )

and \( d\([i]\) + isi\([i]\) < XMax, add isi\([i]\) to the series of intervals for the second bin

and so on...

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

**Reference**

2.29. **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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Position</td>
<td>Continuous variable that describes X position of the animal.</td>
</tr>
<tr>
<td>Y Position</td>
<td>Continuous variable that describes Y position of the animal.</td>
</tr>
<tr>
<td>Fix Positions</td>
<td>This option specifies whether NeuroExplorer needs to fix problems that often occur in recording of position variables. For example, when LED is obscured by the wires, both position variables suddenly jump to 0 and then jump back to the previous position.</td>
</tr>
<tr>
<td>Fix Threshold</td>
<td><em>Fix Positions</em> parameter. See <em>Algorithm</em> section below for details.</td>
</tr>
<tr>
<td>Fix Bad X</td>
<td><em>Fix Positions</em> parameter. The 'bad' value of X variable. See <em>Algorithm</em> section below for details.</td>
</tr>
<tr>
<td>Fix Bad Y</td>
<td><em>Fix Positions</em> parameter. The 'bad' value of Y variable. See <em>Algorithm</em> section below for details.</td>
</tr>
<tr>
<td>X Min</td>
<td>X axis minimum in X Position units.</td>
</tr>
<tr>
<td>X Max</td>
<td>X axis maximum in X Position units.</td>
</tr>
<tr>
<td>Y Min</td>
<td>Y axis minimum in Y Position units.</td>
</tr>
<tr>
<td>Y Max</td>
<td>Y axis maximum in Y Position units.</td>
</tr>
<tr>
<td>Number of Bins (X)</td>
<td>Number of bins along X axis.</td>
</tr>
<tr>
<td>Number of Bins (Y)</td>
<td>Number of bins along Y axis.</td>
</tr>
<tr>
<td>Display</td>
<td>This parameter determines what kind of Place Cell Analysis display is shown (Path, Firing Positions, Time Spent (in each cell), Number of Visits (to each cell), Spike Counts (for each cell), Firing Rates (for each cell), Firing Fields).</td>
</tr>
<tr>
<td>Min. Time Spent</td>
<td>Minimum time spent. This parameter is used only with Firing Rates display. If the animal spent less time in the cell than <em>Min Time Spent</em>, the firing rate for the cell is set to zero.</td>
</tr>
<tr>
<td>Min Visits</td>
<td>Minimum number of visits. This parameter is used only with Firing Rates display. If the animal visited the cell less than <em>Min Visits</em> number of times, the firing rate for the cell is set to zero.</td>
</tr>
<tr>
<td>Firing Rate Bin</td>
<td>Bin size (in seconds) to calculate firing rates.</td>
</tr>
<tr>
<td>Min Std. Dev. in Field</td>
<td>Minimum standard deviation in the field. See <em>Algorithm</em> section below for details.</td>
</tr>
<tr>
<td>Min. Pixels in Field</td>
<td>Minimum number of pixels in the field. See <em>Algorithm</em> section below for details.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is <em>From Time Range</em>, only the data from the specified (by <em>Select Data From</em> and <em>Select Data To</em> parameters) time range will be used in analysis. See also <em>Data Selection Options</em>.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Create filter on-the-fly</td>
<td>Specifies if a temporary interval filter needs to be created (and used to preselect data).</td>
</tr>
<tr>
<td>Create filter around</td>
<td>Specifies an event that will be used to create a temporary filter.</td>
</tr>
<tr>
<td>Start offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the start of interval for the temporary filter.</td>
</tr>
<tr>
<td>End offset</td>
<td>Offset (in seconds, relative to the event specified in Create filter around parameter) for the end of interval for the temporary filter.</td>
</tr>
<tr>
<td>Fix overlaps</td>
<td>An option to automatically merge the overlapping intervals in the temporary filter.</td>
</tr>
<tr>
<td>Smooth Matrix</td>
<td>An option to smooth the matrix after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Radius</td>
<td>The radius of the smoothing filter (in cells) See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Colors</td>
<td>An option to smooth colors (blend colors across cells to make smooth color transitions). This option may require considerable computation time.</td>
</tr>
<tr>
<td>Bins with Num Visits Less than Min</td>
<td>An option on how to display cells where the number of visits is less than the specified minimum (Min Visits parameter).</td>
</tr>
<tr>
<td>Bins with Time Spent Less than Min</td>
<td>An option on how to display cells where the time spent is less than the specified minimum (Min. Time Spent parameter).</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>

### Summary of Numerical Results

The following information is available in the Summary of Numerical Results:

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

**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:

$$\text{aver} = \frac{x[t-2]+x[t-1]+x[t+1]+x[t+2]}{4}$$

if $|x[t] - \text{aver}| > \text{FixThreshold}$, then assign $x[t]$ the average of its neighbors:

$$x[t] = \text{aver}$$

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

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

The following steps are then performed:

The position space is divided into cells with width $(X_{\text{Max}} - X_{\text{Min}}) / \text{Number of Bins (X)}$ and height $(Y_{\text{Max}} - Y_{\text{Min}}) / \text{Number of Bins (Y)}$.

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

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

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

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

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

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

**Reference**

2.30. 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>the variable that contains timestamps of the stimuli.</td>
</tr>
<tr>
<td>Stim. Sequence File</td>
<td>The path of the text file that contains the sequence of images used for stimulation. See Algorithm below for file format description.</td>
</tr>
<tr>
<td>Char. per pixel</td>
<td>The number of characters per pixel in image file.</td>
</tr>
<tr>
<td>Pixels in Row</td>
<td>The number of pixels in a row for each stimulus image.</td>
</tr>
<tr>
<td>Rows in Image</td>
<td>The number of rows of pixels in each stimulus image.</td>
</tr>
<tr>
<td>Cache Stim. Data</td>
<td>An option to cache stimulus image data. If this option is selected, the image data is loaded once and stored in memory (until a different stimulus sequence file is specified). To clear cache, run this analysis once with this option disabled.</td>
</tr>
<tr>
<td>XMin (deg)</td>
<td>Minimum of the X axis of the average stimulus display (in degrees).</td>
</tr>
<tr>
<td>XMax (deg)</td>
<td>Maximum of the X axis of the average stimulus display (in degrees).</td>
</tr>
<tr>
<td>YMin (deg)</td>
<td>Minimum of the Y axis of the average stimulus display (in degrees).</td>
</tr>
<tr>
<td>YMax (deg)</td>
<td>Maximum of the Y axis of the average stimulus display (in degrees).</td>
</tr>
<tr>
<td>Time Min (sec)</td>
<td>Time minimum in seconds.</td>
</tr>
<tr>
<td>Time Max (sec)</td>
<td>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.</td>
</tr>
<tr>
<td>Time Bin (sec)</td>
<td>Time bin.</td>
</tr>
<tr>
<td>Display</td>
<td>This option specifies what view of the average stimulus will be displayed. See Algorithm below.</td>
</tr>
<tr>
<td>Show Slice At</td>
<td>This option specifies the slice that will be shown. The units and valid range depend on the Display option See Algorithm below.</td>
</tr>
<tr>
<td>Z Min</td>
<td>Color scale minimum.</td>
</tr>
<tr>
<td>Z Max</td>
<td>Color scale maximum.</td>
</tr>
<tr>
<td>Smooth Colors</td>
<td>An option to smooth colors of the average stimulus matrix.</td>
</tr>
<tr>
<td>Matrix Scale</td>
<td>An option on what color scale to use when drawing the matrix.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
<tr>
<td>Save 3D Matrix</td>
<td>An option to save 3D matrix of numerical results in a .csv file. 3D matrix (in X vs. Y display case) is saved starting from the first slice (at Time Min), then the second slice, etc. Within the slice, the first row corresponds to Y Min, the first column corresponds to X Min.</td>
</tr>
<tr>
<td>Save File</td>
<td>The name of the file that will contain 3D matrix data.</td>
</tr>
</tbody>
</table>

**Summary of Numerical Results**
The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>RefCount</td>
<td>Number of reference events (and images used).</td>
</tr>
<tr>
<td>Color Scale Min</td>
<td>Minimum of color scale.</td>
</tr>
<tr>
<td>Color Scale Max</td>
<td>Maximum of color scale.</td>
</tr>
<tr>
<td>Spikes</td>
<td>The number of spikes used in calculation.</td>
</tr>
</tbody>
</table>

**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 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 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
```

Page 103
The computational algorithm works the following way. For each spike of a selected neuron, 
NeuroExplorer identifies the images that precede the spike and are within the specified time interval 
(between Tmin and Tmax; for example, if Tmin = -0.4 and Tmax = 0, NeuroExplorer will analyze all 
the stimuli that were presented up to 400 msec before each spike).

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

Display option identifies what 2-dimensional view of this matrix is presented:
X vs. Y display shows an average stimulus image for the specified time slice
X vs. Time display shows average X pixels of the stimuli for the specified Y value
Y vs. Time display shows average Y pixels of the stimuli for the specified X value

Reference
Izumi Ohzawa, Gregory C. DeAngelis, and Ralph D. Freeman (1996). Encoding of binocular disparity 
2.31. Epoch Counts

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Type</td>
<td>Specifies if the analysis will use a single or multiple reference events.</td>
</tr>
<tr>
<td>Reference</td>
<td>Specifies a reference neuron or event (or a group of reference neurons or events).</td>
</tr>
<tr>
<td>Epochs</td>
<td>A table of epochs in seconds.</td>
</tr>
<tr>
<td>No Selfcount</td>
<td>An option not to count reference events if the target event is the same as the reference event (prevents a histogram to have a huge peak at zero when calculating PSTH versus itself).</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is From Time Range, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Int. filter type</td>
<td>Specifies if the analysis will use a single or multiple interval filters.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter(s) that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each epoch) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each epoch) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Reference</td>
<td>The name of the reference event.</td>
</tr>
<tr>
<td>NumRefEvents</td>
<td>The number of reference events.</td>
</tr>
<tr>
<td>YMin</td>
<td>Minimum epoch count for this variable.</td>
</tr>
<tr>
<td>YMax</td>
<td>Maximum epoch count for this variable.</td>
</tr>
<tr>
<td>Filter Length</td>
<td>The length of the interval filter.</td>
</tr>
</tbody>
</table>

**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:

   \[ \text{if } d[i] \geq \text{EpochStart}[1] \text{ and } d[i] < \text{EpochEnd}[1] \]
   \[ \text{then } \text{epochcount}[1] = \text{epochcount}[1] + 1 \]

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

   \[ \text{if } d[i] \geq \text{EpochStart}[2] \text{ and } d[i] < \text{EpochEnd}[2] \]
   \[ \text{then } \text{epochcount}[2] = \text{epochcount}[2] + 1 \]

   and so on...
2.32. Coherence Analysis

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

**Parameters**

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

Summary of Numerical Results

The following information is available in the Summary of Numerical Results

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

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:

$$\text{Bin} = 1./(2.^{\text{Maximum\_Frequency}})$$

Number_of_Bins $= 2^{\text{Number\_of\_Frequency\_Values}}$

For two variables X (reference) and Y (target) the following entities are calculated. The time intervals of length Bin*NumberOfBins with specified overlaps are defined. For each interval, rate histograms are calculated and detrended, Hanning window is applied and FFTs are calculated. Then, individual and cross-densities are calculated:

$$P_{xx} = \text{FFT}(X)\cdot\text{Conj}(\text{FFT}(X)),$$

$$P_{yy} = \text{FFT}(Y)\cdot\text{Conj}(\text{FFT}(Y))$$

$$P_{xy} = \text{FFT}(X)\cdot\text{Conj}(\text{FFT}(Y)).$$

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

$$\text{Mean}(P_{xy})^2\cdot\text{Mean}(P_{xy}) / (\text{Mean}(P_{xx})\cdot\text{Mean}(P_{yy})).$$

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

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

Calculation of the Confidence Level

Confidence levels are calculated as described in Kattla and Lowery (2010). The confidence level $Z$ is calculated as
\[ Z = 1 - \text{pow}(\alpha, 1/(w^*L - 1)) \]

where

\[ \text{pow}(x,y) \text{ returns } x \text{ to the power of } y, \]

\[ \alpha = 1 - \text{Confidence Level}^{0.01}, \]

L is the number of overlapped windows

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

**Reference**

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Freq.</td>
<td>Maximum frequency for the spectrograms. If at least one continuous variable is selected, the value of the Maximum Frequency parameter is set as 0.5*Digitizing frequency of the first selected continuous variable. All continuous variables selected for this analysis should have the same digitizing rate.</td>
</tr>
<tr>
<td>Number of Fr. Values</td>
<td>Number of frequency values.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Spectrum units (Raw PSD (Power Spectral Density), Log of PSD).</td>
</tr>
<tr>
<td>Start</td>
<td>Start of the first window.</td>
</tr>
<tr>
<td>Shift</td>
<td>How much sliding window is shifted each time.</td>
</tr>
<tr>
<td>Number of Shifts</td>
<td>total number of sliding windows.</td>
</tr>
<tr>
<td>Show Freq. From</td>
<td>An option to show a subset of frequencies. Specifies minimum frequency to be displayed.</td>
</tr>
<tr>
<td>Show Freq. To</td>
<td>An option to show a subset of frequencies. Specifies maximum frequency to be displayed.</td>
</tr>
<tr>
<td>X Axis</td>
<td>Allows to specify what values are shown in the sliding window X axis. There are two options: Start of Window and Center of Window.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the spectrum after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a center of each frequency bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin right</td>
<td>An option to add an additional vector (containing a right edge of each frequency bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Send to Matlab</td>
<td>An option to send the matrix of numerical results to Matlab. See also Matlab Options.</td>
</tr>
<tr>
<td>Matrix Name</td>
<td>Specifies the name of the results matrix in Matlab workspace.</td>
</tr>
<tr>
<td>Matlab command</td>
<td>Specifies a Matlab command that is executed after the numerical results are sent to Matlab.</td>
</tr>
<tr>
<td>Send to Excel</td>
<td>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</td>
</tr>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>

**Summary of Numerical Results**

The following information is available in the Summary of Numerical Results:

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

**Algorithm**

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

**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:

\[ \text{power_spectrum}[i] = 10.*\log10(\text{raw_spectrum}[i]) \]
2.34. Perievent Spectrograms

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>Reference event.</td>
</tr>
<tr>
<td>Max. Freq.</td>
<td>Maximum frequency for the spectrograms. If at least one continuous variable is selected, the value of the Maximum Frequency parameter is set as 0.5*Digitizing frequency of the first selected continuous variable. All continuous variables selected for this analysis should have the same digitizing rate.</td>
</tr>
<tr>
<td>Number of Fr. Values</td>
<td>Number of frequency values.</td>
</tr>
<tr>
<td>Normalization</td>
<td>Spectrum units (Raw PSD (Power Spectral Density), Log of PSD).</td>
</tr>
<tr>
<td>Start</td>
<td>Start of the first window (relative to the reference event).</td>
</tr>
<tr>
<td>Shift</td>
<td>How much sliding window is shifted each time.</td>
</tr>
<tr>
<td>Number of Shifts</td>
<td>Total number of sliding windows.</td>
</tr>
<tr>
<td>Show Freq. From</td>
<td>An option to show a subset of frequencies. Specifies minimum frequency to be displayed.</td>
</tr>
<tr>
<td>Show Freq. To</td>
<td>An option to show a subset of frequencies. Specifies maximum frequency to be displayed.</td>
</tr>
<tr>
<td>X Axis</td>
<td>Allows to specify what values are shown in the sliding window X axis. There are two options: Start of Window and Center of Window. Suppose your data has the strongest spectral value at 50 seconds. This means that the window that has 50 seconds as its center will show the highest spectrum values. If the window width is 20 seconds, it will be the window [40,60]. If you are using 'Start of window' option, you will see the peak at 40 instead of 50 seconds. However, with 'Center of window' option, the peak will be at 50 seconds.</td>
</tr>
<tr>
<td>Select Data</td>
<td>If Select Data is <em>From Time Range</em>, only the data from the specified (by Select Data From and Select Data To parameters) time range will be used in analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Select Data From</td>
<td>Start of the time range in seconds.</td>
</tr>
<tr>
<td>Select Data To</td>
<td>End of the time range in seconds.</td>
</tr>
<tr>
<td>Interval filter</td>
<td>Specifies the interval filter that will be used to preselect data before analysis. See also Data Selection Options.</td>
</tr>
<tr>
<td>Smooth</td>
<td>Option to smooth the spectrum after the calculation. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Smooth Filter Width</td>
<td>The width of the smooth filter. See Post-Processing Options for details.</td>
</tr>
<tr>
<td>Add to Results / Bin left</td>
<td>An option to add an additional vector (containing a left edge of each frequency bin) to the matrix of numerical results.</td>
</tr>
<tr>
<td>Add to Results / Bin middle</td>
<td>An option to add an additional vector (containing a center of each frequency bin) to the matrix of numerical results.</td>
</tr>
</tbody>
</table>
Add to Results / Bin right
An option to add an additional vector (containing a right edge of each frequency bin) to the matrix of numerical results.

Send to Matlab
An option to send the matrix of numerical results to Matlab. See also Matlab Options.

Matrix Name
Specifies the name of the results matrix in Matlab workspace.

Matlab command
Specifies a Matlab command that is executed after the numerical results are sent to Matlab.

Send to Excel
An option to send numerical results or summary of numerical results to Excel. See also Excel Options.

Sheet Name
The name of the worksheet in Excel where to copy the numerical results.

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

Summary of Numerical Results
The following information is available in the Summary of Numerical Results

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

Algorithm
For each selected variable, this analysis calculates multiple spectrograms that start at the specified time after each occurrence of the Reference event. These spectrograms are then averaged over all the reference events. Spectrogram calculation algorithm is described in Spectrogram Analysis.
2.35. Joint ISI Distribution

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

**Parameters**

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

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Color Scale Min</td>
<td>Color scale minimum.</td>
</tr>
<tr>
<td>Color Scale Max</td>
<td>Color scale maximum.</td>
</tr>
</tbody>
</table>

Algorithm

If Use Log Bins and Axes option is not selected:

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

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

\[ \text{Interval}_I = t[i] - t[i-1] \]

\[ \text{binX} = (\text{Interval}_I - \text{MinInterval}) / \text{Bin} \]

\[ \text{Interval}_{I+1} = t[i+1] - t[i] \]

\[ \text{binY} = (\text{Interval}_{I+1} - \text{MinInterval}) / \text{Bin} \]

Then, the corresponding bin count is incremented:

\[ \text{binCounts}[\text{binX}, \text{binY}] = \text{binCounts}[\text{binX}, \text{binY}] + 1 \]

The graph shows binCounts matrix values using color scale.

If Use Log Bins and Axes option is selected:

The i-th bin (i=1,2,...) on each axis is \([\text{IntMin} \times 10^{((i-1)/D)}, \text{IntMin} \times 10^{(i/D)})\), where D is Number of Bins Per Decade. binX and binY are calculated accordingly:

\[ \text{binX} = (\log_{10}(\text{Interval}_I) - \log_{10}(\text{MinInterval})) \times \text{NumBinsPerDecade} \]

Reference

For discussion on using logarithm of interspike intervals, see:

### 2.36. Autocorrelograms Versus Time

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

#### Parameters

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

<table>
<thead>
<tr>
<th>Send to Excel</th>
<th>An option to send numerical results or summary of numerical results to Excel. See also Excel Options.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sheet Name</td>
<td>The name of the worksheet in Excel where to copy the numerical results.</td>
</tr>
<tr>
<td>TopLeft</td>
<td>Specifies the Excel cell where the results are copied. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
</tbody>
</table>

### Summary of Numerical Results

The following information is available in the Summary of Numerical Results

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable name.</td>
</tr>
<tr>
<td>YMin</td>
<td>Y axis minimum.</td>
</tr>
<tr>
<td>YMax</td>
<td>Y axis maximum.</td>
</tr>
<tr>
<td>Color Scale Min</td>
<td>Color scale minimum.</td>
</tr>
<tr>
<td>Color Scale Max</td>
<td>Color scale maximum.</td>
</tr>
<tr>
<td>Spikes</td>
<td>Number of spikes used.</td>
</tr>
</tbody>
</table>

### Algorithm

NeuroExplorer calculates here multiple autocorrelograms (see Autocorrelograms for more information on how each autocorrelogram is calculated). Each autocorrelogram is calculated for a different interval (window) in time:

\[
\text{window}_\text{start}[i], \text{window}_\text{end}[i], \ i = 1, \ldots, \ \text{Number of Shifts}.
\]

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

The following rules are used to calculate the windows:

\[
\begin{align*}
\text{window}_\text{start}[1] &= \text{Start} \\
\text{window}_\text{end}[1] &= \text{Start} + \text{Duration} \\
\text{window}_\text{start}[2] &= \text{Start} + \text{Shift} \\
\text{window}_\text{end}[2] &= \text{Start} + \text{Shift} + \text{Duration} \\
\ldots
\end{align*}
\]
3. Working with Graphics

Graphical analysis results in NeuroExplorer can be adjusted by the user to include custom labels, lines and arrows. The topics in this section describe the graphics elements, list the attributes that you can use with them, and show how to customize NeuroExplorer graphics output.

3.1. 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 above 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):

3.2. 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 Graphics | Edit Graphics | Graph Layout Mode and Graphics | Edit Graphics | Page Layout Mode menu commands to switch the graphics mode in NeuroExplorer. You can also use the following toolbar buttons to switch the graphics 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 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:
3.3. Positioning the 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 relative to its parent. When you move or resize the owner, other objects will move together with their parent.

There are several types of coordinates 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 owner, 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 on 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 on 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:
3.4. Text Labels

To add a label in the NeuroExplorer Graph Window:

- Select **Insert | Text** menu command
- Move the 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).

### 3.5. Lines

To add a line in the NeuroExplorer Graph Window:

- Select **Insert | Line** menu command
- Move the 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 mouse button
To edit the properties of the line, double-click at the line. The following dialog will be displayed:

![Line Edit Dialog](image)

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.

### 3.6. 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 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.
4. Working with 3D Graphics

NeuroExplorer 3D Graphics Module 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 or 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:
4.1. Viewing Multiple Histograms in 3D

To view a set of histograms in 3 dimensions:

- Calculate the histograms by applying any of the histogram analyses (Rate Histograms, Interspike Interval Histograms, Auto- and Cross-correlograms, Perievent Histograms, etc.)
- 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:

\[ z = \frac{\text{histogram_value} - \text{histogram_mean}}{\text{histogram_standard_deviation}} \]

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.
4.2. 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 3DView | 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 - overwritten 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.
4.3. Viewing the Neuronal Activity "Movie"

To view a 'movie' of neuronal activity:

- 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.

- Press Positions button in the Variables sheet of the Data View:

  ![Image of Neuron Position Dialog Box]

- 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:

  ![Image of 3D Neuron Positions]

- Select 3D | Activity Animation menu command.
- Use the buttons in the Properties Panel 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**.
4.4. 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 3DView | **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 peri-event 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).
5. Programming with NexScript

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:

NexScript editor has two docked panels: Functions Panel and Output Panel. In the Functions Panel, you can click at the function name to display function description. You can also double-click at the function name to insert the function into the script. The Output Panel displays compiler messages and debug output (generated using Trace function).

**Scripts**

Each script is a text file that has an extension .nsc. Scripts are usually saved in Windows Application Data directory:

