



# Clampfit Batch Analysis

Electrophysiology Data Management and Analysis Software

User Guide

## Clampfit Batch Analysis User Guide

This document is provided to customers who have purchased Molecular Devices equipment, software, reagents, and consumables to use in the operation of such Molecular Devices equipment, software, reagents, and consumables. This document is copyright protected and any reproduction of this document, in whole or any part, is strictly prohibited, except as Molecular Devices may authorize in writing.

Software that may be described in this document is furnished under a non-transferrable license. It is against the law to copy, modify, or distribute the software on any medium, except as specifically allowed in the license agreement. Furthermore, the license agreement may prohibit the software from being disassembled, reverse engineered, or decompiled for any purpose.

Portions of this document may make reference to other manufacturers and/or their products, which may contain parts whose names are registered as trademarks and/or function as trademarks of their respective owners. Any such usage is intended only to designate those manufacturers' products as supplied by Molecular Devices for incorporation into its equipment and does not imply any right and/or license to use or permit others to use such manufacturers' and/or their product names as trademarks.

Each product is shipped with documentation stating specifications and other technical information. Molecular Devices products are warranted to meet the stated specifications. Molecular Devices makes no other warranties or representations express or implied, including but not limited to, the fitness of this product for any particular purpose and assumes no responsibility or contingent liability, including indirect or consequential damages, for any use to which the purchaser may put the equipment described herein, or for any adverse circumstances arising therefrom. The sole obligation of Molecular Devices and the customer's sole remedy are limited to repair or replacement of the product in the event that the product fails to do as warranted.

### **For research use only. Not for use in diagnostic procedures.**

The trademarks mentioned herein are the property of Molecular Devices, LLC or their respective owners. These trademarks may not be used in any type of promotion or advertising without the prior written permission of Molecular Devices, LLC.

Patents: <http://www.moleculardevices.com/patents>

Product manufactured by Molecular Devices, LLC.  
3860 N. First Street, San Jose, California, 95134, United States of America.  
Molecular Devices, LLC is ISO 9001 registered.  
©2021 Molecular Devices, LLC.  
All rights reserved.



# Contents

<b>Chapter 1: Introduction</b> .....	<b>7</b>
Configuring MDC File Server .....	7
First Time Use Database Configuration .....	8
Importing Data Files .....	10
Running a Sample Macro .....	11
Creating a Macro .....	12
Modifying the Sample Macros .....	13
<b>Chapter 2: From Import to Analysis</b> .....	<b>15</b>
Import Data Files .....	15
Interface Overview .....	15
Create Datasets .....	20
Attributes .....	21
Browse Datasets .....	22
Analysis .....	25
Export Results .....	30
Macros .....	30
Database .....	32
<b>Chapter 3: Trial Editing and Analysis Commands</b> .....	<b>33</b>
Trial Editing Menu .....	33
Analysis Menu .....	36
<b>Chapter 4: Configuration</b> .....	<b>49</b>
Configuring MDC File Server .....	49
First Time Use Database Configuration .....	50
<b>Chapter 5: Digital Filters</b> .....	<b>53</b>
Finite vs. Infinite Impulse Response Filters .....	53
Digital Filter Characteristics .....	54
End Effects .....	55
Bessel Lowpass Filter (8 pole) Specifications .....	55
Boxcar Smoothing Filter Specifications .....	57
Butterworth Lowpass Filter (8 pole) Specifications .....	58
Chebyshev Lowpass Filter (8 pole) Specifications .....	60
Gaussian Lowpass Filter Specifications .....	62
Notch Filter (2 pole) Specifications .....	63

Settling Points .....	64
RC Lowpass Filter (single pole) Specifications .....	64
RC Lowpass Filter (8 pole) Specifications .....	66
RC Highpass Filter (single pole) Specifications .....	67
Bessel Highpass Filter (8-pole analog) Specifications .....	68
Electrical Interference Filter .....	69
<b>Chapter 6: Curve Fitting .....</b>	<b>77</b>
Fitting Model .....	77
Function Parameters .....	77
Parameter Errors .....	77
Fitting Failure .....	78
Numerical Limitations .....	78
Units .....	79
Levenberg-Marquardt Method .....	79
Simplex Method .....	80
Variable Metric Method .....	82
Chebyshev Transform .....	83
Model Comparison .....	96
Define a Custom Function .....	97
Minimization Functions .....	98
Weighting .....	99
Normalized Proportions .....	100
Zero-shifting .....	101
<b>Chapter 7: Fitting Functions .....</b>	<b>103</b>
Beta Functions .....	103
Binomial .....	104
Boltzmann, Charge Voltage .....	104
Boltzmann, Shifted .....	105
Boltzmann, Standard .....	105
Boltzmann, Z Delta .....	106
Current Time Course (Hodgkin Huxley) .....	106
Exponential, Alpha .....	106
Exponential, Cumulative Probability .....	106
Exponential, Log Probability .....	107
Exponential, Power .....	107
Exponential, Product .....	107
Exponential, Sloping Baseline .....	107

Exponential, Standard .....	107
Exponential, Weighted .....	108
Exponential, Weighted/Constrained .....	108
Gaussian .....	108
Goldman Hodgkin Katz .....	108
Hill (4 Parameter Logistic) .....	109
Hill, Langmuir .....	109
Hill, Steady State .....	109
Lineweaver Burk .....	109
Logistic Growth .....	110
Lorentzian Distribution .....	110
Lorentzian Power 1 .....	110
Lorentzian Power 2 .....	110
Michaelis Menten .