```
C:\Documents and Settings\All Users\Application Data\Nex Technologies\NeuroExplorer\Scripts
```

(when running under Windows XP), or

```
C:\Program Data\Nex Technologies\NeuroExplorer\Scripts
```

(when running under Vista or Windows 7).

Note that you can create subfolders in the scripts folder. The scripts tree shown in the Scripts View is a copy of the scripts directory specified above. You can create subfolders in your scripts directory and then NeuroExplorer will allow you to navigate through the scripts directory tree within the Scripts View.

**Lines and Comments**

Each line of the script may contain only one statement, for example:
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:

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

you can write:

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

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

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

**Variables and Expressions**

NexScript supports numeric variables and standard expressions:

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

**strings:**

```plaintext
name = "SPK" + "01"
```

You can also query and modify file variables (spike trains, intervals, continuous variables, etc.).

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:

```plaintext
imax = 0
doc = GetActiveDocument()
for i = 2 to n
    interval = doc.SPK01a[i] - doc.SPK01a[i-1]
    if interval > imax
        imax = interval
    end
end
```

See Flow Control for more information.

**Functions**

NexScript offers more than 140 functions that allow you to edit data, open and close files, perform analyses, save the results and send results to Matlab of Excel. See Functions for more information.
5.1. Script Variables

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

The following names are valid:
Neuron01 Bar_press Nr_1a SPK02b

These variable names cannot be used in NexScript:
2Neuron Bar-press

The variables from the opened data file can be accessed using the prefix specifying the document. For example, the spike train SPK01a from the active document can be addressed as:
doc.SPK01a

where doc is a reference to the document. You can get this reference by calling GetActiveDocument or calling OpenDocument.

An alternative way to access file variable is by getting a reference to the variable:
SPK01a = GetVarByName(doc, "SPK01a")

See Also File Variables

Variable Types
NexScript supports numeric variables:
xmean = (xmax - xmin)/2.

and strings:
name = "SPK" + "01"

A variable can also be a reference to the existing variable in the file:
neuron1 = GetVar(doc, 1, "neuron")
or a reference to the opened document:
doc = GetActiveDocument()

A variable type can be changed if the right-hand side of the assignment has a different type. For example:
x = 0.005 % x now is a numeric variable
x = "SPK" % after this statement, x is a string

Global Variables
A variable can be declared global so that it can be accessed from several scripts:
Global name

Global statements should be placed at the beginning of the script.

See Also File Variables
5.2. File Variables

The variables from the opened data file can be accessed using the prefix specifying the document. For example, the spike train SPK01a from the active document can be addressed as:

```python
doc.SPK01a
```

where `doc` is a reference to the document. You can get this reference by calling `GetActiveDocument` or calling `OpenDocument`.

An alternative way to access a file variable is by getting a reference to the variable:

```python
spikeTrain = GetVarByName(doc, "SPK01a")
```

Yet another way to access a file variable is by getting a reference to the variable via `doc["VarName"]` notation:

```python
spikeTrainA = doc["SPK01a"]
name = "SPK01b"
spikeTrainB = doc[name]
```

Access to the Data in File Variables

**Timestamped Variables**

You can access the timestamp value by specifying the timestamp index (1-based, i.e. the first timestamp has index 1) in square brackets:

```python
doc = GetActiveDocument()
% first option: use doc.VarName notation
timestamp = doc.SPK01a[3]
% second option: get variable by name and then use the variable directly
spikeTrain = GetVarByName(doc, "SPK01a")
timestamp = spikeTrain[3]
% third option: get variable using doc["VarName"] notation and then use the variable directly
spikeTrain1 = doc["SPK01a"]
timestamp = spikeTrain1[3]
```

You can assign a new value for any timestamp in the current file. For example, to assign the value 0.5 (sec) to the third timestamp of the variable SPK01a, you can use this script:

```python
doc = GetActiveDocument()
doc.SPK01a[3] = 0.5
```

You can also add timestamps to a variable using NexScript functions. See Properties of Variables and Adding Data to Variables for details.

**Interval Variables**

`IntVar[i,1]` gives you read/write access to the start of the i-th interval, `IntVar[i,2]` gives you read/write access to the end of the i-th interval.

For example, to assign the value 27.5 seconds to the end of the first interval of interval variable `Frame`, you would use this script:

```python
doc = GetActiveDocument()
doc.Frame[1,2] = 27.5
```

You can also add intervals to an interval variable using NexScript functions. See Properties of Variables and Adding Data to Variables for details.

**Continuous Variables**

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

`ContVar[i,2]` gives you read-write access to the value of the i-th data point.
For example, the following script prints the timestamp and the value of the fifth data point in variable ContChannel01:

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

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

```
doc.ContChannel01[5,2] = 100.
```

You can add new data points to a continuous variable using NexScript functions. See Properties of Variables and Adding Data to Variables for details.

**Marker Variables**

MarkerVar[i,1] gives you read-only access to the timestamp of the i-th marker.

MarkerVar[i,2] gives you read-only access to the value of the first field of the i-th marker. Note that marker field values are stored as strings, so you will need to use StrToNum() function to convert these values to numbers.

For example, the following script prints the timestamp and the first field value of the fifth marker in variable Strobed:

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

**Operations on Variables**

Many operations on data variables are available via Edit | Operations on Data Variables menu command. This menu command opens Operations dialog:

The operation is specified in the top left corner of the dialog. When you select an operation, an operation description is shown in the central panel of the dialog. When you press Run the Operation button, the specified operation is performed and a script line is added to the 'Operations Performed'
/window at the bottom of the dialog. You can save the list of operations in a NexScript file if you press the "Save.." button.

See Also Properties of Variables and Adding Data to Variables
See Also Document Variables and Adding New Variables
See Also Operations on Document Variables

5.3. Expressions

Standard algebraic expressions are supported:

\[ x_{\text{mean}} = \frac{\text{xmax} - \text{xmin}}{2}. \]

Addition operation can also be applied to the strings:

\[ \text{name} = \text{"SPK" + "01"} \]

Logical expressions may be used in if and while statements:

\[ x = 2 \]

\[
\begin{align*}
\text{if } x \geq 2 & \\
& \quad \text{Trace("x is greater or equal to 2")}
\end{align*}
\]

\[
\begin{align*}
\text{if } x > 2 & \\
& \quad \text{Trace("x > 2")}
\end{align*}
\]

\[
\begin{align*}
\text{if } x \leq 2 & \\
& \quad \text{Trace("x <= 2")}
\end{align*}
\]

\[
\begin{align*}
\text{if } x = 2 & \\
& \quad \text{Trace("x equals 2")}
\end{align*}
\]

\[
\begin{align*}
\text{if } x \neq 1 & \\
& \quad \text{Trace("x is not equal to 1")}
\end{align*}
\]

Logical expressions may be combined using logical AND (\&) or logical OR (|) operators:

\[
\begin{align*}
\text{if } (x \geq 2) \& (y < 4) & \\
& \quad \text{Trace("x <=2 and y <4")}
\end{align*}
\]

\[
\begin{align*}
\text{if } (x \geq 2) \mid (y < 4) & \\
& \quad \text{Trace("x <=2 or y <4")}
\end{align*}
\]
5.4. Flow Control

Loops
NexScript supports two types of loops: for loops and while loops.

for loop has the following syntax:

```plaintext
for variable = expression to expression
    statements ...
end
```

Example:

```plaintext
for i = 1 to 10
    SelectVar(doc, i, "neuron")
end
```

while loop has the following syntax:

```plaintext
while logical_expression
    statements ...
end
```

Example:

```plaintext
i = 1
while i < 10
    SelectVar(doc, i, "neuron")
    i = i + 1
end
```

break

`break` statement causes an immediate exit from the loop.

The following loop

```plaintext
for i = 1 to 10
    Trace(i)
    if i == 5
        break
    end
end
```

will produce output: 1 2 3 4 5

continue

`continue` statement returns to the loop’s beginning skipping the statements that follow it.

The following loop

```plaintext
for i = 1 to 5
    if i == 3
        continue
    end
    Trace(i)
end
```

will produce output: 1 2 4 5

return

return statement causes an immediate exit from the script.
Conditional operators

Operator if has the following syntax:

```plaintext
if logical_expression statements ... end
```

or

```plaintext
if logical_expression statements ... else statements ... end
```

Example

```plaintext
% select a variable if it has at least one spike
% otherwise, deselect the variable
if GetVarCount(doc, i, "neuron") > 0
   SelectVar(doc, i, "neuron")
else
   DeselectVar(doc, i, "neuron")
end
```
5.5. Functions
The following function categories are available in NexScript

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File Read and Write Functions</strong></td>
<td>Functions to read data and text files, save data and results.</td>
</tr>
<tr>
<td><strong>Document Properties</strong></td>
<td>Functions to access general document properties such as file name and comment.</td>
</tr>
<tr>
<td><strong>Document Variables and Adding New Variables</strong></td>
<td>Functions to get the number of spike trains, etc. in the document; to get variable by name; to add new variables to the document.</td>
</tr>
<tr>
<td><strong>Selecting Document Variables</strong></td>
<td>Functions to select variables for analysis.</td>
</tr>
<tr>
<td><strong>Properties of Variables and Adding Data to Variables</strong></td>
<td>Functions to get the number of timestamps and data points and add timestamps and data points.</td>
</tr>
<tr>
<td><strong>Analysis Functions</strong></td>
<td>Functions to run analysis templates, modify templates, print results.</td>
</tr>
<tr>
<td><strong>Numerical Results Functions</strong></td>
<td>Functions to query and save numerical results.</td>
</tr>
<tr>
<td><strong>Operations on Document Variables</strong></td>
<td>Functions to modify existing variables (for example, frequency filter) and to run calculations based on the variables (for example, find synchronous spikes).</td>
</tr>
<tr>
<td><strong>Matlab Functions</strong></td>
<td>Functions to send data and results to Matlab and to read data from Matlab.</td>
</tr>
<tr>
<td><strong>Excel Functions</strong></td>
<td>Functions to send data and results to Excel.</td>
</tr>
<tr>
<td><strong>Power Point Functions</strong></td>
<td>Functions to send graphical results to Power Point.</td>
</tr>
<tr>
<td><strong>Running Script Functions</strong></td>
<td>Functions to run another script or to pause script.</td>
</tr>
<tr>
<td><strong>Math Functions</strong></td>
<td>Various math functions (random numbers, bitwise operations, square root, etc.).</td>
</tr>
<tr>
<td><strong>String Functions</strong></td>
<td>Functions related to strings (search in string, substrings, convert numbers to strings, etc.).</td>
</tr>
<tr>
<td><strong>Debug Functions</strong></td>
<td>Functions that are helpful in debugging (for example, print script variables).</td>
</tr>
</tbody>
</table>

See Also
Introduction to NexScript Programming
## 5.5.1. File Read and Write Functions

The following file read and write functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetFileCount</td>
<td>Returns the number of files that match the specified file filter.</td>
</tr>
<tr>
<td>GetFileName</td>
<td>Returns the file name for the specified index after GetFileCount() was called.</td>
</tr>
<tr>
<td>OpenFile</td>
<td>Opens text file using the specified mode, returns file ID.</td>
</tr>
<tr>
<td>CloseFile</td>
<td>Closes the file.</td>
</tr>
<tr>
<td>ReadLine</td>
<td>Reads a line from the specified text file.</td>
</tr>
<tr>
<td>WriteLine</td>
<td>Writes a line of text to the specified file.</td>
</tr>
<tr>
<td>OpenDocument</td>
<td>Opens a data file with the specified path.</td>
</tr>
<tr>
<td>NewDocument</td>
<td>Opens a new document (data file) with the specified timestamp frequency.</td>
</tr>
<tr>
<td>CloseDocument</td>
<td>Closes the specified document.</td>
</tr>
<tr>
<td>SaveDocument</td>
<td>Saves the specified document.</td>
</tr>
<tr>
<td>SaveDocumentAs</td>
<td>Saves the specified document in a file with the specified file path.</td>
</tr>
<tr>
<td>SaveNumResults</td>
<td>Saves the numerical results to a text file with the specified name.</td>
</tr>
<tr>
<td>SaveNumSummary</td>
<td>Saves the summary of numerical results to a text file with the specified name.</td>
</tr>
<tr>
<td>SaveAsTextFile</td>
<td>Saves the document in the text file with the specified file name.</td>
</tr>
<tr>
<td>MergeFiles</td>
<td>Merges the specified files, returns the reference to the merged file.</td>
</tr>
<tr>
<td>ReadBinary</td>
<td>Reads binary value of a specified type.</td>
</tr>
<tr>
<td>FileSeek</td>
<td>Repositions file pointer by the specified offset.</td>
</tr>
<tr>
<td>SelectFile</td>
<td>Returns the path of the file selected in File Open dialog.</td>
</tr>
</tbody>
</table>

### See Also

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.1.1. GetFileCount Function

GetFileCount Function
Returns the number of files that match the file filter.

Syntax
`double GetFileCount(string fileFilter)`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileFilter</td>
<td>string</td>
<td>File filter specification. Can contain wildcards (*). For example, to get all .nex files in folder C:\data, use filter &quot;C:\data*.nex&quot;.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of files that match file filter.

Comments
None

Usage
NexScript

```nекс
% repeat analysis for all .nex files in the folder
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
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.2. GetFileName Function

GetFileName Function
Returns the file name for the specified index after GetFileCount() was called.

Syntax
GetFileName(double index)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>double</td>
<td>Index of the file in the file list created by the last call to GetFileCount.</td>
</tr>
</tbody>
</table>

Returns
Returns the file name (the full path of the file) for the specified index after GetFileCount() was called.

Comments
None

Usage

NexScript
% repeat analysis for all .nex files in the folder
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

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.3. OpenFile Function

OpenFile Function
Opens text file using the specified mode, returns file ID.

Syntax

double OpenFile(string filePath, string mode)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filePath</td>
<td>string</td>
<td>Full path of the file.</td>
</tr>
<tr>
<td>mode</td>
<td>string</td>
<td>File open mode. Can be either &quot;r&quot;, &quot;rt&quot; (read), &quot;w&quot; or &quot;wt&quot; (write) or &quot;a&quot; or &quot;at&quot; (append).</td>
</tr>
</tbody>
</table>

Returns

Returns file ID.

Comments

None

Usage

NexScript

% open a file in read mode
file = OpenFile("C:\Data\parameters.txt", "r")
% read all the lines in the file and print them
if file > 0
    line = "" % make line a string variable
    while ReadLine(file, line) > 0
        Trace(line)
    end
end
CloseFile(file)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.1.4. CloseFile Function

CloseFile Function
Closes the specified file.

Syntax
CloseFile(fileID)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileID</td>
<td>double</td>
<td>File ID received from OpenFile function.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

% open a file in read mode
file = OpenFile("C:\Data\parameters.txt", "r")
% read all the lines in the file and print them
if file > 0
    line = " " % make line a string variable
    while ReadLine(file, line) > 0
        Trace(line)
    end
end
CloseFile(file)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.1.5. ReadLine Function

ReadLine Function
Reads a line from the text file.

Syntax

```nxml
double ReadLine(fileID, string lineString)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileID</td>
<td>double</td>
<td>File ID received from OpenFile function.</td>
</tr>
<tr>
<td>lineString</td>
<td>string</td>
<td>String that receives the text from the file.</td>
</tr>
</tbody>
</table>

Returns

Returns 1 if more text to read is available in the file, otherwise, returns 0.

Comments

None

Usage

NexScript

```nxml
% open a file in read mode
file = OpenFile("C:\Data\parameters.txt", "r")
% read all the lines in the file and print them
if file > 0
   line = " " % make line a string variable
   while ReadLine(file, line) > 0
      Trace(line)
   end
end
CloseFile(file)
end
```

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.1.6. **WriteLine Function**

*WriteLine Function*

Writes a line of text to a text file.

**Syntax**

`WriteLine(fileID, lineString)`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileID</td>
<td>double</td>
<td>File ID received from OpenFile function.</td>
</tr>
<tr>
<td>lineString</td>
<td>string</td>
<td>The string to be written to the file.</td>
</tr>
</tbody>
</table>

**Returns**

None

**Comments**

None

**Usage**

**NexScript**

```nexus
% open a file in write mode
fileID = OpenFile("C:\results.txt", "w")
WriteLine(fileID, "first line")
CloseFile(fileID)
```

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.1.7. OpenDocument Function

OpenDocument Function
Opens a data file with the specified path. Returns a reference to the opened document.

Syntax

dooleanReference OpenDocument(string filePath)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filePath</td>
<td>string</td>
<td>Full path of the file.</td>
</tr>
</tbody>
</table>

Returns

Comments
None

Usage

NexScript
This function can 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.

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.1.8. NewDocument Function

NewDocument Function
Creates a new document (data file) with the specified timestamp frequency.

Syntax
documentReference NewDocument(frequency)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>double</td>
<td>Timestamp frequency.</td>
</tr>
</tbody>
</table>

Returns
Returns the reference to the new document.

Comments
None

Usage

NexScript

doc = NewDocument(25000.)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.1.9. CloseDocument Function

CloseDocument Function
Closes the specified document.

Syntax
CloseDocument(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript
% this script prints the number of variables in the file
doc = GetActiveDocument()
Trace("document contains", GetVarCount(doc, "all"), "variables")
CloseDocument(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.10. SaveDocument Function

SaveDocument Function
Saves the specified document.

Syntax
SaveDocument(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Document is saved as a .nex file.

Usage

NexScript

doc = GetActiveDocument()
SaveDocument(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.11. SaveDocumentAs Function

SaveDocumentAs Function
Saves the specified document (in .nex format) in a file with the specified file path.

Syntax
SaveDocumentAs(doc, filePath)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>filePath</td>
<td>string</td>
<td>File path.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SaveDocumentAs(doc, "C:\Data\file1.nex")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.12. SaveNumResults Function

SaveNumResults Function
Saves the numerical results to a text file with the specified name.

Syntax
SaveNumResults(doc, fileName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>fileName</td>
<td>string</td>
<td>File path for saved results.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript
doc = GetActiveDocument()
SaveNumResults(doc, "C:\Data\res1.txt")

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.1.13. SaveNumSummary Function

**SaveNumSummary Function**
Saves the summary of numerical results to a text file with the specified name.

**Syntax**

```nexscript
SaveNumSummary(doc, filename)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>File path for saved results.</td>
</tr>
</tbody>
</table>

**Returns**
None

**Comments**
None

**Usage**

*NexScript*

```nexscript
doc = GetActiveDocument()
SaveNumSummary(doc, "C:\Data\res1summary.txt")
```

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.1.14. SaveAsTextFile Function

SaveAsTextFile Function
Saves the document in the text file with the specified file name.

Syntax
SaveAsTextFile(doc, filePath)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>filePath</td>
<td>string</td>
<td>Text file path.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
This function uses options that were specified the last time the menu command File | Export Data | As Text File was executed.

Usage
NexScript
doc = GetActiveDocument()
SaveAsTextFile(doc, fileNename)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.15. MergeFiles Function

MergeFiles Function
Opens and merges the specified files, returns the reference to the merged file.

Syntax
`documentReference MergeFiles(name_list, gap)`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name_list</td>
<td>string</td>
<td>The list of files to be merged. File names should be separated by commas.</td>
</tr>
<tr>
<td>gap</td>
<td>number</td>
<td>Gap between the files (in seconds). See comments below.</td>
</tr>
</tbody>
</table>

Returns

Returns the reference to the merged files.

Comments

In merge process, the data (for the same variable) from the second file is appended to the data from the first file. Before appending, the timestamps of the variable in the second file are shifted by the value equal to duration_of_the_first_file + gap. Since the function uses commas as separators between file names, it is assumed that each file name does not contain commas.

Usage

NexScript

```nxml
dataDir = "C:\Data"
file1 = dataDir + "file1.plx"
file2 = dataDir + "file2.plx"
file3 = dataDir + "file3.plx"
mergeList = file1 + "", file2 + ",", file3
doc = MergeFiles(mergeList, 1.)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.1.16. ReadBinary Function

ReadBinary Function
Reads a binary value of a specified type from a file.

Syntax

double ReadBinary(fileID, valueType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileID</td>
<td>double</td>
<td>File ID received from OpenFile function.</td>
</tr>
<tr>
<td>valueType</td>
<td>string</td>
<td>Binary type. Should be &quot;char&quot;, &quot;uchar&quot;, &quot;short&quot;, &quot;ushort&quot;, &quot;int&quot;, &quot;uint&quot;, &quot;int64&quot;, &quot;uint64&quot;, &quot;float&quot; or &quot;double&quot;.</td>
</tr>
</tbody>
</table>

Returns
The value read from the file.

Comments
None

Usage

NexScript

% open binary file in read mode
file = OpenFile("C:\Data\binaryfile.dat", "r")
% read short (2-byte signed) value
shortValue = ReadBinary(file, "short")

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.1.17. FileSeek Function

FileSeek Function
Repositions file pointer by the specified offset.

Syntax
double FileSeek(fileID, offset, type)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileID</td>
<td>double</td>
<td>File ID received from OpenFile function.</td>
</tr>
<tr>
<td>offset</td>
<td>double</td>
<td>Number of bytes to move the file pointer.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Pointer movement mode. Should be &quot;begin&quot;, &quot;current&quot; or &quot;end&quot;.</td>
</tr>
</tbody>
</table>

Returns
The new byte offset from the beginning of the file.

Comments
FileSeek(file,0,"end") returns file size in bytes. FileSeek(file,0,"current") returns current file position.

Usage

NexScript

```
% open binary file in read mode
file = OpenFile("C:\Data\binaryfile.dat", "r")
% get file length
fileLength = FileSeek(file, 0, "end")
% move pointer 4 bytes from the beginning of the file
newPosition = FileSeek(file, 4, "begin")
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.1.18. SelectFile Function

SelectFile Function
Opens File Open dialog and returns the path of the file selected in the dialog.

Syntax
string SelectFile()

Returns
Returns the path of the file selected in File Open dialog.

Comments
None

Usage

NexScript
path = SelectFile()
% path can be empty if the user pressed Cancel in file dialog
if StrLength(path) > 0
  % open file for reading
  file = OpenFile(path, "r")
  line = ""
  % read the first line of the file
  ReadLine(file, line)
  Trace(line)
  CloseFile(file)
end

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.2. Document Properties Functions

The following document properties functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDocPath</td>
<td>Returns the full path of the data file.</td>
</tr>
<tr>
<td>GetDocTitle</td>
<td>Returns the data file name.</td>
</tr>
<tr>
<td>GetTimestampFrequency</td>
<td>Returns the frequency used in the specified file to store the timestamps.</td>
</tr>
<tr>
<td>GetDocEndTime</td>
<td>Returns the maximum timestamp value (in seconds) for all the document variables.</td>
</tr>
<tr>
<td>SetDocEndTime</td>
<td>Sets the length of experimental session (in seconds) for the document.</td>
</tr>
<tr>
<td>GetDocComment</td>
<td>Returns the document comment string.</td>
</tr>
</tbody>
</table>

**See Also**

*Introduction to NexScript Programming*

*NexScript Function Categories*
5.5.2.1. GetDocPath Function

GetDocPath Function
Returns the full path of the data file.

Syntax
string GetDocPath(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
Returns the full path of the data file.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
path = GetDocPath(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.2.2. **GetDocTitle Function**

**GetDocTitle Function**

Returns the data file name.

**Syntax**

```plaintext
string GetDocTitle(doc)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

**Returns**

Returns the data file name. For example, if the document has the path "C:\Data\data1.nex", this function will return "data1.nex"

**Comments**

None

**Usage**

**NexScript**

```plaintext
doc = GetActiveDocument()
fileName = GetDocTitle(doc)
```

**See Also**

*Introduction to NexScript Programming*

*NexScript Function Categories*
5.5.2.3. GetTimestampFrequency Function

GetTimestampFrequency Function
Returns the frequency used in the internal representation of the timestamps.

Syntax

```plaintext
double GetTimestampFrequency(doc)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
Returns the frequency used in the specified file to store the timestamps.

Comments
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./Timestamp_Frequency.

Usage

**NexScript**

```plaintext
doc = GetActiveDocument()
tsFreq = GetTimestampFrequency(doc)
```

See Also

*Introduction to NexScript Programming*
*NexScript Function Categories*
5.5.2.4. GetDocEndTime Function

GetDocEndTime Function
Returns the maximum timestamp value (in seconds) for all the document variables.

Syntax

double GetDocEndTime(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
Returns the maximum timestamp value (in seconds) for all the document variables.

Comments
None

Usage

NexScript

doc = GetActiveDocument()
endTime = GetDocEndTime(doc)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.2.5. SetDocEndTime Function

SetDocEndTime Function
Sets the length of experimental session for the document.

Syntax
SetDocEndTime(doc, endtime)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>endtime</td>
<td>double</td>
<td>Document end time in seconds.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
NeuroExplorer determines document end time when the document is loaded. Then, the duration of experimental session (endTime - startTime) may be used in the calculations of the confidence limits. If your script modifies or adds the variables, you may need to set the document end time directly to maintain correctness of the confidence calculations.

Usage
NexScript
doc = GetActiveDocument()
SetDocEndTime(doc, 1200.3)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.2.6. GetDocComment Function

GetDocComment Function
Returns the document comment string.

Syntax
string GetDocComment(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
Returns the document comment string.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
comment = GetDocComment(doc)
Trace(comment)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3. Document Variables Functions

The following document variables functions are available in NexScript.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetVarCount</td>
<td>Returns the number of variables of the specified type in the file.</td>
</tr>
<tr>
<td>GetVarName</td>
<td>Returns a string -- the name of the variable of the specified type in the file.</td>
</tr>
<tr>
<td>GetVarSpikeCount</td>
<td>Returns the number of timestamps in the variable.</td>
</tr>
<tr>
<td>GetVar</td>
<td>Returns the reference to the specified variable.</td>
</tr>
<tr>
<td>DeleteVar</td>
<td>Deletes the specified variable from the file.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the specified variable from the file.</td>
</tr>
<tr>
<td>GetVarByName</td>
<td>Returns the reference to the variable which has the specified name.</td>
</tr>
<tr>
<td>NewEvent</td>
<td>Creates a new timestamped variable.</td>
</tr>
<tr>
<td>NewIntEvent</td>
<td>Creates a new interval variable.</td>
</tr>
<tr>
<td>NewPopVector</td>
<td>Creates a new population vector.</td>
</tr>
<tr>
<td>GetContNumDataPoints</td>
<td>Returns the number of data points in the continuous variable.</td>
</tr>
<tr>
<td>NewContVar</td>
<td>Creates a new continuous variable.</td>
</tr>
<tr>
<td>CopySelectedVarsToAnotherFile</td>
<td>Copies selected variables from one file to another.</td>
</tr>
</tbody>
</table>

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.3.1. GetVarCount Function

GetVarCount Function
Returns the number of variables of the specified type in the file.

Syntax
double GetVarCount(doc, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>varType</td>
<td>string</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of variables of the specified type in the file.

Comments
None

Usage
NexScript
doc = GetActiveDocument()
% get the number of continuous variables
numContVars = GetVarCount(doc, "continuous")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.2. GetVarName Function

GetVarName Function
Returns the name of the specified variable.

Syntax
string GetVarName(doc, number, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>1-based variable number within the group specified by varType.</td>
</tr>
<tr>
<td>varType</td>
<td>variableReference</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
Returns the name of the specified variable.

Comments
None

Usage

NexScript
doc = GetActiveDocument()
% get the name of the first event variable
name = GetVarName(doc, 1, "event")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.3. GetVarSpikeCount Function

GetVarSpikeCount Function
Returns the number of timestamps in the specified variable.

Syntax
double GetVarSpikeCount(doc, number, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>1-based variable number within the group specified by varType.</td>
</tr>
<tr>
<td>varType</td>
<td>string</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of timestamps in the variable.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
numSpikes = GetVarSpikeCount(doc, 3, "neuron")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.4. GetVar Function

GetVar Function
Returns the reference to the specified variable.

Syntax
variableReference GetVar(doc, number, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>1-based variable number within the group specified by varType.</td>
</tr>
<tr>
<td>varType</td>
<td>string</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
Returns the reference to the specified variable.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
% get the second event variable
event = GetVar(doc, 2, "event")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.5. DeleteVar Function

DeleteVar Function
Deletes the specified variable from the file.

Syntax
DeleteVar(doc, number, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>1-based variable number within the group specified by varType.</td>
</tr>
<tr>
<td>varType</td>
<td>string</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
All the analysis windows are closed before executing this function. Please note that this function produces an immediate result - after the execution of this function, the number of variables of the specified type is reduced by 1. Therefore, you cannot use a simple for loop to delete all the variables of a certain type. You can use while loop as shown in the example script below.

Usage

NexScript

doc = GetActiveDocument()
% delete the first continuous variable
DeleteVar(doc, 1, "continuous")

% delete all waveform variables
% note that we always delete the first variable
while GetVarCount(doc, "wave") > 0
    DeleteVar(doc, 1, "wave")
end

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.6. Delete Function

Delete Function
Deletes the specified variable from the file.

Syntax
Delete(doc, var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
All the analysis windows are closed before executing this function.

Usage
NexScript

```nexit

doc = GetActiveDocument()
var = GetVarByName(doc, "Event04")
Delete(doc, var)
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.7. GetVarByName Function

GetVarByName Function
Returns the reference to the variable which has the specified name.

Syntax
variableReference GetVarByName(doc, name)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Variable name.</td>
</tr>
</tbody>
</table>

Returns
Returns the reference to the variable which has the specified name.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
% get the variable with the name TrialStart
start = GetVarByName(doc, "TrialStart")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.8. NewEvent Function

NewEvent Function
Creates a new timestamped variable.

Syntax
variableReference NewEvent(doc, count)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>count</td>
<td>double</td>
<td>Initial number of the timestamps in the document.</td>
</tr>
</tbody>
</table>

Returns
Returns a reference to the new variable.

Comments
None

Usage

NexScript

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

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.3.9. NewIntEvent Function

NewIntEvent Function
Creates a new interval variable.

Syntax

```
variableReference NewIntEvent(doc, count)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>count</td>
<td>double</td>
<td>Initial number of intervals in the variable. This parameter is optional.</td>
</tr>
</tbody>
</table>

Returns

Returns a reference to the new variable.

Comments

If initial number of intervals is positive, intervals are initialized with values [0, 0.5], [1, 1.5], etc.

Usage

NexScript

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

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

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.3.10. NewPopVector Function

NewPopVector Function
Creates a new population vector.

Syntax
variableReference NewPopVector(doc, type)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>type</td>
<td>double</td>
<td>if type = 0, all weights of the new vector are equal to zero.</td>
</tr>
</tbody>
</table>

Returns
Returns a reference to the new vector.

Comments
None

Usage

NexScript
doc = GetActiveDocument()
doc.neuronAverage = NewPopVector(doc, 1)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.11. GetContNumDataPoints Function

GetContNumDataPoints Function

Returns the number of data points in the continuous variable.

Syntax

double GetContNumDataPoints(var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to a continuous variable.</td>
</tr>
</tbody>
</table>

Returns

Returns the number of data points in the continuous variable.

Comments

None

Usage

NexScript

doc = GetActiveDocument()
var = GetVarByName(doc, "AD_01")
numPoints = GetContNumDataPoints(var)

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.3.12. NewContVar Function

NewContVar Function
Creates a new continuous variable.

Syntax
\[
\text{variableReference NewContVar(doc, frequency, mVmin, mVmax)}
\]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>frequency</td>
<td>double</td>
<td>Specifies the sampling frequency of the new variable (in Hz).</td>
</tr>
<tr>
<td>mVmin</td>
<td>double</td>
<td>Minimum of the values of the new variable (in milliVolts).</td>
</tr>
<tr>
<td>mVmax</td>
<td>double</td>
<td>Maximum of the values of the new variable (in milliVolts).</td>
</tr>
</tbody>
</table>

Returns
Returns a reference to the new variable.

Comments
NeuroExplorer stores the values of continuous variables as scaled 2-byte integers. Specifying minimum and maximum of the variable values helps to determine the correct scaling factor for the new variable.

Usage
NexScript
\[
\begin{align*}
doc & = \text{GetActiveDocument()} \\
freq & = 1000. \\
% & create new variable in the file \\
doc.Temp1 & = \text{NewContVar(doc, 1000., -500., 500.)} \\
% & add the values to the new variable \\
for i & = 1 \text{ to } 10000 \\
\quad % & timestamp \\
\quad ts & = i/freq \\
\quad % & value \\
\quad value & = 500.*\sin(ts) \\
\quad \text{AddContValue(doc.Temp1, ts, value)} \\
end
\end{align*}
\]

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.3.13. CopySelectedVarsToAnotherFile Function

CopySelectedVarsToAnotherFile Function
Copies selected variables from one file to another.

Syntax
CopySelectedVarsToAnotherFile(fromDoc, toDoc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromDoc</td>
<td>documentReference</td>
<td>Reference to the document. The variables are copied from this document.</td>
</tr>
<tr>
<td>toDoc</td>
<td>documentReference</td>
<td>Reference to the document. The variables are copied to this document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

```nexit
fromDoc = OpenDocument("C:\Data\File1.nex")
toDoc = OpenDocument("C:\Data\File2.nex")
% copy all events from fromDoc to toDoc
% first, select all events
SelectAllEvents(fromDoc)
% call CopySelectedVarsToAnotherFile
CopySelectedVarsToAnotherFile(fromDoc, toDoc)
```

See Also
- Introduction to NexScript Programming
- NexScript Function Categories
5.5.4. Variable Selection Functions

The following selection functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isSelected</td>
<td>Returns selection status of the specified variable.</td>
</tr>
<tr>
<td>Select</td>
<td>Selects the specified variable (with variable type and index) for analysis.</td>
</tr>
<tr>
<td>Deselect</td>
<td>Deselects the specified variable (with variable type and index).</td>
</tr>
<tr>
<td>SelectVar</td>
<td>Selects the specified variable for analysis.</td>
</tr>
<tr>
<td>DeselectVar</td>
<td>Deselects the specified variable.</td>
</tr>
<tr>
<td>SelectAll</td>
<td>Selects all variables for analysis.</td>
</tr>
<tr>
<td>DeselectAll</td>
<td>Deselects all variables.</td>
</tr>
<tr>
<td>SelectAllNeurons</td>
<td>Selects all the neuron type variables for analysis.</td>
</tr>
<tr>
<td>SelectAllEvents</td>
<td>Selects all the event type variables for analysis.</td>
</tr>
<tr>
<td>EnableRecalcOnSelChange</td>
<td>Enables recalculation of analyses when the list of selected variables changes.</td>
</tr>
<tr>
<td>DisableRecalcOnSelChange</td>
<td>Disables recalculation of analyses when the list of selected variables changes.</td>
</tr>
</tbody>
</table>

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.4.1. IsSelected Function

IsSelected Function
Returns 1 if the variable var is selected, 0 otherwise.

Syntax
double IsSelected(var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
Returns 1 if the variable var is selected, 0 otherwise.

Comments
Only selected variables used in analysis.

Usage

NexScript
doc = GetActiveDocument()
isSel = IsSelected(doc.Neuron1)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.2. Select Function

Select Function
Selects the specified variable for analysis.

Syntax
Select(doc, var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage
NexScript

doc = GetActiveDocument()
Select(doc, GetVarByName(doc, "Neuron1"))

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.3. Deselect Function

Deselect Function
Deselects the specified variable.

Syntax
Deselect(doc, var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage
NexScript

doc = GetActiveDocument()
% deselect variable Neuro04a from analysis
Deselect(doc, GetVarByName(doc, "Neuro04a"))

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.4. SelectVar Function

SelectVar Function
Selects the specified variable for analysis.

Syntax
SelectVar(doc, number, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>Variable number within the group specified by varType.</td>
</tr>
<tr>
<td>varType</td>
<td>string</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage
NexScript

doc = GetActiveDocument()
% select the second neuron for analysis
SelectVar(doc, 2, "neuron")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.5. DeselectVar Function

DeselectVar Function
Deselects the specified variable.

Syntax
DeselectVar(doc, number, varType)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>Variable number within the group specified by varType.</td>
</tr>
<tr>
<td>varType</td>
<td>string</td>
<td>Variable type. Should be &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot; &quot;wave&quot;, &quot;popvector&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage

NexScript

```
doc = GetActiveDocument()
% deselect the second neuron
DeselectVar(doc, 2, "neuron")
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.4.6. Select Function

Select Function
Selects the specified variable for analysis.

Syntax
Select(doc, var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage

NexScript

doc = GetActiveDocument()
Select(doc, GetVarByName(doc, "Neuron1"))

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.7. Deselect Function

Deselect Function
Deselects the specified variable.

Syntax
Deselect(doc, var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage
NexScript

```nexscript
doc = GetActiveDocument()
% deselect variable Neuro04a from analysis
Deselect(doc, GetVarByName(doc, "Neuro04a"))
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.8. SelectAll Function

SelectAll Function
Selects all the variables for analysis.

Syntax
SelectAll(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage

NexScript
```
doc = GetActiveDocument()
SelectAll(doc)
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.9. DeselectAll Function

DeselectAll Function
Deselects all variables.

Syntax
DeselectAll(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage

NexScript
doc = GetActiveDocument()
DeselectAll(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.10. SelectAllNeurons Function

SelectAllNeurons Function
Selects all neuron type variables for analysis.

Syntax
SelectAllNeurons(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage

NexScript

doc = GetActiveDocument()
SelectAllNeurons(doc)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.4.11. SelectAllEvents Function

SelectAllEvents Function
Selects all event type variables for analysis.

Syntax
SelectAllEvents(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Only selected variables used in analysis. Use select and deselect functions to specify what variables you want to analyze.

Usage

NexScript

doc = GetActiveDocument()
SelectAllEvents(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.12. EnableRecalcOnSelChange Function

EnableRecalcOnSelChange Function
Enables recalculation of analyses when the list of selected variables changes.

Syntax
EnableRecalcOnSelChange ()

Parameters
None

Returns
None

Comments
If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or ApplyTemplate was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This automatic recalculation was enabled by default in NeuroExplorer prior to version 4.095. In versions 4.095 and higher, automatic recalculation on selection change is disabled. This function allows you to enable automatic recalculation on selection change.

Usage
NexScript
EnableRecalcOnSelChange ()

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.4.13. DisableRecalcOnSelChange Function

DisableRecalcOnSelChange Function
Disables recalculation of analyses when the list of selected variables changes.

Syntax
DisableRecalcOnSelChange()

Parameters
None

Returns
None

Comments
If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or ApplyTemplate was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This automatic recalculation was enabled by default in NeuroExplorer prior to version 4.095. In versions 4.095 and higher, automatic recalculation on selection change is disabled. This function allows you to disable automatic recalculation on selection change if you are using version prior to 4.095.

Usage

NexScript
DisableRecalcOnSelChange()

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.5. Properties of Variables Functions

The following functions that can get or change properties of variables are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetName</td>
<td>Returns the name of the variable.</td>
</tr>
<tr>
<td>GetSpikeCount</td>
<td>Returns the number of timestamps in the variable.</td>
</tr>
<tr>
<td>AddTimestamp</td>
<td>Adds a new timestamp to the specified variable.</td>
</tr>
<tr>
<td>SetNeuronType</td>
<td>Changes the type of the timestamp variable to either 'neuron' or 'event'.</td>
</tr>
<tr>
<td>AddContValue</td>
<td>Adds a new data point to the specified continuous variable.</td>
</tr>
<tr>
<td>AddInterval</td>
<td>Adds a new interval to the specified interval variable.</td>
</tr>
</tbody>
</table>

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.5.1. GetName Function

GetName Function
Returns the name of the variable.

Syntax
string GetName(var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
Returns the name of the variable.

Comments
None

Usage
NexScript

```
doc = GetActiveDocument()
% get the first neuron variable
var = GetVar(doc, 1, "neuron")
% get the variable name
name = GetName(var)
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.5.2. GetSpikeCount Function

GetSpikeCount Function
Returns the number of timestamps in the variable.

Syntax
double GetSpikeCount(var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of timestamps in the variable.

Comments
None

Usage
NexScript

```
doc = GetActiveDocument()
count = GetSpikeCount(GetVarByName("Neuron04a"))
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.5.3. AddTimestamp Function

AddTimestamp Function

Adds a new timestamp to the specified event or neuron variable.

Syntax

AddTimestamp(var, timestamp)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the event or neuron variable.</td>
</tr>
<tr>
<td>timestamp</td>
<td>double</td>
<td>New timestamp (in seconds).</td>
</tr>
</tbody>
</table>

Returns

None

Comments

The new timestamp should not be equal to one of the existing timestamps of the specified variable.

Usage

NexScript

doc = GetActiveDocument()
eventVar = GetVarByName(doc, "Event04")
% add timestamp at 1.5 seconds
AddTimestamp(eventVar, 1.5)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.5.4. SetNeuronType Function

SetNeuronType Function
Changes the type of the specified timestamped variable.

Syntax
SetNeuronType(doc, var, type)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>type</td>
<td>double</td>
<td>If type is positive, the variable type is set to ‘neuron’, if type is zero or negative, the variable type is set to ’event’.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Neuron and event types are almost identical. The main difference is that when a data file is opened by NeuroExplorer, all the neuron variables in this file are selected for analysis.

You may need to use this function when creating new neuron variables using NewEvent. NewEvent creates an event variable and the variable type can later be changed to ‘neuron’ using SetNeuronType.

Usage
NexScript

doc = GetActiveDocument()
doc.neuron1 = NewEvent(doc, 0)
SetNeuronType(doc, doc.neuron1, 1)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.5.5. AddContValue Function

AddContValue Function
Adds a new data point to the specified continuous variable

Syntax
AddContValue(var, timestamp, value)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the continuous variable.</td>
</tr>
<tr>
<td>timestamp</td>
<td>double</td>
<td>Timestamps value in seconds.</td>
</tr>
<tr>
<td>value</td>
<td>double</td>
<td>Voltage value in millivolts.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
contVar = GetVarByName(doc, "ContVar1")
% add voltage value of 25.7 mV at the time 100.3 seconds
AddContValue(contVar, 100.3, 25.7)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.5.6. AddInterval Function

AddInterval Function

Adds a new interval to the specified interval variable.

Syntax

AddInterval(var, interval_start, interval_end)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
<tr>
<td>interval_start</td>
<td>double</td>
<td>Start of new interval (in seconds).</td>
</tr>
<tr>
<td>interval_end</td>
<td>double</td>
<td>End of new interval (in seconds).</td>
</tr>
</tbody>
</table>

Returns

None

Comments

The new interval should not overlap with any of the existing intervals of the specified interval variable.

Usage

NexScript

doc = GetActiveDocument()
intervalVar = GetVarByName(doc, "Trials")
% add interval [100s,120s]
AddInterval(intervalVar, 100, 120)

See Also

Introduction to NexScript Programming

NexScript Function Categories
### 5.5.6. Analysis Functions

The following analysis functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyTemplate</td>
<td>Runs the analysis specified in the template.</td>
</tr>
<tr>
<td>ApplyTemplateToWindow</td>
<td>Runs the analysis specified in the template and shows the result in the specified Graph window.</td>
</tr>
<tr>
<td>EnableRecalcOnSelChange</td>
<td>Enables recalculation of analyses when the list of selected variables changes.</td>
</tr>
<tr>
<td>DisableRecalcOnSelChange</td>
<td>Disables recalculation of analyses when the list of selected variables changes.</td>
</tr>
<tr>
<td>PrintGraphics</td>
<td>Prints the contents of the first graphical window of the document.</td>
</tr>
<tr>
<td>Dialog</td>
<td>Shows a dialog allowing to set script parameters.</td>
</tr>
<tr>
<td>ModifyTemplate</td>
<td>Modifies one of the template parameters.</td>
</tr>
<tr>
<td>RecalculateAnalysisInWindow</td>
<td>Forces recalculation of analysis in the specified graphic window.</td>
</tr>
<tr>
<td>SendGraphicsToPowerPoint</td>
<td>Sends the contents of the first graphical window of the document to the specified Power Point presentation.</td>
</tr>
<tr>
<td>SaveGraphics</td>
<td>Saves the graphics to a WMF or PNG file.</td>
</tr>
</tbody>
</table>

See also [Numerical Results functions](#).

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.6.1. ApplyTemplate Function

ApplyTemplate Function
Runs the analysis specified in the analysis template.

Syntax
ApplyTemplate(doc, templateName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentReference</td>
<td>doc</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>string</td>
<td>templateName</td>
<td>The name of the template.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
% run PSTH analysis saved in the template Peril
ApplyTemplate(doc, "Peril")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.6.2. ApplyTemplateToWindow Function

ApplyTemplateToWindow Function
Runs the analysis specified in the template and shows the result in the specified Graph window.

Syntax
ApplyTemplateToWindow(doc, templatename, windownumber)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentReference</td>
<td>doc</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>string</td>
<td>templatename</td>
<td>The name of the template.</td>
</tr>
<tr>
<td>double</td>
<td>windownumber</td>
<td>The number of the graph window of the document. Graph windows are named Graphs1, Graphs2, etc. Thus, if you need to specify window Graphs2, windownumber should be equal to 2. If the Graph window with the specified number does not exist, a new Graph window is created</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
% run PSTH analysis saved in the template Peri1 and show the results in Graph2 window
ApplyTemplate(doc, "Peri1", 2)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.6.3. PrintGraphics Function

PrintGraphics Function
Prints the contents of the first graphical window of the document.

Syntax
PrintGraphics(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

doc = GetActiveDocument()
PrintGraphics(doc)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.6.4. Dialog Function

Dialog Function
Shows a dialog that can be used to specify the script parameters.

Syntax
double Dialog(doc, par1, name1, type1, par2, name2, type2, ...)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document. Could be zero if all the type values are &quot;number&quot; or &quot;string&quot;.</td>
</tr>
<tr>
<td>par1</td>
<td>variable</td>
<td>A variable that will be assigned a value after Dialog exits. The variable should be created before the Dialog function is called</td>
</tr>
<tr>
<td>name1</td>
<td>string</td>
<td>Prompt that will be shown in the dialog.</td>
</tr>
<tr>
<td>type1</td>
<td>string</td>
<td>Parameter type. It should be one of: &quot;number&quot;, &quot;string&quot;, &quot;neuron&quot;, &quot;neuronorevent&quot;, &quot;event&quot;, &quot;interval&quot;, &quot;wave&quot;, &quot;continuous&quot;, &quot;marker&quot; or &quot;all&quot;.</td>
</tr>
</tbody>
</table>

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

Comments
None

Usage

NexScript
% 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 a user can type the new value in the File Filter edit box. If the user presses OK button, the Dialog function returns 1, otherwise, it returns 0.
The following script will allow a user to choose one of the neurons in the active document and select this neuron for analysis:

```plaintext
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)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.6.5. ModifyTemplate Function

ModifyTemplate Function
Modifies one of the template parameters.

Syntax
ModifyTemplate(doc, templateName, paramName, newParValue)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>templateName</td>
<td>string</td>
<td>The name of the template.</td>
</tr>
<tr>
<td>paramName</td>
<td>string</td>
<td>The name of the parameter to be modified.</td>
</tr>
<tr>
<td>newParValue</td>
<td>string</td>
<td>The new value of the parameter (as a string).</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript
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 Properties Panel (e.g. not "Bin", but "Bin (sec)" as in the example above). You can select the parameter name in the left column of Properties Panel and press Ctrl+C to copy the parameter name to the clipboard and then paste the name of the parameter into your script.

If you need to use a numeric value as newparvalue, you need to convert it to string using NumToStr function. For example, if you need to set Select Data From = doc.Start[1]:

```
ModifyTemplate(doc, "Peri", "Select Data From (sec)", NumToStr(doc.Start[1]))
```

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

```
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. 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.
doc = GetActiveDocument()
ModifyTemplate(doc, templateName, paramName, newParValue)

**See Also**

Introduction to NexScript Programming
NexScript Function Categories
5.5.6.6. RecalculateAnalysisInWindow Function

RecalculateAnalysisInWindow Function
Forces recalculation of analysis in the specified graphic window.

Syntax
RecalculateAnalysisInWindow(doc, graphWindowNumber)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>graphWindowNumber</td>
<td>double</td>
<td>Index of the Graph window.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

doc = GetActiveDocument()
RecalculateAnalysisInWindow(doc, 1)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.6.7. EnableRecalcOnSelChange Function

EnableRecalcOnSelChange Function

Enables recalculation of analyses when the list of selected variables changes.

Syntax

EnableRecalcOnSelChange ()

Parameters

None

Returns

None

Comments

If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or ApplyTemplate was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This automatic recalculation was enabled by default in NeuroExplorer prior to version 4.095. In versions 4.095 and higher, automatic recalculation on selection change is disabled. This function allows you to enable automatic recalculation on selection change.

Usage

NexScript

EnableRecalcOnSelChange ()

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.6.8. DisableRecalcOnSelChange Function

DisableRecalcOnSelChange Function
Disables recalculation of analyses when the list of selected variables changes.

Syntax
DisableRecalcOnSelChange()

Parameters
None

Returns
None

Comments
If there is an analysis window open in NeuroExplorer (for example, if an analysis window was open before script began or ApplyTemplate was called in the script) and a list of selected variables changes, NeuroExplorer can automatically recalculate the analysis results. This automatic recalculation was enabled by default in NeuroExplorer prior to version 4.095. In versions 4.095 and higher, automatic recalculation on selection change is disabled. This function allows you to disable automatic recalculation on selection change if you are using version prior to 4.095.

Usage
NexScript
DisableRecalcOnSelChange()

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.6.9. SendGraphicsToPowerPoint Function

SendGraphicsToPowerPoint Function
Sends the contents of the first graphical window of the document to the specified Power Point presentation.

Syntax
SendGraphicsToPowerPoint (doc, presentationPath, slideTitle, comment, addParameters, useBitmap)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>presentationPath</td>
<td>string</td>
<td>Path of the presentation file.</td>
</tr>
<tr>
<td>slideTitle</td>
<td>string</td>
<td>Slide title.</td>
</tr>
<tr>
<td>comment</td>
<td>string</td>
<td>Slide comment. Will be shown below graphics.</td>
</tr>
<tr>
<td>addParameters</td>
<td>double</td>
<td>If 1, add a text box with analysis parameter values.</td>
</tr>
<tr>
<td>useBitmap</td>
<td>double</td>
<td>If 1, transfer graphics as a bitmap, otherwise, transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>graphics as a metafile.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SendGraphicsToPowerPoint(doc, "C:\Data\NexResults.ppt", "Slide 1", "Sample slide", 1, 0)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.6.10. SaveGraphics Function

SaveGraphics Function
Saves the graphics of the first graphics window of the document to a WMF or PNG file.

Syntax
SaveGraphics(doc, filePath, asPng)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>filePath</td>
<td>string</td>
<td>Graphics file path</td>
</tr>
<tr>
<td>asPng</td>
<td>double</td>
<td>If this parameter value is zero, graphics is saved in Windows Metafile format. Otherwise, the graphics is saved in PNG format.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SaveGraphics(doc, "C:\Data\NexGraphics.wmf", 0)

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.7. Numerical Results Functions

The following numerical results functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetNumRes</td>
<td>Returns the value of the specified cell in the Numerical Results Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>GetNumResNCols</td>
<td>Returns the number of columns of the Numerical Results Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>GetNumResNRows</td>
<td>Returns the number of rows of the Numerical Results Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>GetNumResColumnName</td>
<td>Returns the name of the specified column of the Numerical Results Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>SendResultsToExcel</td>
<td>Sends numerical results (of the first graphics window of the document) to Excel.</td>
</tr>
<tr>
<td>GetNumResSummaryNCols</td>
<td>Returns the number of columns of the Numerical Results Summary Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>GetNumResSummaryNRows</td>
<td>Returns the number of rows of the Numerical Results Summary Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>GetNumResSummaryColumnName</td>
<td>Returns the name of the specified column of the Numerical Results Summary Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>GetNumResSummaryData</td>
<td>Returns the value of the specified cell in the Numerical Results Summary Window of the first graphical view of the document.</td>
</tr>
<tr>
<td>SendResultsSummaryToExcel</td>
<td>Sends summary of numerical results (of the first graphics window of the document) to Excel.</td>
</tr>
<tr>
<td>SaveNumResults</td>
<td>Saves the numerical results to a text file with the specified name.</td>
</tr>
<tr>
<td>SaveNumSummary</td>
<td>Saves the summary of numerical results to a text file with the specified name.</td>
</tr>
</tbody>
</table>

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.7.1. GetNumRes Function

GetNumRes Function

Returns the value of the specified cell in the Numerical Results Window of the first graphical view of the document.

Syntax

double GetNumRes(doc, row, col)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>row</td>
<td>double</td>
<td>1-based row number.</td>
</tr>
<tr>
<td>col</td>
<td>double</td>
<td>1-based column number.</td>
</tr>
</tbody>
</table>

Returns

Returns the numeric value of the specified cell in the Numerical Results Window of the first graphical view of the document.

Comments

None

Usage

NexScript

doc = GetActiveDocument()
ApplyTemplate(doc, "PSTH")
% get the value of the second bin of the first histogram
binCount = GetNumRes(doc, 2, 1)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.7.2. GetNumResNCols Function

GetNumResNCols Function

Returns the number of columns of the Numerical Results Window of the first graphical view of the document.

Syntax

double GetNumResNCols(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns

Returns the number of columns of the Numerical Results Window of the first graphical view of the document.

Comments

None

Usage

NexScript

doc = GetActiveDocument()
numCols = GetNumResNCols(doc)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.7.3. GetNumResNRows Function

GetNumResNRows Function
Returns the number of rows of the Numerical Results Window of the first graphical view of the document.

Syntax
double GetNumResNRows(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of rows of the Numerical Results Window of the first graphical view of the document.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
numRows = GetNumResNRows(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.7.4. GetNumResColumnName Function

GetNumResColumnName Function
Returns the name of the specified column of the Numerical Results Window.

Syntax
string GetNumResColumnName(doc, col)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>col</td>
<td>number</td>
<td>1-based column number.</td>
</tr>
</tbody>
</table>

Returns
Returns the name of the column of the Numerical Results Window of the first graphical view of the document.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
% get the name of the second column of the Numerical Results Window
colName = GetNumResColumnName(doc, 2)

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.7.5. SendResultsToExcel Function

**SendResultsToExcel Function**
Sends numerical results (of the first graphics window of the document) to Excel.

**Syntax**

```
SendResultsToExcel(doc, fileName, worksheetName, useFirstEmptyRow, cellName, includeHeader, includeFileName)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>fileName</td>
<td>string</td>
<td>Excel file path.</td>
</tr>
<tr>
<td>worksheetName</td>
<td>string</td>
<td>Excel worksheet name.</td>
</tr>
<tr>
<td>useFirstEmptyRow</td>
<td>double</td>
<td>If 1, NeuroExplorer will ignore <code>cellName</code> parameter and add the results to the first row where the cell in column A is empty. If 0, NeuroExplorer will copy the data starting with the cell specified in <code>cellName</code>.</td>
</tr>
<tr>
<td>cellName</td>
<td>string</td>
<td>Excel cell where to copy the results. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
<tr>
<td>includeHeader</td>
<td>double</td>
<td>If 1, NeuroExplorer will paste column names.</td>
</tr>
<tr>
<td>includeFileName</td>
<td>double</td>
<td>If 1, NeuroExplorer will add a column with the data file name.</td>
</tr>
</tbody>
</table>

**Returns**
None

**Comments**
None

**Usage**

**NexScript**

```
doc = GetActiveDocument()
SendResultsToExcel(doc, "C:\Data\NexResults.xls", "Nex", 0, "A1", 1, 0)
```

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.7.6. GetNumResSummaryNCols Function

GetNumResSummaryNCols Function
Returns the number of columns of the Numerical Results Summary Window of the first graphical view of the document.

Syntax

```nexscript
double GetNumResSummaryNCols(doc)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns

Returns the number of columns of the Numerical Results Summary Window of the first graphical view of the document.

Comments

None

Usage

NexScript

```nexscript
doc = GetActiveDocument()
nCols = GetNumResSummaryNCols(doc)
```

See Also

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.7.7. GetNumResSummaryNRows Function

GetNumResSummaryNRows Function
Returns the number of rows of the Numerical Results Summary Window of the first graphical view of the document.

Syntax
double GetNumResSummaryNRows(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of rows of the Numerical Results Summary Window of the first graphical view of the document.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
nRows = GetNumResSummaryNRows(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.7.8. GetNumResSummaryColumnName Function

GetNumResSummaryColumnName Function
Returns the name of the specified column of the Numerical Results Summary Window of the first graphical view of the document.

Syntax
string GetNumResSummaryColumnName(doc, col)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>col</td>
<td>double</td>
<td>1-based column number.</td>
</tr>
</tbody>
</table>

Returns
Returns the name of the column of the Numerical Results Summary Window of the first graphical view of the document.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
% get the name of the third column in Numerical Results Summary
colName = GetNumResSummaryColumnName(doc, 3)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.7.9. GetNumResSummaryData Function

GetNumResSummaryData Function
Returns the string value of the specified cell in the Numerical Results Summary Window of the first graphical view of the document.

Syntax
string GetNumResSummaryData(doc, row, col)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document</td>
</tr>
<tr>
<td>row</td>
<td>double</td>
<td>1-based row number.</td>
</tr>
<tr>
<td>col</td>
<td>double</td>
<td>1-based column number.</td>
</tr>
</tbody>
</table>

Returns
Returns the string value of the specified cell in the Numerical Results Summary Window of the first graphical view of the document.

Comments
None

Usage
NexScript

```
doc = GetActiveDocument()  % get the value of the cell in row 3, column 2
summaryCellString = GetNumResSummaryData(doc, 3, 2)
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.7.10. SendResultsSummaryToExcel Function

SendResultsSummaryToExcel Function
Sends summary of numerical results (of the first graphics window of the document) to Excel.

Syntax
SendResultsSummaryToExcel(doc, fileName, worksheetName, useFirstEmptyRow, cellName, includeHeader, includeFileName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>fileName</td>
<td>string</td>
<td>Excel file path.</td>
</tr>
<tr>
<td>worksheetName</td>
<td>string</td>
<td>Excel worksheet name.</td>
</tr>
<tr>
<td>useFirstEmptyRow</td>
<td>double</td>
<td>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.</td>
</tr>
<tr>
<td>cellName</td>
<td>string</td>
<td>Excel cell where to paste the results. Should be in the form CR where C is Excel column name, R is the row number. For example, A1 is the top-left cell in the worksheet.</td>
</tr>
<tr>
<td>includeHeader</td>
<td>double</td>
<td>If 1, will paste column names.</td>
</tr>
<tr>
<td>includeFileName</td>
<td>double</td>
<td>If 1, will add a column with the data file name.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript
doc = GetActiveDocument()
SendResultsSummaryToExcel(doc, "C:\Data\res.xls", "FromNex", 0, "A1", 1, 0)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.7.11. SaveNumResults Function

SaveNumResults Function
Saves the numerical results to a text file with the specified name.

Syntax
SaveNumResults(doc, fileName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>fileName</td>
<td>string</td>
<td>File path for saved results.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SaveNumResults(doc, "C:\Data\res1.txt")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.7.12. SaveNumSummary Function

SaveNumSummary Function
Saves the summary of numerical results to a text file with the specified name.

Syntax
SaveNumSummary(doc, filename)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>filename</td>
<td>string</td>
<td>File path for saved results.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SaveNumSummary(doc, "C:\Data\res1summary.txt")

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.8. Operations on Variables Functions

The following operations on variables functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rename</td>
<td>Renames the specified variable.</td>
</tr>
<tr>
<td>Join</td>
<td>Creates the new event that contains both the timestamps of var1 and the timestamps of var2.</td>
</tr>
<tr>
<td>Sync</td>
<td>Creates the new event containing all the timestamps of var1 that are in the intervals ([\text{var2}+\text{from}, \text{var2}+\text{to}]).</td>
</tr>
<tr>
<td>NotSync</td>
<td>Creates the new event containing all the timestamps of var1 that are NOT in the intervals ([\text{var2}+\text{from}, \text{var2}+\text{to}]).</td>
</tr>
<tr>
<td>FirstAfter</td>
<td>Creates the new event containing the first timestamp of var1 in each of the intervals ([\text{var2}+\text{from}, \text{var2}+\text{to}]).</td>
</tr>
<tr>
<td>FirstNAfter</td>
<td>Creates the new event containing the first N timestamps of var1 after each of the var2 timestamps.</td>
</tr>
<tr>
<td>LastBefore</td>
<td>Creates the new event containing the last timestamp of var1 in each of the intervals ([\text{var2}+\text{from}, \text{var2}+\text{to}]).</td>
</tr>
<tr>
<td>IntervalFilter</td>
<td>Creates the new event containing all the timestamps of var that are in the intervals of the intervalVar.</td>
</tr>
<tr>
<td>SelectTrials</td>
<td>Creates the new event containing the specified timestamps of variable var. selectList can contain comma-separated indexes or ranges of timestamps for example: 1,3-5,10.</td>
</tr>
<tr>
<td>SelectRandom</td>
<td>Creates the new event containing randomly selected timestamps of variable var.</td>
</tr>
<tr>
<td>SelectEven</td>
<td>Creates the new event containing even (2nd, 4th, etc.) timestamps of variable var.</td>
</tr>
<tr>
<td>SelectOdd</td>
<td>Creates the new event containing odd (1st, 3rd, etc.) timestamps of variable var.</td>
</tr>
<tr>
<td>ISIFilter</td>
<td>Creates the new event containing timestamps of the variable var that have preceding interspike intervals larger than min ISI.</td>
</tr>
<tr>
<td>FirstInInterval</td>
<td>Creates the new event. For each interval of intervalVar, the first var timestamp in this interval is copied to the result.</td>
</tr>
<tr>
<td>LastInInterval</td>
<td>Creates the new event. For each interval of intervalVar, the last var timestamp in this interval is copied to the result.</td>
</tr>
<tr>
<td>StartOfInterval</td>
<td>Creates the new event. Copies the start of each interval of intervalVar to the result.</td>
</tr>
<tr>
<td>EndOfInterval</td>
<td>Creates the new event. Copies the end of each interval of intervalVar to the result.</td>
</tr>
<tr>
<td>MakeIntervals</td>
<td>Creates new interval variable with intervals ([\text{varTimestamp}+\text{shiftmin}, \text{varTimestamp}+\text{shiftMax}]).</td>
</tr>
<tr>
<td>MakeIntFromStart</td>
<td>Creates new interval variable. For each timestamp ((\text{tstart})) of intStartVar, it looks for the first timestamp ((\text{tend})) of the intEndVar after (\text{tstart}). If (\text{tend}) is before the next timestamp of intStartVar, it adds the interval ([\text{tstart}+\text{shift1}, \text{tend}+\text{shift2}]) to the result.</td>
</tr>
<tr>
<td>MakeIntFromEnd</td>
<td>Creates new interval variable. For each timestamp of the</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>intEndVar (tend)</td>
<td>It looks for the last timestamp (tstart) of the intStartVar before tend. If tstart is after the previous timestamp of intEndVar, it adds the interval [tstart+shift1, tend+shift2] to the result.</td>
</tr>
<tr>
<td>IntOpposite</td>
<td>Creates a new interval variable that contains intervals 'complementary' to the intervals of intervalVar.</td>
</tr>
<tr>
<td>IntOr</td>
<td>Creates a new Interval Variable that contains unions of the intervals of intervalVar1 and intervalVar2.</td>
</tr>
<tr>
<td>IntAnd</td>
<td>Creates a new Interval Variable that contains intersections of the intervals of intervalVar1 and intervalVar2.</td>
</tr>
<tr>
<td>IntSize</td>
<td>Creates a new Interval Variable that contains all of the intervals of intervalVar that have the length which is more or equal to minInt and less than or equal to intMax.</td>
</tr>
<tr>
<td>IntFind</td>
<td>Creates a new Interval Variable. Each interval of intervalVar that contains one or more timestamps of eventVar is copied to the result.</td>
</tr>
<tr>
<td>MarkerExtract</td>
<td>Creates a new event variable based on an existing marker variable.</td>
</tr>
<tr>
<td>Shift</td>
<td>Returns a new variable with all the timestamps of variable var shifted in time by shiftBy seconds.</td>
</tr>
<tr>
<td>NthAfter</td>
<td>Returns the N-th timestamp in var1 after the timestamp in var2.</td>
</tr>
<tr>
<td>PositionSpeed</td>
<td>Calculates position speed from X and Y coordinate variables. Speed is calculated from positions at times (t-deltaT, t+deltaT).</td>
</tr>
<tr>
<td>FilterContinuousVariable</td>
<td>Applies the specified frequency filter to the specified continuous variable.</td>
</tr>
<tr>
<td>LinearCombinationOfContVars</td>
<td>Calculates a linear combination of two continuous variables.</td>
</tr>
<tr>
<td>DecimateContVar</td>
<td>Decimates a continuous variable.</td>
</tr>
</tbody>
</table>

**See Also**

Introduction to NexScript Programming

NexScript Function Categories
5.5.8.1. Rename Function

Rename Function
Renames the specified variable.

Syntax
Rename(doc, var, newName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>newName</td>
<td>string</td>
<td>The new name of the variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Use this function when you need to assign the variable name generated in the script.

Usage
NexScript

doc = GetActiveDocument()
doc.Temp = IntervalFilter(GetVarByName("Neuron04a"), GetVarByName("Trials"))
Rename(doc, doc.Temp, "Neuron04a_filtered")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.2. Join Function

Join Function
Creates the new event that contains the timestamps of the two specified variables.

Syntax
variableReference Join(var1, var2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event that contains both the timestamps of var1 and the timestamps of var2.

Usage

NexScript

doc = GetActiveDocument()
doc.Events4and5 = Join(doc.Event4, doc.Event5)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.3. Sync Function

Sync Function
Creates the new event containing all the timestamps of var1 that are in the intervals [var2+from, var2+to].

Syntax
variableReference Sync(var1, var2, from, to)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>from</td>
<td>double</td>
<td>Offset minimum (seconds).</td>
</tr>
<tr>
<td>to</td>
<td>double</td>
<td>Offset maximum (seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event containing all the timestamps of var1 that are in the intervals [var2+from, var2+to]

Usage

NexScript

doc = GetActiveDocument()
doc.synced_1_and_2 = Sync(doc.Neuron1, doc.Neuron2, -0.01, 0.01)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.4. NotSync Function

NotSync Function
Creates the new event containing all the timestamps of var1 that are NOT in the intervals \([var2+\text{from}, var2+\text{to}]\).

Syntax
variableReference NotSync(var1, var2, from, to)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>from</td>
<td>double</td>
<td>Offset minimum (seconds).</td>
</tr>
<tr>
<td>to</td>
<td>double</td>
<td>Offset maximum (seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
None

Usage
NexScript

```nексскрипт
doc = GetActiveDocument()
doc.notSync_4_and_5 = NotSync(doc.Neuron04, doc.Neuron05, -0.01, 0.01)
```
5.5.8.5. FirstAfter Function

FirstAfter Function
Creates the new event containing the first timestamp of var1 in each of the intervals [var2+from, var2+to].

Syntax

variableReference FirstAfter(var1, var2, from, to)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>from</td>
<td>double</td>
<td>Offset of interval start around var2 timestamps.</td>
</tr>
<tr>
<td>to</td>
<td>double</td>
<td>Offset of interval end around var2 timestamps.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event containing the first timestamp of var1 in each of the intervals [var2+from, var2+to].

Usage

NexScript

doc = GetActiveDocument()
% find the first spike of Neuron1 after each stimulus (in the first second after stimulus)
doc.FirstN1 = FirstAfter(GetVarByName(doc, "Neuron1"), GetVarByName(doc, "Stimulus"), 0, 1)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.8.6. FirstNAfter Function

FirstNAfter Function
Creates the new event containing the first N timestamps of one variable after each of the timestamps of the second variable.

Syntax
variableReference FirstNAfter(var1, var2, count)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>count</td>
<td>double</td>
<td>How many timestamps of var1 (that are after each timestamp of var2) to copy to the result.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new variable.

Comments
None

Usage

NexScript

doc = GetActiveDocument()
doc.first5SpikesAfterStimulus = FirstNAfter(GetVarByName(doc,"Neuron1"),
GetVarByName(doc,"Stimulus"), 5)

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.8.7. LastBefore Function

LastBefore Function
Creates the new event containing the last timestamp of var1 in each of the intervals [var2+from, var2+to].

Syntax

\[
\text{variableReference LastBefore(var1, var2, from, to)}
\]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>from</td>
<td>double</td>
<td>Interval start offset (seconds).</td>
</tr>
<tr>
<td>to</td>
<td>double</td>
<td>Interval end offset (seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event containing the last timestamp of var1 in each of the intervals [var2+from, var2+to].

Usage

NexScript

```nexit
doc = GetActiveDocument()
doc.last = LastBefore(doc.Neuron1, doc.Event4, 0, 5)
```

See Also

Introduction to NexitScript Programming

NexScript Function Categories
5.5.8.8. IntervalFilter Function

IntervalFilter Function
Creates the new event containing all the timestamps of the specified event or neuron variable that are in the intervals of the specified interval variable.

Syntax
variableReference IntervalFilter(var, intervalVar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>intervalVar</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event containing all the timestamps of var that are in the intervals of the intervalVar

Usage
NexScript

```nexit
doc = GetActiveDocument()
doc.filtered = IntervalFilter(GetVarByName(doc, "Neuron1"), GetVarByName(doc, "Trials"))
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.9. SelectTrials Function

SelectTrials Function
Creates the new event containing the specified timestamps of a variable.

Syntax
variableReference SelectTrials(var, selectList)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>selectList</td>
<td>string</td>
<td>A list of comma-separated indexes or ranges of timestamps. for example: 1,3-5,10.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new variable.

Comments
None

Usage

NexScript

doc = GetActiveDocument()
doc.selectedEvents = SelectTrials(GetVarByName("Event04"), "1,5-10")

See Also
Introduction to NexScript Programming
NexScript Function Categories

Page 236
5.5.8.10. SelectRandom Function

SelectRandom Function
Creates the new event containing randomly selected timestamps of the specified variable.

Syntax
variableReference SelectRandom(var, nSelect)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>nSelect</td>
<td>double</td>
<td>Number of timestamps to select.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new event containing randomly selected timestamps of the specified variable.

Comments
None

Usage
NexScript

doc = GetActiveDocument()
doc.randomEvents04 = SelectRandom(GetVarByName(doc, "Event04"), 10)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.11. SelectEven Function

SelectEven Function
Creates the new event containing even (2nd, 4th, etc.) timestamps of the specified variable.

Syntax
variableReference SelectEven(var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new event containing even (2nd, 4th, etc.) timestamps of the specified variable.

Comments
None

Usage
NexScript
doc = GetActiveDocument()
doc.evenEvents04 = SelectEven(GetVarByName(doc, "Event04"))

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.12. SelectOdd Function

SelectOdd Function
Creates the new event containing the odd (1st, 3rd, etc.) timestamps of the specified variable.

Syntax
variableReference SelectOdd(var)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new event containing the odd (1st, 3rd, etc.) timestamps of the specified variable.

Comments
None

Usage

NexScript

doc = GetActiveDocument()
doc.oddEvents04 = SelectOdd(GetVarByName(doc, "Event04"))

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.13. ISIFilter Function

ISIFilter Function
Creates the new event containing the timestamps of the specified variable that have preceding interspike intervals larger than the specified value.

Syntax
variableReference ISIFilter(var, minISI)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>minISI</td>
<td>double</td>
<td>Minimum ISI (in seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event containing timestamps of the variable var that have preceding interspike intervals larger than minISI.

Usage
NexScript

doc = GetActiveDocument()
doc.RemovedSmallISI = ISIFilter(doc.Neuron1, 0.1)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.14. FirstInInterval Function

FirstInInterval Function
Creates the new event. For each interval of the specified interval variable, the first timestamp in this interval is copied to the result.

Syntax
variableReference FirstInInterval(var, intervalVar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the neuron or event variable.</td>
</tr>
<tr>
<td>intervalVar</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new variable.

Comments
Creates the new event. For each interval of intervalVar, the first var timestamp in this interval is copied to the result.

Usage
NexScript
```nxml
doc = GetActiveDocument()
doc.firstInTrial = FirstInInterval(GetVarByName(doc,"Neuron1"), GetVarByName(doc,"Trials"))
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.15. LastInInterval Function

LastInInterval Function
Creates the new event. For each interval of the specified interval variable, the last timestamp in this interval is copied to the result.

Syntax
variableReference LastInInterval(var, intervalVar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>intervalVar</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event. For each interval of intervalVar, the last var timestamp in this interval is copied to the result.

Usage

NexScript

doc = GetActiveDocument()
doc.LastSpikeInTrial = LastInInterval(doc.Neuron1, doc.Trials)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.8.16. StartOfInterval Function

StartOfInterval Function
Creates the new event. Copies the start of each interval of the specified interval variable to the result.

Syntax
variableReference StartOfInterval(intervalVar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalVar</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event. Copies the start of each interval of intervalVar to the result.

Usage

NexScript

doc = GetActiveDocument()
doc.TrialStarts = StartOfInterval(doc.Trial)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.17. EndOfInterval Function

EndOfInterval Function
Creates the new event based on the specified interval variable. Copies the end of each interval of the interval variable to the result.

Syntax

```plaintext
variableReference EndOfInterval(intervalVar)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalVar</td>
<td>double</td>
<td>Reference to an interval variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new event based on the specified interval variable. Copies the end of each interval of the interval variable to the result.

Usage

```plaintext
NexScript

doc = GetActiveDocument()
doc.trialEnds = EndOfInterval(GetVarByName(doc, "Trials"))
```

See Also

* Introduction to NexScript Programming
* NexScript Function Categories
5.5.8.18. MakeIntervals Function

MakeIntervals Function
Creates new interval variable with intervals \([\text{varTimestamp}+\text{shiftmin}, \text{varTimestamp}+\text{shiftMax}]\).

Syntax
`variableReference MakeIntervals(var, shiftMin, shiftMax)`

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>shiftMin</td>
<td>double</td>
<td>Shift minimum in seconds.</td>
</tr>
<tr>
<td>shiftMax</td>
<td>double</td>
<td>Shift maximum in seconds.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates new interval variable with intervals \([\text{varTimestamp}+\text{shiftmin}, \text{varTimestamp}+\text{shiftMax}]\).

Usage

**NexScript**

```nexscript
doc = GetActiveDocument()
doc.IntAroundEvent04 = MakeIntervals(doc.Event04, 0, 2)
```

See Also

* Introduction to NexScript Programming
* NexScript Function Categories
5.5.8.19. MakeIntFromStart Function

MakeIntFromStart Function
Creates new interval variable. For each timestamp $t_{start}$ of intStartVar, it looks for the first timestamp $t_{end}$ of the intEndVar after $t_{start}$. If $t_{end}$ is before the next timestamp of intStartVar, it adds the interval $[t_{start} + \text{shift}1, t_{end} + \text{shift}2]$ to the result.

Syntax

```nexscript
variableReference MakeIntFromStart(intStartVar, intEndVar, shift1, shift2)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intStartVar</td>
<td>variableReference</td>
<td>Reference to the variable</td>
</tr>
<tr>
<td>intEndVar</td>
<td>variableReference</td>
<td>Reference to the variable</td>
</tr>
<tr>
<td>shift1</td>
<td>double</td>
<td>Shift minimum (seconds).</td>
</tr>
<tr>
<td>shift2</td>
<td>double</td>
<td>Shift maximum (seconds).</td>
</tr>
</tbody>
</table>

Returns

Reference to the new variable.

Comments

None

Usage

NexScript

```nexscript
doc = GetActiveDocument()
doc.Int2 = MakeIntFromStart(doc.Event04, doc.Event05, -0.1, 0.1)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.8.20. MakeIntFromEnd Function

MakeIntFromEnd Function
Creates new interval variable. For each timestamp \( t_{end} \) of the intEndVar, it looks for the last timestamp ( \( t_{start} \) ) of the intStartVar before \( t_{end} \). If \( t_{start} \) is after the previous timestamp of intEndVar, it adds the interval \([ t_{start} + \text{shift1}, t_{end} + \text{shift2}]\) to the result.

Syntax
variableReference MakeIntFromEnd(intStartVar, intEndVar, shift1, shift2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intStartVar</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>intEndVar</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>shift1</td>
<td>double</td>
<td>Shift minimum (seconds).</td>
</tr>
<tr>
<td>shift2</td>
<td>double</td>
<td>Shift maximum (seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
None

Usage
NexScript
doc = GetActiveDocument()
doc.Int2 = MakeIntFromEnd(doc.Event04, doc.Event05, -0.1, 0.1)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.21. IntOpposite Function

IntOpposite Function
Creates a new interval variable that contains intervals 'complementary' to the intervals of the specified interval variable.

Syntax
variableReference IntOpposite(doc, intervalVariable)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document</td>
</tr>
<tr>
<td>intervalVariable</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new interval variable.

Comments
Creates a new interval variable that contains intervals 'complementary' to the intervals of intervalVariable.

Usage

NexScript

doc = GetActiveDocument()
doc.OutsideTrials = IntOpposite(doc, doc.Trials)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.22. IntOr Function

IntOr Function
Creates a new Interval Variable that contains unions of the intervals of intervalVar1 and intervalVar2.

Syntax
variableReference IntOr(doc, intervalVar1, intervalVar2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>intervalVar1</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
<tr>
<td>intervalVar2</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new interval variable.

Comments
Creates a new Interval Variable that contains unions of the intervals of intervalVar1 and intervalVar2

Usage
NexScript

```nxml
doc = GetActiveDocument()
doc.Trials1and2 = IntOr(doc, doc.Trials1, doc.Trials2)
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.23. IntAnd Function

IntAnd Function
Creates a new Interval Variable that contains intersections of the intervals of intervalVar1 and intervalVar2.

Syntax
IntAnd(intervalVar1, intervalVar2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalVar1</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
<tr>
<td>intervalVar2</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
</tbody>
</table>

Returns
The reference to the new interval variable.

Comments
Creates a new Interval Variable that contains intersections of the intervals of intervalVar1 and intervalVar2

Usage
NexScript

doc.Int3 = IntAnd(doc.Interval1, doc.Interval2)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.24. IntSize Function

IntSize Function
Creates a new Interval Variable that contains the intervals with the specified length range.

Syntax
variableReference IntSize(intervalVar, minInt, maxInt)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalVar</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
<tr>
<td>minInt</td>
<td>double</td>
<td>Minimum interval length (in seconds).</td>
</tr>
<tr>
<td>maxInt</td>
<td>double</td>
<td>Maximum interval length (in seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new interval variable.

Comments
Creates a new Interval Variable that contains all of the intervals of intervalVar that have the length which is more or equal to minInt and less than or equal to intMax.

Usage
NexScript

doc = GetActiveDocument()
doc.TrialsLessThan10secDuration = IntSize(doc.Trials, 0, 10)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.25. IntFind Function

IntFind Function
Finds all intervals that contain at least one timestamp of the specified event or neuron variable.

Syntax
variableReference IntFind(intervalVar, eventVar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalVar</td>
<td>variableReference</td>
<td>Reference to the interval variable.</td>
</tr>
<tr>
<td>eventVar</td>
<td>variableReference</td>
<td>Reference to the event or neuron variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Creates a new Interval Variable. Each interval of intervalVar that contains one or more timestamps of eventVar is copied to the result.

Usage
NexScript

doc = GetActiveDocument()
doc.Ints1 = IntFind(doc.Interval1, doc.Event4)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.26. MarkerExtract Function

MarkerExtract Function
Creates a new event variable based on existing marker variable.

Syntax
```
variableReference MarkerExtract(doc, MarkerVariableName, ExtractString)
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document</td>
</tr>
<tr>
<td>MarkerVariableName</td>
<td>string</td>
<td>The name of the marker variable.</td>
</tr>
<tr>
<td>ExtractString</td>
<td>string</td>
<td>Extract string. See comments.</td>
</tr>
</tbody>
</table>

Returns
None

Comments

`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:
```
```

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")
```

Usage

NexScript
```
doc = GetActiveDocument()
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.8.27. Shift Function

Shift Function
Returns a new variable with all the timestamps of variable var shifted in time by shiftBy seconds.

Syntax
variableReference Shift(var, shiftBy)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>shiftBy</td>
<td>double</td>
<td>Shift value (in seconds).</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Returns a new variable with all the timestamps of variable var shifted in time by shiftBy seconds.

Usage
NexScript

doc = GetActiveDocument()
doc.Event04Shifted = Shift(doc.Event04, 10)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.28. NthAfter Function

NthAfter Function
Creates the new variable with the N-th timestamp in var1 after each timestamp in var2.

Syntax
variableReference NthAfter(var1, var2, N)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>variableReference</td>
<td>Reference to the variable</td>
</tr>
<tr>
<td>var2</td>
<td>variableReference</td>
<td>Reference to the variable</td>
</tr>
<tr>
<td>N</td>
<td>double</td>
<td>Spike number.</td>
</tr>
</tbody>
</table>

Returns
Reference to the new variable.

Comments
Creates the new variable with the N-th timestamp in var1 after each timestamp in var2.

Usage

NexScript

doc = GetActiveDocument()

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.8.29. **PositionSpeed Function**

**PositionSpeed Function**
Calculates the position speed from X and Y coordinate variables and creates a new continuous variable.

**Syntax**

```nexscript
variableReference PositionSpeed(varX, varY, deltaT, smoothRadius)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>varX</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>varY</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>deltaT</td>
<td>double</td>
<td>Time step for speed calculation.</td>
</tr>
<tr>
<td>smoothRadius</td>
<td>double</td>
<td>Smooth parameter. See Comments.</td>
</tr>
</tbody>
</table>

**Returns**
Reference to the new variable

**Comments**
PositionSpeed operation calculates the scalar speed of a pair of the position variables.
1. First, for each data point of Position variable PosX[T], where T is time, the raw scalar speed is calculated:
   
   \[
   dX = PosX[T + DeltaT] - PosX[T] \\
   dY = PosY[T + DeltaT] - PosY[T] \\
   RawScalarSpeed[T] = \frac{\sqrt{dX^2 + dY^2}}{DeltaT}
   \]

   If there is no data point at time T + DeltaT, a linear interpolation is used to calculate PosX[T + DeltaT] and PosY[T + DeltaT].

2. Second, RawScalarSpeed is smoothed with the Gaussian filter. The parameters of the filter are such that the width (in seconds) of the Gaussian curve at half the height is equal to the value of Smooth parameter. If Smooth = 0, Gaussian filter is not applied.

**Usage**

**NexScript**

```nexscript
doc = GetActiveDocument() 
doc.speed = PositionSpeed(doc.LED1_X, doc.LED1_Y, 0.1, 0.5)
```

**See Also**

*Introduction to NexScript Programming*
*NexScript Function Categories*
5.5.8.30. FilterContinuousVariable Function

FilterContinuousVariable Function
Filters the specified continuous variable using the specified frequency filter.

Syntax

FilterContinuousVariable(doc, contVar, filteredVarName, filterType, filterOrder, freq1, freq2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>contVar</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>filteredVarName</td>
<td>string</td>
<td>The name of the filtered variable.</td>
</tr>
<tr>
<td>filterType</td>
<td>string</td>
<td>The type of the filter. Should be &quot;Lowpass&quot;, &quot;Highpass&quot;, &quot;Bandpass&quot;, &quot;Bandstop&quot; or &quot;Notch&quot;.</td>
</tr>
<tr>
<td>filterOrder</td>
<td>double</td>
<td>The number specifying the filter order. Should be between 3 and 11 inclusive.</td>
</tr>
<tr>
<td>freq1</td>
<td>double</td>
<td>Filter frequency parameter (in Hz). See comments below.</td>
</tr>
<tr>
<td>freq2</td>
<td>double</td>
<td>Filter frequency parameter (in Hz). See comments below.</td>
</tr>
</tbody>
</table>

Returns

None

Comments

If the filter type is Lowpass or Highpass, freq1 is a cutoff frequency and freq2 is not used.
If the filter type is Bandpass or Bandstop, freq1 is a minimum of the frequency range and freq2 is a maximum of the frequency range.
If the filter type is Notch, freq1 is the center of Notch filter and freq2 is the width of the Notch filter.

Usage

NexScript

The following sample script applies Bandpass filter to the variable ContChannel01. The result of filtering will be saved in continuous variable Cont1BandFiltered. The filter order is 5 and the frequency band is from 1000 Hz to 2000 Hz:

```nexus
doc = GetActiveDocument()
var = GetVarByVarName(doc, "ContChannel01")
FilterContinuousVariable(doc, var, "Cont1BandFiltered", "Bandpass", 5, 1000, 2000)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.8.31. LinearCombinationOfContVars Function

LinearCombinationOfContVars Function
Calculates a linear combination of two continuous variables.

Syntax
LinearCombinationOfContVars(doc, resultName, contVar1, coeff1, contVar2, coeff2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>resultName</td>
<td>string</td>
<td>The name of the result.</td>
</tr>
<tr>
<td>contVar1</td>
<td>variableReference</td>
<td>Reference to the first continuous variable.</td>
</tr>
<tr>
<td>coeff1</td>
<td>double</td>
<td>Coefficient for the first continuous variable.</td>
</tr>
<tr>
<td>contVar2</td>
<td>variableReference</td>
<td>Reference to the second continuous variable.</td>
</tr>
<tr>
<td>coeff2</td>
<td>double</td>
<td>Coefficient for the second continuous variable.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
This function calculates a linear combination of two continuous variables. The values of the resulting variable are:

\[
\text{contVar1\_value} \times \text{coeff1} + \text{contVar2\_value} \times \text{coeff2}
\]

Usage
NexScript

doc = GetActiveDocument()
% calculate average of contVar1 and contVar2
LinearCombinationOfContVars(doc, "average", doc.contVar1, 0.5, doc.contVar2, 0.5)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.8.32. DecimateContVar Function

DecimateContVar Function
Decimates a continuous variable.

Syntax
DecimateContVar(doc, resultName, contVar, decimationFactor)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>resultName</td>
<td>string</td>
<td>The name of the decimated continuous variable.</td>
</tr>
<tr>
<td>contVar</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>decimationFactor</td>
<td>double</td>
<td>If decimationFactor = 2, every second data point of contVar is copied to the result, if decimationFactor = 3, every third, etc.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
DecimateContVar(doc, "decimated", doc.contVar1, 10)

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.9. Matlab Functions

The following matlab functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendSelectedVarsToMatlab</td>
<td>Sends the selected variables to Matlab.</td>
</tr>
<tr>
<td>ExecuteMatlabCommand</td>
<td>Sends the string command to Matlab and executes the command in Matlab.</td>
</tr>
<tr>
<td>GetVarFromMatlab</td>
<td>Gets the specified timestamped variable (array of timestamps) from Matlab.</td>
</tr>
<tr>
<td>GetContVarFromMatlab</td>
<td>Imports the specified matrix containing continuous variable data from Matlab.</td>
</tr>
<tr>
<td>GetContVarWithTimestampsFromMatlab</td>
<td>Imports the specified matrix containing continuous variable data from Matlab.</td>
</tr>
<tr>
<td>GetIntervalVarFromMatlab</td>
<td>Imports the specified matrix containing intervals from Matlab.</td>
</tr>
</tbody>
</table>

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.9.1. SendSelectedVarsToMatlab Function

SendSelectedVarsToMatlab Function
Sends selected variables to Matlab.

Syntax
SendSelectedVarsToMatlab(doc)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SendSelectedVarsToMatlab(doc)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.9.2. ExecuteMatlabCommand Function

ExecuteMatlabCommand Function
Sends the string command to Matlab and executes the command in Matlab.

Syntax

ExecuteMatlabCommand(command)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>string</td>
<td>Matlab command to be run.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Any valid Matlab command that you can type at Matlab prompt can be used. For example, you can call a Matlab script or a function.

Usage

NexScript

ExecuteMatlabCommand("x=randn(10,1);plot(x)")

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.9.3. GetVarFromMatlab Function

GetVarFromMatlab Function
Gets the specified neuron or event variable from Matlab.

Syntax
GetVarFromMatlab(doc, varname, isneuron)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>varname</td>
<td>string</td>
<td>The name of the matrix in Matlab workspace. The matrix should be a double matrix with either one row or one column of data containing timestamps in seconds.</td>
</tr>
<tr>
<td>isneuron</td>
<td>double</td>
<td>If isneuron is 1, the imported variable is added to the list of Neurons. Otherwise, the variable is added to the list of events.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

```nexit
doc = GetActiveDocument()
% get the 1-column matrix ev1 and add it to the list of events
GetVarFromMatlab(doc, "ev1", 0)
```

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.9.4. GetContVarFromMatlab Function

GetContVarFromMatlab Function

Imports the specified matrix from Matlab. Each column of the matrix is imported as a continuous variable.

Syntax

GetContVarFromMatlab(doc, MatrixName, TimestampOfFirstValue, TimeStep)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>MatrixName</td>
<td>string</td>
<td>The name of a matrix in Matlab workspace.</td>
</tr>
<tr>
<td>TimestampOfFirstValue</td>
<td>double</td>
<td>The timestamp (in seconds) of the first value of each continuous variable.</td>
</tr>
<tr>
<td>TimeStep</td>
<td>double</td>
<td>Digitizing time step (in seconds) of the imported variables.</td>
</tr>
</tbody>
</table>

Returns

None

Comments

This function adds continuous variables to the specified document. The names of the variables include the MatrixName and the column number.

Usage

NexScript

doc = GetActiveDocument()
% import matrix contData from Matlab. first timestamp is 0, time step is 0.001s.
GetContVarFromMatlab(doc, "contData", 0, 0.001)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.9.5. GetContVarWithTimestampsFromMatlab Function

GetContVarWithTimestampsFromMatlab Function
Imports the specified 2-column matrix containing continuous variable data from Matlab.

Syntax
GetContVarWithTimestampsFromMatlab(doc, MatrixName, UseFirstDeltaAsDigRate)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document</td>
</tr>
<tr>
<td>MatrixName</td>
<td>string</td>
<td>The name of the Matrix in Matlab workspace. The first column of the matrix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>should contain the values of a continuous variable, the second column - the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>corresponding timestamps.</td>
</tr>
<tr>
<td>UseFirstDeltaAsDigRate</td>
<td>double (0 or 1)</td>
<td>If this parameter is positive, the difference between the second and the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>first timestamp is used as the variable digitizing rate.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
This function adds a continuous variable to the specified document. The name of the new variable is the MatrixName.

Usage

NexScript

```
doc = GetActiveDocument()
GetContVarWithTimestampsFromMatlab(doc, MatrixName, UseFirstDeltaAsDigRate)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.9.6. GetIntervalVarFromMatlab Function

GetIntervalVarFromMatlab Function
Imports the specified matrix containing intervals from Matlab.

Syntax
GetIntervalVarFromMatlab(doc, MatrixName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>MatrixName</td>
<td>string</td>
<td>The name of the 2-column matrix in Matlab workspace.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Imports the specified 2-column matrix from Matlab. The first column of the matrix should contain interval start times in seconds, the second column - interval end times in seconds.

Usage
NexScript
doc = GetActiveDocument()
GetIntervalVarFromMatlab(doc, "trials")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.10. Excel Functions

The following Excel functions are available in NexScript

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetExcelCell</td>
<td>Sets the text value of the specified cell in Excel.</td>
</tr>
<tr>
<td>CloseExcelFile</td>
<td>Closes the specified Excel file if the file is open.</td>
</tr>
<tr>
<td>SendResultsToExcel</td>
<td>Sends numerical results (of the first graphics window of the document) to Excel.</td>
</tr>
<tr>
<td>SendResultsSummaryToExcel</td>
<td>Sends summary of numerical results (of the first graphics window of the document) to Excel.</td>
</tr>
</tbody>
</table>

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.10.1. SetExcelCell Function

SetExcelCell Function
Sets the text value of the specified cell in Excel.

Syntax
SetExcelCell(worksheet, cell, text, excelFilePath)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>worksheet</td>
<td>string</td>
<td>The name of the worksheet.</td>
</tr>
<tr>
<td>cell</td>
<td>string</td>
<td>Excel cell specification, Should be in the form CR where C is Excel column name, R is the row number. For example, &quot;A1&quot; is the top-left cell in the worksheet.</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>The text to be copied to the cell.</td>
</tr>
<tr>
<td>excelFilePath</td>
<td>string</td>
<td>Full path of the Excel file. This parameter is optional. See Usage below.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

% this call will open Excel, Excel will create a new workbook (Excel file) and paste the text to the specified cell
SetExcelCell("fromNex", "A1", "cell text")

% this call will set the cell in the specified Excel file
SetExcelCell("fromNex", "A1", "cell text", "C:\Data\Results.xls")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.10.2. CloseExcelFile Function

CloseExcelFile Function
Closes the specified Excel file if the file is open.

Syntax
CloseExcelFile(filePath)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filePath</td>
<td>string</td>
<td>Full path of the Excel file.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

excelFilePath = "C:\Data\NexResults.xls"
SendResultsToExcel(doc, excelFilePath, "Nex", 0, "A1", 1, 0)
CloseExcelFile(excelFilePath)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.11. Power Point Functions
The following power point functions are available in NexScript

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendGraphicsToPowerPoint</td>
<td>Sends analysis graphics to the specified Power Point file.</td>
</tr>
<tr>
<td>ClosePowerPointFile</td>
<td>Closes the specified Power Point file if the file is open.</td>
</tr>
</tbody>
</table>

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.11.1. SendGraphicsToPowerPoint Function

SendGraphicsToPowerPoint Function
Sends the contents of the first graphical window of the document to the specified Power Point presentation.

Syntax
SendGraphicsToPowerPoint (doc, presentationPath, slideTitle, comment, addParameters, useBitmap)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>presentationPath</td>
<td>string</td>
<td>Path of the presentation file.</td>
</tr>
<tr>
<td>slideTitle</td>
<td>string</td>
<td>Slide title.</td>
</tr>
<tr>
<td>comment</td>
<td>string</td>
<td>Slide comment. Will be shown below graphics.</td>
</tr>
<tr>
<td>addParameters</td>
<td>double</td>
<td>If 1, add a text box with analysis parameter values.</td>
</tr>
<tr>
<td>useBitmap</td>
<td>double</td>
<td>If 1, transfer graphics as a bitmap, otherwise, transfer graphics as a metafile.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

doc = GetActiveDocument()
SendGraphicsToPowerPoint(doc, "C:\Data\NexResults.ppt", "Slide 1", "Sample slide", 1, 0)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.11.2. ClosePowerPointFile Function

ClosePowerPointFile Function
Closes the specified Power Point file if the file is open.

Syntax
ClosePowerPointFile(filePath)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filePath</td>
<td>string</td>
<td>Full path of the Power Point file.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

pptFilePath = "C:\Data\NexResults.ppt"
SendGraphicsToPowerPoint(doc, pptFilePath, "Slide 1", "Sample slide", 1, 0)
ClosePowerPointFile(pptFilePath)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.12. Running Script Functions

The following running script functions are available in NexScript

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunScript</td>
<td>Runs the script with the specified file name.</td>
</tr>
<tr>
<td>Sleep</td>
<td>Pauses execution of the script.</td>
</tr>
</tbody>
</table>

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.12.1. RunScript Function

RunScript Function
Runs the script with the specified name.

Syntax
RunScript(scriptName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scriptName</td>
<td>string</td>
<td>Script name.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

RunScript("MyOtherScript")

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.12.2. Sleep Function

Sleep Function
Pauses execution of the script for nms milliseconds.

Syntax
Sleep(nms)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nms</td>
<td>double</td>
<td>The number of milliseconds to pause.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript

% pause for 2 seconds
Sleep(2000)

See Also

Introduction to NexScript Programming
NexScript Function Categories
## 5.5.13. Math Functions

The following math functions are available in NexScript

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>seed</code></td>
<td>Initializes random number generator.</td>
</tr>
<tr>
<td><code>expo</code></td>
<td>Generates a random value exponentially distributed with the specified mean.</td>
</tr>
<tr>
<td><code>floor</code></td>
<td>Returns the largest integer that is less than or equal to the specified number.</td>
</tr>
<tr>
<td><code>ceil</code></td>
<td>Returns the smallest integer that is greater than or equal to the specified number.</td>
</tr>
<tr>
<td><code>round</code></td>
<td>Rounds the specified number to the nearest integer.</td>
</tr>
<tr>
<td><code>abs</code></td>
<td>Returns the absolute value of the specified number.</td>
</tr>
<tr>
<td><code>sqrt</code></td>
<td>Returns square root of the specified number.</td>
</tr>
<tr>
<td><code>pow</code></td>
<td>Returns the specified number raised to the specified power.</td>
</tr>
<tr>
<td><code>exp</code></td>
<td>Returns exponential of the specified number.</td>
</tr>
<tr>
<td><code>min</code></td>
<td>Returns minimum of two numbers.</td>
</tr>
<tr>
<td><code>max</code></td>
<td>Returns maximum two number.</td>
</tr>
<tr>
<td><code>log</code></td>
<td>Returns logarithm of the specified number.</td>
</tr>
<tr>
<td><code>sin</code></td>
<td>Returns sine of the specified number.</td>
</tr>
<tr>
<td><code>cos</code></td>
<td>Returns cosine of the specified number.</td>
</tr>
<tr>
<td><code>tan</code></td>
<td>Returns tangent of the specified number.</td>
</tr>
<tr>
<td><code>acos</code></td>
<td>Returns arccosine of the specified number.</td>
</tr>
<tr>
<td><code>asin</code></td>
<td>Returns arcsine of the specified number.</td>
</tr>
<tr>
<td><code>atan</code></td>
<td>Returns arctangent of the specified number.</td>
</tr>
<tr>
<td><code>RoundToTS</code></td>
<td>Rounds the number to the nearest timestamp value.</td>
</tr>
<tr>
<td><code>GetFirstGE</code></td>
<td>Returns the index of the first timestamp in the specified variable that is greater or equal to the specified number.</td>
</tr>
<tr>
<td><code>GetFirstGT</code></td>
<td>Returns the index of the first timestamp in the specified variable that is greater than the specified number.</td>
</tr>
<tr>
<td><code>GetBinCount</code></td>
<td>Calculates the number of timestamps in neuron the specified time interval.</td>
</tr>
<tr>
<td><code>BitwiseAnd</code></td>
<td>Converts values to unsigned integers and performs bitwise AND operation.</td>
</tr>
<tr>
<td><code>BitwiseOr</code></td>
<td>Converts values to unsigned integers and performs bitwise OR operation.</td>
</tr>
<tr>
<td><code>GetBit</code></td>
<td>Converts the specified number to an unsigned integer and returns the value of the specified bit (1 to 32).</td>
</tr>
</tbody>
</table>
See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.1. seed Function

seed Function
Initializes random number generator.

Syntax
seed(double iseed)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iseed</td>
<td>double</td>
<td>Numeric seed value. Should be a positive integer.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage

NexScript
seed(1717)

See Also
Introdcution to NexScript Programming
NexScript Function Categories
5.5.13.2. expo Function

expo Function
Generates a random value exponentially distributed with the specified mean.

Syntax

double expo(double fmean)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmean</td>
<td>double</td>
<td>Mean of the exponential distribution.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
None

Usage
NexScript

```nexit
y = expo(10)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.3. floor Function

floor Function
Returns the largest integer that is less than or equal to the specified number.

Syntax

double floor(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns the largest integer that is less than or equal to x.

Comments
None

Usage

NexScript

```nексscript
x = 1.7
y = floor(x)
% y now equals to 1
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.4. ceil Function

ceil Function
Returns the smallest integer that is greater than or equal to the specified number.

Syntax

double ceil(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns the smallest integer that is greater than or equal to x.

Comments
None

Usage

NexScript

x = ceil(1.01)
% x now equals to 2

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.5. round Function

round Function
Rounds the specified number to the nearest integer.

Syntax
round(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns the integer that is closest to x

Comments
None

Usage

NexScript

```nексscript
y = round(1.3)
% y now is 1
y = round(1.6)
% y now is 2
```

See Also

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.13.6. abs Function

abs Function
Returns absolute value of the specified number.

Syntax

double abs(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns

Returns absolute value of x.

Comments

None

Usage

NexScript

% this script tests abs function
x = -2
ax = abs(x)
% ax equals to 2

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.7. sqrt Function

sqrt Function
Returns the square root of the specified number.

Syntax
double sqrt(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value (should be non-negative).</td>
</tr>
</tbody>
</table>

Returns
Returns the square root of x.

Comments
None

Usage
NexScript

\[ y = \text{sqrt}(2) \]

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.8. pow Function

pow Function
Returns x raised to the power of y.

Syntax

double pow(double x, double y)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Number to be raised to the specified power.</td>
</tr>
<tr>
<td>y</td>
<td>double</td>
<td>The power.</td>
</tr>
</tbody>
</table>

Returns
Returns x raised to the power of y.

Comments
None

Usage

NexScript

z = pow(2, 3)
% z now equals to 8

See Also

Introduction to NexScript Programming

NexScript Function Categories
5.5.13.9. exp Function

exp Function
Returns exponential of x.

Syntax

double exp(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns exponential of x.

Comments
None

Usage

NexScript

```nexit
y = exp(2.5)
```

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.10. min Function

min Function
Returns the minimum of two numbers.

Syntax
double min(double x, double y)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
<tr>
<td>y</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns minimum of x and y.

Comments
None

Usage
NexScript
x = 4
y = 2
z = min(x, y)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.11. max Function

max Function
Returns maximum of two numbers.

Syntax
max(double x, double y)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
<tr>
<td>y</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns maximum of x and y.

Comments
None

Usage
NexScript

```nexit
x = 7
y = max(x, 5)
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.12. log Function

log Function
Returns logarithm of the specified number.

Syntax
log(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value (the value should be positive).</td>
</tr>
</tbody>
</table>

Returns
Returns logarithm of x.

Comments
None

Usage

NexScript
y = log(2.5)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.13. **sin Function**

**sin Function**
Returns sine of the specified number.

**Syntax**
`sin(double x)`

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value (sine parameter in radians).</td>
</tr>
</tbody>
</table>

**Returns**
Returns the sine of x.

**Comments**
None

**Usage**

**NexScript**

```nextr
x = 10
y = sin(x)
```

**See Also**
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.14. cos Function

**cos Function**
Returns cosine of the specified number.

**Syntax**

```nexscript
double cos(double x)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value (cosine parameter in radians).</td>
</tr>
</tbody>
</table>

**Returns**

Returns cosine of x.

**Comments**

None

**Usage**

**NexScript**

```nexscript
y = cos(0.5)
```

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.13.15. tan Function

**tan Function**

Returns tangent of the specified number.

**Syntax**

\[ \text{tan(double x)} \]

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value (tangent parameter in radians).</td>
</tr>
</tbody>
</table>

**Returns**

Returns tangent of x.

**Comments**

None

**Usage**

**NexScript**

\[ y = \text{tan}(1) \]

**See Also**

Introduction to NexScript Programming

NexScript Function Categories
5.5.13.16. acos Function

acos Function
Returns the arccosine (in radians) of the specified number.

Syntax

double acos(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>double</td>
<td>Numeric value (the value should be from -1 to +1).</td>
</tr>
</tbody>
</table>

Returns
Returns y such that x = cos(y).

Comments
None

Usage

NexScript
x = 0.5
y = acos(x)
% y is 1.047197551

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.17. asin Function

asin Function
Returns the arcsine of the specified number.

Syntax

double asin(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double x</td>
<td>x</td>
<td>Numeric value (the value should be from -1 to +1).</td>
</tr>
</tbody>
</table>

Returns

Returns y such that x = sin(y).

Comments

None

Usage

NexScript

x = 0.5
y = asin(x)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.18. atan Function

atan Function
Returns the arctangent of the specified number.

Syntax
double acos(double x)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>double</td>
<td>x</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Returns y such that x = tan(y).

Comments
None

Usage
NexScript

x = 2
y = atan(x)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.19. RoundToTS Function

RoundToTS Function
Rounds the specified number to the nearest timestamp value.

Syntax

double RoundToTS(doc, time)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>documentReference</td>
<td>Reference to the document.</td>
</tr>
<tr>
<td>time</td>
<td>double</td>
<td>The time value (in seconds) to be rounded.</td>
</tr>
</tbody>
</table>

Returns
The document timestamp value (in seconds) nearest to the specified time.

Comments
None

Usage

NexScript

doc = GetActiveDocument()
ts = RoundToTS(doc, 1.234)

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.20. GetFirstGE Function

GetFirstGE Function
Returns the index of the first timestamp in the specified variable that is greater than or equal to the specified number.

Syntax
double GetFirstGE(var, number)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>Time value in seconds.</td>
</tr>
</tbody>
</table>

Returns
Returns the index of the first timestamp in the specified variable that is greater than or equal to the specified number.

Comments
None

Usage
NexScript
doc = GetActiveDocument()
neuron = GetVarByName(doc, "Neuron04a")
% get the index of the first spike at or after 5.7s
index = GetFirstGE(var, 5.7)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.21. GetFirstGT Function

GetFirstGT Function
Returns the index of the first timestamp in the specified variable that is greater than the specified number.

Syntax
double GetFirstGT(var, number)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to the variable.</td>
</tr>
<tr>
<td>number</td>
<td>double</td>
<td>Time value in seconds.</td>
</tr>
</tbody>
</table>

Returns
Returns the index of the first timestamp in the specified variable that is greater than the specified number.

Comments
None

Usage
NexScript
doc = GetActiveDocument()
neuron = GetVarByName(doc, "Neuron04a")
% get the index of the first spike after 5.7s
index = GetFirstGT(var, 5.7)

See Also
Introduction to NexScript Programming
NexScript Function Categories
### 5.5.13.22. GetBinCount Function

**GetBinCount Function**
Calculates the number of timestamps in the specified time range.

**Syntax**

```nexit
double GetBinCount(var, timeMin, timeMax)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>variableReference</td>
<td>Reference to a variable. Should be a reference to a neuron, event, interval or continuous variable.</td>
</tr>
<tr>
<td>timeMin</td>
<td>double</td>
<td>Time range minimum (in seconds).</td>
</tr>
<tr>
<td>timeMax</td>
<td>double</td>
<td>Time range maximum (in seconds).</td>
</tr>
</tbody>
</table>

**Returns**
The number of timestamps of the specified variable in the specified time range.

**Comments**
None

**Usage**

**NexScript**

```nexit
doc = GetActiveDocument()
neuron = GetVarByName("Neuron04a")
% calculate how many timestamps of Neuron04a are in the interval [5.3s, 10.2s]
count = GetBinCount(neuron, 5.3, 10.2)
```

**See Also**

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.13.23. BitwiseAnd Function

BitwiseAnd Function
Returns the result of the bitwise AND operation.

Syntax
\[
\text{double BitwiseAnd(value1, value2)}
\]

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value1</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
<tr>
<td>value2</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Result of the bitwise AND operation.

Comments
value1 and value2 are converted to integers and then bitwise AND operation is applied to these integers.

Usage

NexScript
\[
x = \text{BitwiseAnd}(7, 1)
\%
x now equals to 1
\]

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.13.24. BitwiseOr Function

BitwiseOr Function
Returns the result of the bitwise OR operation.

Syntax

double BitwiseOr(value1, value2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value1</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
<tr>
<td>value2</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
</tbody>
</table>

Returns
Result of the bitwise OR operation.

Comments
value1 and value2 are converted to integers and then the bitwise OR operation is applied to these integers.

Usage

NexScript

x = BitwiseOr(2, 1)
% x now equals to 3

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.13.25. GetBit Function

GetBit Function
Returns the value of the specified bit (1 to 32).

Syntax

double GetBit(number, bitNumber)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>double</td>
<td>Numeric value.</td>
</tr>
<tr>
<td>bitNumber</td>
<td>double</td>
<td>1-based bin number. 1 is the least significant bit, 32 is the most significant bit.</td>
</tr>
</tbody>
</table>

Returns

Returns the value (0 or 1) of the specified bit.

Comments

The first parameter is converted to an unsigned 32-bit integer and then the bit value of this unsigned integer is returned.

Usage

NexScript

% get the second bit of 3
b2 = GetBit(3, 2)
% b2 is now equal to 1

See Also

Introduction to NexScript Programming

NexScript Function Categories
## 5.5.14. String Functions

The following string functions are available in NexScript

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Left</strong></td>
<td>Extracts substring from the left part of the string, returns string.</td>
</tr>
<tr>
<td><strong>Mid</strong></td>
<td>Extracts substring from the middle of the string.</td>
</tr>
<tr>
<td><strong>Right</strong></td>
<td>Extracts substring from the right part of the string.</td>
</tr>
<tr>
<td><strong>Find</strong></td>
<td>Looks for a substring inside the specified string.</td>
</tr>
<tr>
<td><strong>StrLength</strong></td>
<td>Calculates the number of characters in the string, returns number.</td>
</tr>
<tr>
<td><strong>NumToStr</strong></td>
<td>Converts a number to string using optional format, returns string.</td>
</tr>
<tr>
<td><strong>StrToNum</strong></td>
<td>Converts string to number.</td>
</tr>
<tr>
<td><strong>GetNumFields</strong></td>
<td>Returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas.</td>
</tr>
<tr>
<td><strong>GetField</strong></td>
<td>Returns the field with the specified field index.</td>
</tr>
<tr>
<td><strong>CharToNum</strong></td>
<td>Converts a one-character string to a number (a character’s ASCII code).</td>
</tr>
<tr>
<td><strong>NumToChar</strong></td>
<td>Converts a number to a one-character string containing the character with the ASCII code equal to the number.</td>
</tr>
</tbody>
</table>

### See Also

- Introduction to NexScript Programming
- NexScript Function Categories
5.5.14.1. Left Function

Left Function
Returns a substring that starts at the beginning of the string.

Syntax
string Left(string, nchar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>String parameter.</td>
</tr>
<tr>
<td>nchar</td>
<td>number</td>
<td>Number of characters in the substring.</td>
</tr>
</tbody>
</table>

Returns
Returns a substring that starts at the beginning of the string.

Comments
None

Usage

NexScript
sub = Left("abcdefg", 3)
% sub now is "abc"

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.14.2. Mid Function

Mid Function
Returns the specified substring.

Syntax
string Mid(string, nstartchar, nchar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>String parameter.</td>
</tr>
<tr>
<td>nstartchar</td>
<td>double</td>
<td>1-based index of the start character.</td>
</tr>
<tr>
<td>nchar</td>
<td>double</td>
<td>Number of characters to select.</td>
</tr>
</tbody>
</table>

Returns
Returns the substring that starts at character nstartchar and contains nchar characters.

Comments
None

Usage
NexScript

```
sub = Mid("abcdefg", 2, 3)
% sub now is "cde"
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.14.3. Right Function

Right Function
Returns a substring that ends at the end of the string.

Syntax
string Right(string, nchar)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>String parameter.</td>
</tr>
<tr>
<td>nchar</td>
<td>string</td>
<td>Number of characters in the substring.</td>
</tr>
</tbody>
</table>

Returns
Extracts right nchar characters from string, returns string.

Comments
None

Usage

NexScript

```
sub = Right("abcdefg", 3)
% sub now is "efg"
```

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.14.4. Find Function

Find Function
Looks for a substring inside a specified string.

Syntax

double Find(string1, string2)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>string</td>
<td>The string where we look for a substring.</td>
</tr>
<tr>
<td>string2</td>
<td>string</td>
<td>The substring that we are trying to find.</td>
</tr>
</tbody>
</table>

Returns
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 string2 is not found.

Comments
None

Usage

NexScript

% this script selects only the neurons that have "05" in their name
doc = GetActiveDocument()
DeselectAll(doc)
for i=1 to GetVarCount(doc, "neuron")
    name = GetVarName(doc, i, "neuron")
    if Find(name, "05") > 0
        SelectVar(doc, i, "neuron")
    end
end

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.14.5. StrLength Function

StrLength Function
Calculates the number of characters in the string.

Syntax

double StrLength(string)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string</td>
<td>The string parameter.</td>
</tr>
</tbody>
</table>

Returns

Returns the number of characters in the string.

Comments

None

Usage

NexScript

n = StrLength("abcd")

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.14.6. NumToStr Function

NumToStr Function
Converts number to string using optional format string.

Syntax
string NumToStr(number, formatString)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>double</td>
<td>Number to be converted to string.</td>
</tr>
<tr>
<td>formatString</td>
<td>string</td>
<td>Optional format string (a standard C/C++ format specifier). See, for example, <a href="http://www.cplusplus.com/reference/clibrary/cstdio/printf/">http://www.cplusplus.com/reference/clibrary/cstdio/printf/</a></td>
</tr>
</tbody>
</table>

Returns
Returns string representing the specified number.

Comments
None

Usage

NexScript
For example, to generate strings:
Event001, Event002, ..., Event016
use the following loop
for i=1 to 16
    str = "Event0" + NumToStr(i, "%02.0f")
end

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.14.7. StrToNum Function

StrToNum Function
Converts string to number.

Syntax

double StrToNum(stringRepresentingNumber)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringRepresentingNumber</td>
<td>string</td>
<td>A string containing a valid representation of the number, for example, &quot;1&quot;, &quot;002&quot;, &quot;123.456&quot;.</td>
</tr>
</tbody>
</table>

Returns

Returns the number corresponding to the specified string.

Comments

None

Usage

NexScript

x = StrToNum("003")
% x now equals to 3

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.14.8. GetNumFields Function

GetNumFields Function
Returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas.

Syntax
GetNumFields(stringWithFields)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringWithFields</td>
<td>string</td>
<td>The string containing fields. The field is a substring that does not contain spaces, tabs or commas.</td>
</tr>
</tbody>
</table>

Returns
Returns the number of fields in the string. The field is a substring that does not contain spaces, tabs or commas.

Comments
None

Usage

NexScript
numFields = GetNumFields("One two 3 4")
% numFields is now equal to 4

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.14.9. GetField Function

GetField Function
Returns the string field.

Syntax
string GetField(stringWithFields, fieldnumber)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringWithFields</td>
<td>string</td>
<td>The string containing multiple data fields. The field is a substring that does not contain spaces, tabs or commas.</td>
</tr>
<tr>
<td>fieldnumber</td>
<td>double</td>
<td>The field number.</td>
</tr>
</tbody>
</table>

Returns
Returns the field with the specified number as a string. The field is a substring that does not contain spaces, tabs or commas. For example,
GetField("One two 3 4", 3) returns "3".

Comments
None

Usage
NexScript
secondField = GetField("Neuro04a Neuron05b", 2)
% secondField now equals to "Neuron05b"

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.14.10. CharToNum Function

CharToNum Function
Converts a one-character string to a number (a character's ASCII code).

Syntax

double CharToNum(oneCharString)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oneCharString</td>
<td>string</td>
<td>A string of length 1 (for example, &quot;a&quot;, &quot;5&quot;).</td>
</tr>
</tbody>
</table>

Returns
The string character's ASCII code.

Comments
None

Usage

NexScript

\[ x = \text{CharToNum}("1") \]
\% x now equals to 49

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.14.11. **NumToChar Function**

**NumToChar Function**
Converts a number to a one-character string containing a character with the ASCII code equal to the number.

**Syntax**

```
string NumToChar(number)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>double</td>
<td>ASCII code of the character.</td>
</tr>
</tbody>
</table>

**Returns**

Returns a one-character string containing the character with the ASCII code equal to the specified number.

**Comments**

None

**Usage**

**NexScript**

```nexit
oneAsString = NumToChar(49)
% oneAsString now equals to "1"
```

**See Also**

- *Introduction to NexScript Programming*
- *NexScript Function Categories*
5.5.15. Debug Functions

The following debug functions are available in NexScript:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace</td>
<td>Prints arguments to output window.</td>
</tr>
<tr>
<td>MsgBox</td>
<td>Displays a message box.</td>
</tr>
</tbody>
</table>

See Also

Introduction to NexScript Programming
NexScript Function Categories
5.5.15.1. Trace Function

Trace Function
Prints arguments to the output window.

Syntax
Trace(arg1, arg2, ...)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg1</td>
<td>any type</td>
<td>String, number or any other valid NexScript value.</td>
</tr>
<tr>
<td>arg2</td>
<td>any type</td>
<td>String, number or any other valid NexScript value.</td>
</tr>
</tbody>
</table>

Returns
None

Comments
Converts each parameter to string and the prints the result to the output window.

Usage
NexScript
x = 30
% print the value of x
Trace("x=", x)

See Also
Introduction to NexScript Programming
NexScript Function Categories
5.5.15.2. MsgBox Function

MsgBox Function
This function is equivalent to Trace function.

See Also
Introduction to NexScript Programming
NexScript Function Categories
6. COM/ActiveX Interfaces

NeuroExplorer exposes COM (Component Object Model) interfaces that allow other applications to launch and control NeuroExplorer.

For example, the following Matlab code starts NeuroExplorer, opens a data file and loads Neuron04a timestamps into the Matlab workspace:

```matlab
nenex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
zeuron = doc.Variable('Neuron04a');
timestamps = neuron.Timestamps();
```

The table below lists the objects that NeuroExplorer exposes via COM methods and properties.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Neuroexplorer application interface. Provides methods and properties for the main application (for example, open document).</td>
</tr>
<tr>
<td>Document</td>
<td>Document interface. Provides methods and properties of the document (for example, lists variables in the document).</td>
</tr>
<tr>
<td>Variable</td>
<td>Variable interface. Provides methods and properties of a variable inside a document (for example, gets variable timestamps).</td>
</tr>
</tbody>
</table>
6.1. Application

Application Interface

External applications can open an instance of NeuroExplorer using its Program ID "NeuroExplorer.Application".

For example, to open NeuroExplorer (or to connect to a running instance of NeuroExplorer) in Matlab, you can use the following command:

```matlab
nex = actxserver('NeuroExplorer.Application');
```

The object returned by the actxserver command is an Application object. The properties and methods of this object are listed below.

Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveDocument</td>
<td>Read-only access to the active document</td>
</tr>
<tr>
<td>DocumentCount</td>
<td>The number of open documents</td>
</tr>
<tr>
<td>Version</td>
<td>NeuroExplorer version</td>
</tr>
<tr>
<td>Visible</td>
<td>Controls visibility of the application</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenDocument</td>
<td>Opens a specified document (data file)</td>
</tr>
<tr>
<td>Document</td>
<td>Returns one of the open documents</td>
</tr>
<tr>
<td>Sleep</td>
<td>Pauses the program</td>
</tr>
<tr>
<td>RunNexScript</td>
<td>Runs NexScript saved in a file</td>
</tr>
<tr>
<td>RunNexScriptCommands</td>
<td>Runs the specified NexScript text</td>
</tr>
</tbody>
</table>

See Also

COM/ActiveX Interfaces
6.1.1. ActiveDocument Property

Application.ActiveDocument Property
Read-only property that returns a Document object that represents the active document (the document corresponding to the active window of the application). Returns null if there are no open documents.

Syntax
Document ActiveDocument

Usage
Matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.ActiveDocument;

See Also
Application Interface
Document Interface
COM/ActiveX Interfaces
6.1.2. DocumentCount Property

Application.DocumentCount Property
Read-only property that returns the number of open documents (data files).

Syntax

```matlab
int DocumentCount
```

Usage

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
numDocs = nex.DocumentCount;
```

See Also

Application Interface

COM/ActiveX Interfaces
6.1.3. Version Property

Application.Version Property
Read-only property that returns a string with the current version of NeuroExplorer.

Syntax
string Version

Usage
Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
version = nex.Version;
```

See Also
Application Interface
COM/ActiveX Interfaces
6.1.4. Visible Property

Application.Visible Property
Boolean read/write property that controls the visibility of NeuroExplorer.

Syntax
Boolean Visible

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
isVisible = nex.Visible;
% make sure that NeuroExplorer is visible
nex.Visible = true;

See Also
Application Interface
COM/ActiveX Interfaces
6.1.5. OpenDocument Method

Application.OpenDocument Method
Opens the specified data file. Returns Document object if succeeded.

Syntax
Document OpenDocument ( string documentPath )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentPath</td>
<td>string</td>
<td>Full data file path</td>
</tr>
</tbody>
</table>

Returns
Returns Document object.

Usage
Matlab
```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
```

See Also
Application Interface
Document Interface
COM/ActiveX Interfaces
6.1.6. Document Method

Application.Document Method
Returns Document object for the specified document index.

Syntax
Document Document(int documentIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentIndex</td>
<td>int</td>
<td>1-based document index</td>
</tr>
</tbody>
</table>

Returns
Returns Document object.

Usage
Matlab
nex = actxserver('NeuroExplorer.Application');
% get the first open document
doc = nex.Document(1);

See Also
Application Interface
Document Interface
COM/ActiveX Interfaces
6.1.7. Sleep Method

Application.Sleep Method
Pauses the application.

Syntax
void Sleep ( int millisecondsToSleep )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>millisecondsToSleep</td>
<td>int</td>
<td>Number of milliseconds to sleep</td>
</tr>
</tbody>
</table>

Returns
None.

Usage
Matlab
```
nex = actxserver('NeuroExplorer.Application');
% pause NeuroExplorer for 1 second
nex.Sleep(1000);
```

See Also
Application Interface
COM/ActiveX Interfaces
6.1.8. RunNexScript Method

Application.RunNexScript Method
Runs the specified NexScript. Returns true if succeeded.

Syntax

bool RunNexScript ( string scriptPath )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scriptPath</td>
<td>string</td>
<td>Script path.</td>
</tr>
</tbody>
</table>

Returns

Returns true if script succeeded, otherwise, returns false.

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
res = nex.RunNexScript('C:\Data\Scripts\TestScript.nsc');

See Also

Application Interface
Document Interface
COM/ActiveX Interfaces
6.1.9. RunNexScriptCommands Method

Application.RunNexScriptCommands Method
Runs the specified NexScript text. Returns true if succeeded.

Syntax
bool RunNexScriptCommands ( string script )

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>string</td>
<td>Script text. Script lines should be separated by \n.</td>
</tr>
</tbody>
</table>

Returns
Returns true if script succeeded, otherwise, returns false.

Usage
Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
res = nex.RunNexScriptCommands('doc=GetActiveDocument()
doc.NewEvent = Sync(doc.Neuron04a, doc.Neuron05b, -0.01, 0.01)')
res = nex.RunNexScriptCommands('doc=GetActiveDocument()
ApplyTemplate(doc,"AutoCorrelograms")');
```

See Also
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2. Document

Document Interface

NeuroExplorer Application object provides access to the open documents. For example, to open a
document in NeuroExplorer in Matlab script, you can use:

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
```

Here OpenDocument method returns a Document object corresponding to the specified data file.
The properties and methods of the Document object are listed below.

**Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Document file path</td>
</tr>
<tr>
<td>FileName</td>
<td>Document file name</td>
</tr>
<tr>
<td>Comment</td>
<td>Document comment</td>
</tr>
<tr>
<td>TimestampFrequency</td>
<td>Document timestamp frequency</td>
</tr>
<tr>
<td>StartTime</td>
<td>Document data start time</td>
</tr>
<tr>
<td>EndTime</td>
<td>Document data end time</td>
</tr>
<tr>
<td>VariableCount</td>
<td>The number of variables in the document</td>
</tr>
<tr>
<td>NeuronCount</td>
<td>The number of neuron variables in the document</td>
</tr>
<tr>
<td>EventCount</td>
<td>The number of event variables in the document</td>
</tr>
<tr>
<td>IntervalCount</td>
<td>The number of interval variables in the document</td>
</tr>
<tr>
<td>MarkerCount</td>
<td>The number of marker variables in the document</td>
</tr>
<tr>
<td>WaveCount</td>
<td>The number of waveform variables in the document</td>
</tr>
<tr>
<td>ContinuousCount</td>
<td>The number of continuous variables in the document</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Returns the specified variable</td>
</tr>
<tr>
<td>Neuron</td>
<td>Returns the specified neuron variable</td>
</tr>
<tr>
<td>Event</td>
<td>Returns the specified event variable</td>
</tr>
<tr>
<td>Marker</td>
<td>Returns the specified marker variable</td>
</tr>
<tr>
<td>Interval</td>
<td>Returns the specified interval variable</td>
</tr>
<tr>
<td>Wave</td>
<td>Returns the specified waveform variable</td>
</tr>
<tr>
<td>Continuous</td>
<td>Returns the specified continuous variable</td>
</tr>
<tr>
<td>DeselectAll</td>
<td>Deselects all the data variables in the document</td>
</tr>
<tr>
<td>SelectAllNeurons</td>
<td>Selects all the neuron variables in the document</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SelectAllContinuous</td>
<td>Selects all the continuous variables in the document</td>
</tr>
<tr>
<td>ApplyTemplate</td>
<td>Runs the analysis specified in the analysis template</td>
</tr>
<tr>
<td>GetNumericalResults</td>
<td>Returns numerical results for the first graph view of the document</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the document</td>
</tr>
</tbody>
</table>

**See Also**

COM/ActiveX Interfaces
6.2.1. Path Property

Document.Path Property
Read-only property that returns a string with the full path of the document.

Syntax
string Path

Usage
Matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
path = doc.Path;

See Also
Document Interface
COM/ActiveX Interfaces
6.2.2. FileName Property

Document.FileName Property
Read-only property that returns a string with the file name.

Syntax

string FileName

Usage

Matlab

nenx = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument ('C:\Data\MyDataFile.nex');
path = doc.Path;
fileName = doc.FileName;
% path is 'C:\Data\MyDataFile.nex'
% fileName is 'MyDataFile.nex'

See Also

Document Interface
COM/ActiveX Interfaces
6.2.3. Comment Property

Document.Comment Property
Read-only property that returns a string with the data file comment.

Syntax

string Comment

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
comment = doc.Comment;

See Also

Document Interface
COM/ActiveX Interfaces
6.2.4. TimestampFrequency Property

Document.TimestampFrequency Property
Read-only property that returns the timestamp frequency of the document in Hertz.

Syntax

double TimestampFrequency

Usage

Matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
tsFrequency = doc.TimestampFrequency;

See Also

Document Interface
COM/ActiveX Interfaces
6.2.5. StartTime Property

Document.StartTime Property
Read-write property that specifies the document data start time (minimum timestamp) in seconds.

Syntax

double StartTime

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
startTime = doc.StartTime;

See Also

Document Interface
COM/ActiveX Interfaces
6.2.6. EndTime Property

Document.EndTime Property
Read-write property that specifies the document data end time (maximum timestamp) in seconds.

Syntax

double EndTime

Usage

Matlab

def = actxserver('NeuroExplorer.Application');  
doc = def.OpenDocument('C:\Data\MyDataFile.nex');  
endTime = doc.EndTime;

See Also

Document Interface
COM/ActiveX Interfaces
6.2.7. VariableCount Property

Document.VariableCount Property
Read-only property that returns the number of variables in the document.

Syntax
int VariableCount

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numVars = doc.VariableCount;

See Also
Document Interface
COM/ActiveX Interfaces
6.2.8. NeuronCount Property

Document.NeuronCount Property
Read-only property that returns the number of neurons in the document.

Syntax
int NeuronCount

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numNeurons = doc.NeuronCount;

See Also
Document Interface
COM/ActiveX Interfaces
6.2.9. EventCount Property

Document.EventCount Property
Read-only property that returns the number of event variables in the document.

Syntax
int EventCount

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numEventVars = doc.EventCount;

See Also
Document Interface
COM/ActiveX Interfaces
6.2.10. IntervalCount Property

Document.IntervalCount Property
Read-only property that returns the number of interval variables in the document.

Syntax

```c
int IntervalCount
```

Usage

**Matlab**

```matlab
nex = actxserver(' NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numIntervalVars = doc.IntervalCount;
```

See Also

Document Interface
COM/ActiveX Interfaces
6.2.11. MarkerCount Property

Document.MarkerCount Property
Read-only property that returns the number of marker variables in the document.

Syntax

```matlab
int MarkerCount
```

Usage

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numMarkerVars = doc.MarkerCount;
```

See Also

Document Interface
COM/ActiveX Interfaces
6.2.12. WaveCount Property

Document.WaveCount Property
Read-only property that returns the number of waveform variables in the document.

Syntax

```csharp
int WaveCount
```

Usage

Matlab
```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numWaveVars = doc.WaveCount;
```

See Also

Document Interface
COM/ActiveX Interfaces
6.2.13. ContinuousCount Property

Document.ContinuousCount Property
Read-only property that returns the number of continuous variables in the document.

Syntax

```matlab
int ContinuousCount
```

Usage

Matlab

```matlab	nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
numContinuousVars = doc.ContinuousCount;
```

See Also

Document Interface
 COM/ActiveX Interfaces
6.2.14. Variable Method

Document.Variable Method

Returns Variable object for the specified variable index or name.

Syntax

Variable Variable(object variableIndexOrName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variableIndexOrName</td>
<td>int or string</td>
<td>1-based variable index or a string with the variable name</td>
</tr>
</tbody>
</table>

Returns

Returns Variable object.

Usage

Matlab

defex = actxserver('NeuroExplorer.Application');
doc = defex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first variable
var1 = doc.Variable(1);
% get the variable with the name Neuron04a
neuron = doc.Variable('Neuron04a');

See Also

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.15. Neuron Method

Document.Neuron Method

Returns Variable object for the specified neuron variable index.

Syntax

Variable Neuron(int neuronIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>neuronIndex</td>
<td>int</td>
<td>1-based neuron variable index</td>
</tr>
</tbody>
</table>

Returns

Returns Variable object.

Usage

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
% get the last neuron variable
neuronLast = doc.Neuron(doc.NeuronCount);
```

See Also

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.16. Event Method

Document.Event Method
Returns Variable object for the specified event variable index.

Syntax
Variable Event(int eventIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventIndex</td>
<td>int</td>
<td>1-based event variable index</td>
</tr>
</tbody>
</table>

Returns
Returns Variable object.

Usage
Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first event variable
Event1 = doc.Event(1);
% get the last event variable
EventLast = doc.Event(doc.EventCount);
```

See Also
Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.17. Interval Method

Document.Interval Method
Returns Variable object for the specified interval variable index.

Syntax
Variable Interval(int IntervalIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IntervalIndex</td>
<td>int</td>
<td>1-based interval variable index</td>
</tr>
</tbody>
</table>

Returns
Returns Variable object.

Usage
Matlab

```matlab
ten = actxserver('NeuroExplorer.Application');
doc = ten.OpenDocument('C:\Data\MyDataFile.nex');
% get the first interval variable
Interval1 = doc.Interval(1);
% get the last interval variable
IntervalLast = doc.Interval(doc.IntervalCount);
```

See Also
Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.18. Marker Method

Document.Marker Method

Returns Variable object for the specified marker variable index.

Syntax

Variable Marker(int markerIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>markerIndex</td>
<td>int</td>
<td>1-based marker variable index</td>
</tr>
</tbody>
</table>

Returns

Returns Variable object.

Usage

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first marker variable
Marker1 = doc.Marker(1);
% get the last marker variable
MarkerLast = doc.Marker(doc.MarkerCount);
```

See Also

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.19. Wave Method

Document.Wave Method
Returns Variable object for the specified waveform variable index.

Syntax
Variable Wave(int waveIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>waveIndex</td>
<td>int</td>
<td>1-based waveform variable index</td>
</tr>
</tbody>
</table>

Returns
Returns Variable object.

Usage
Matlab

```matlab
tex = actxserver('NeuroExplorer.Application');
doc = tex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first waveform variable
Wave1 = doc.Wave(1);
% get the last waveform variable
WaveLast = doc.Wave(doc.WaveCount);
```

See Also
Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.20. Continuous Method

Document.Continuous Method
Returns Variable object for the specified continuous variable index.

Syntax

Variable Continuous(int continuousIndex)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>continuousIndex</td>
<td>int</td>
<td>1-based continuous variable index</td>
</tr>
</tbody>
</table>

Returns

Returns Variable object.

Usage

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
Continuous1 = doc.Continuous(1);
% get the last continuous variable
ContinuousLast = doc.Continuous(doc.ContinuousCount);
```

See Also

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.21. DeselectAll Method

Document.DeselectAll Method
Deselects all the data variables in the document.

Syntax
void DeselectAll()

Returns
None.

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select one variable
doc.Variable('Neuron01').Select();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();

See Also
Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.22. SelectAllNeurons Method

Document.SelectAllNeurons Method

Selects all the neuron variables in the document. Selected variables are used in analysis when
ApplyTemplate document method is called.

Syntax

void SelectAllNeurons()

Returns

None.

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all neurons
doc.SelectAllNeurons();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
dex.delete;

See Also

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.23. SelectAllContinuous Method

**Document.SelectAllContinuous Method**

Selects all the continuous variables in the document.

**Syntax**

```csharp
void SelectAllContinuous()
```

**Returns**

None.

**Usage**

**Matlab**

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all continuous variables
doc.SelectAllContinuous();
% run Perievent Histogram analysis saved in 'PSTH' template
doc.ApplyTemplate('PSTH');
% get numerical results
results = doc.GetNumericalResults();
```

**See Also**

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.24. ApplyTemplate Method

Document.ApplyTemplate Method
Runs the analysis specified in the analysis template.

Syntax
void ApplyTemplate(string templateName)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateName</td>
<td>string</td>
<td>template name</td>
</tr>
</tbody>
</table>

Returns
None.

Usage

Matlab

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all neurons
doc.SelectAllNeurons();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;
```

See Also
Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.25. GetNumericalResults Method

Document.GetNumericalResults Method

Returns 2-dimensional array of numerical results for the first graph view of the document.

Syntax

Array GetNumericalResults()

Returns

Returns 2-dimensional array of numerical results.

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% apply AutoCorrelograms analysis template
nex.RunNexScriptCommands('doc=GetActiveDocument()\nApplyTemplate(doc,"AutoCorrelograms")');
% get numerical results
results = doc.GetNumericalResults();

See Also

Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.2.26. Close Method

Document.Close Method
Closes the document.

Syntax
void Close()

Parameters
None.

Returns
None.

Usage
Matlab

gex = actxserver('NeuroExplorer.Application');
doc = gex.OpenDocument('C:\Data\MyDataFile.nex');
% close the document
doc.Close();

See Also
Variable Interface
Application Interface
Document Interface
COM/ActiveX Interfaces
6.3. Variable

Variable Interface

NeuroExplorer Application object provides access to the open files and variables contained in the files. For example, to open a document in NeuroExplorer and get the first event variable in the document in Matlab, you can use:

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
event1 = doc.Event(1);
```

Here `Event()` method returns a Variable object corresponding to the first event variable in the file. The properties and methods of the Variable object are listed below.

### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Variable name</td>
</tr>
<tr>
<td>TimestampCount</td>
<td>Number of timestamps in the variable</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamps</td>
<td>Returns all the variable timestamps in an array</td>
</tr>
<tr>
<td>IntervalStarts</td>
<td>For interval variables, returns all the interval start values</td>
</tr>
<tr>
<td>IntervalEnds</td>
<td>For interval variables, returns all the interval end values</td>
</tr>
<tr>
<td>FragmentTimestamps</td>
<td>For continuous variables, returns fragment timestamps</td>
</tr>
<tr>
<td>FragmentCounts</td>
<td>For continuous variables, returns the number of data points in each fragment</td>
</tr>
<tr>
<td>ContinuousValues</td>
<td>For continuous variables, returns A/D values in milliVolts</td>
</tr>
<tr>
<td>MarkerValues</td>
<td>For marker variables, returns marker string values</td>
</tr>
<tr>
<td>WaveformValues</td>
<td>For waveform variables, returns waveform values</td>
</tr>
<tr>
<td>SamplingRate</td>
<td>For continuous and waveform variables, returns sampling rate in Hz</td>
</tr>
<tr>
<td>Select</td>
<td>Selects the variable for analysis</td>
</tr>
<tr>
<td>Deselect</td>
<td>Deselects the variable</td>
</tr>
</tbody>
</table>

### See Also

Document Interface
COM/ActiveX Interfaces
6.3.1. Name Property

Variable.Name Property
Read-only property that returns a string with the variable name.

Syntax
string Name

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
neuron1Name = neuron1.Name;

See Also
Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.2. TimestampCount Property

Variable.TimestampCount Property

Read-only property that returns the number of timestamps in the variable. For interval variables, returns the number of intervals. For waveforms variable, returns the number of waveforms. For continuous variables, returns the total number of data points.

Syntax

```c
int TimestampCount
```

Usage

Matlab

```matlab
tenex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
numTimestamps = neuron1.TimestampCount;
```

See Also

Variable Interface

Document Interface

COM/ActiveX Interfaces
6.3.3. Timestamps Method

**Variable.Timestamps Method**

Returns all the timestamps of the variable in seconds. Timestamps are returned as an array (vector) of double values. For interval variables, returns the interval starts. For waveforms variable, returns the waveform timestamps. For continuous variables, returns the timestamps corresponding to all the variable data points.

**Syntax**

SAFEARRAY(double) Timestamps()

**Parameters**

None.

**Usage**

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first neuron variable
neuron1 = doc.Neuron(1);
% get all the timestamps
ts = neuron1.Timestamps();
% now ts is a vector of timestamps
%
% get the first continuous variable
cont1 = doc.Continuous(1);
% get all the timestamps for continuous variable
cont1_ts = cont1.Timestamps();
```

**See Also**

Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.4. IntervalStarts Method

Variable.IntervalStarts Method
For an interval variable, returns the interval start values in seconds. This method is valid only for interval variables.

Syntax
SAFEARRAY(double) IntervalStarts()

Parameters
None.

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first interval variable
int1= doc.Interval(1);
% get all the interval starts
int1Starts = int1.IntervalStarts();

See Also
Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.5. IntervalEnds Method

Variable.IntervalEnds Method

For an interval variable, returns the interval end values in seconds. This method is valid only for interval variables.

Syntax

SAFEARRAY(double) IntervalEnds()

Parameters

None.

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first interval variable
int1= doc.Interval(1);
% get all the interval ends
int1Ends = int1.IntervalEnds();

See Also

Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.6. FragmentTimestamps Method

Variable.FragmentTimestamps Method

For continuous variable, returns the fragment timestamp values in seconds. This method is valid only for continuous variables.

In general, a continuous variable may contain several fragments of data. Each fragment may be of a different length. NeuroExplorer does not store the timestamps for all the A/D values since they would use too much space. Instead, for each fragment, it stores the timestamp of the first A/D value in the fragment and the index of the first data point in the fragment. The timestamps of the first A/D value in each fragment are returned by this method.

Syntax

SAFEARRAY(double) FragmentTimestamps()

Parameters

None.

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1 = doc.Continuous(1);
% get all the fragment timestamps
cont1FragmentTs = cont1.FragmentTimestamps();

See Also

Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.7. FragmentCounts Method

Variable.FragmentCounts Method

For continuous variable, returns the number of data points in fragments. This method is valid only for continuous variables.

In general, a continuous variable may contain several fragments of data. Each fragment may be of a different length. NeuroExplorer does not store the timestamps for all the A/D values since they would use too much space. Instead, for each fragment, it stores the timestamp of the first A/D value in the fragment and the index of the first data point in the fragment. The number of data points in each fragment are returned by this method.

Syntax

SAFEARRAY(double) FragmentCounts()

Parameters

None.

Usage

Matlab

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1= doc.Continuous(1);
% get all the fragment counts
cont1FragmentCounts = cont1.FragmentCounts();
```

See Also

Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.8. ContinuousValues Method

Variable.ContinuousValues Method

For continuous variable, returns all the A/D values in milliVolts. This method is valid only for continuous variables.

Syntax

SAFEARRAY(double) ContinuousValues()

Parameters

None.

Usage

Matlab

```matlab
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1= doc.Continuous(1);
% get all the values
cont1Values = cont1.ContinuousValues();
```

See Also

Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.9. MarkerValues Method

Variable.MarkerValues Method
For a marker variable, returns all the marker values as strings. The values are returned in a two-dimensional array. Each row of the array represents all the marker strings for one timestamp.
This method is valid only for marker variables.

Syntax
SAFEARRAY(string) MarkerValues()

Parameters
None.

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first markervariable
marker1 = doc.Marker(1);
% get all the marker values
marker1Values = marker1.MarkerValues();

See Also
Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.10. WaveformValues Method

Variable.WaveformValues Method
For waveform variable, returns all the waveform values in milliVolts. The values are returned in a two-dimensional array. Each row of the array represents one waveform.
This method is valid only for waveform variables.

Syntax
SAFEARRAY(double) WaveformValues()

Parameters
None.

Usage
Matlab

def = actxserver('NeuroExplorer.Application');
doc = def.OpenDocument('C:\Data\MyDataFile.nex');
% get the first waveform variable
doc.Wave(1);
% get all the values
wavelValues = wavel.WaveformValues();

See Also
Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.11. SamplingRate Property

Variable.SamplingRate Property
Read-only property that returns the sampling rate (in Hz) of a continuous or waveform variable. For other variable types, returns zero.

Syntax

double SamplingRate

Usage

Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% get the first continuous variable
cont1 = doc.Continuous(1);
% get sampling rate
samplingRate = cont1.SamplingRate;

See Also

Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.12. Select Method

Variable.Select Method
Selects the variable. Selected variables are used in analysis when ApplyTemplate document method is called.

Syntax
void Select()

Parameters
None.

Usage
Matlab

nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select variable with the name 'Neuron01'
doc.Variable('Neuron01').Select();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;

See Also
Variable Interface
Document Interface
COM/ActiveX Interfaces
6.3.13. Deselect Method

Variable.Deselect Method
Deselects the variable. Only selected variables are used in analysis when ApplyTemplate document method is called.

Syntax
void Deselect()

Parameters
None.

Usage
Matlab

```
nex = actxserver('NeuroExplorer.Application');
doc = nex.OpenDocument('C:\Data\MyDataFile.nex');
% deselect all variables
doc.DeselectAll();
% select all neurons
doc.SelectAllNeurons();
% deselect variable with the name 'Neuron01'
doc.Variable('Neuron01').Deselect();
% run Interspike Interval Histogram analysis saved in 'ISI' template
doc.ApplyTemplate('ISI');
% get numerical results
results = doc.GetNumericalResults();
% close NeuroExplorer
nex.delete;
```

See Also
Variable Interface
Document Interface
COM/ActiveX Interfaces
# Index

## 1D Data Viewer
- 23

## 3D Graphics
- 124, 126

### A
- Activity Animation 129
- Adjusting Analysis Properties 29
- Analysis Templates 30
- Analyzing Data 24
- Application 319
- Autocorrelograms 55, 116
- Autocorrelograms Versus Time 116

### B
- Burst Analysis 87

### C
- Coherence Analysis 107
- COM/ActiveX Interfaces 318
- Confidence Limits 48
- Continuously Recorded Data 43
- Correlations with Continuous Variable 94
- Crosscorrelograms 63, 68
- Cumulative Activity Graphs 76
- Cumulative Sum Graphs 49

### D
- Data Selection 45
- Data Types 38
- Document 146, 147, 148, 149, 150, 158, 165, 320, 321, 324, 325, 329

### E
- Epoch Counts 105
- Events 39, 190
- Excel 35, 47, 218, 223, 267, 268, 269
- Expressions 135

### F
- File Variables 133
- Flow Control 136
- Functions 138, 139, 158, 165, 179, 193, 200, 213, 226, 260, 267, 270, 273, 276, 303, 315

### G
- Graphics 36, 118, 120, 124, 126, 203, 211, 212, 271
- Graphics Modes 118

### H
- How to Select Variables 26, 28

### I
- Importing Data 15, 18, 19, 21
- Importing Files 16
- Instant Frequency 77
- Interspike Interval Histograms 52
- Interspike Intervals vs. Time 78

### J
- Joint ISI Distribution 114
- Joint PSTH 74

### L
- Lines 122

### M
- Markers 41
- Matlab 21, 34, 47, 260, 261, 262, 263, 264, 265, 266

### N
- NexScript 130, 327, 328
- Numerical Results 31, 213

### O
- Opening Files 15

### P
- Perievent Histograms 48, 58
- Perievent Rasters 71
- Perievent Spectrograms 112
- Place Cell Analysis 98
- Poincare Maps 79
- Population Vectors 42
- Post-Processing 46
- Power Point 32, 270
- Power Spectral Densities 84
- Principal Component Analysis 90
- Programming with NexScript 130
- PSTH Versus Time 92

### R
- Rasters 70, 71
- Rate Histograms 50
- Rectangles 123
- Regularity Analysis 96
- Reverse Correlation 102

### S
- Saving Graphics 36
- Saving Results 32
- Screen Elements 11
- Script Variables 132
- Selecting Variables 25
- Sentinel Keys 9
- Shift-Predictor 68
- Spectrogram 110, 112
- Spike Trains 38
- Synchrony vs. Time 80

### T
- Text Files 18
- Text Labels 121
- Trial Bin Counts 82

### V
- Variable 25, 26, 28, 94, 132, 133, 165, 179, 193, 226, 257, 337, 344, 357
- Viewing Multiple Histograms 125
- Viewing the Neuronal Activity 127

### W
- Waveforms 43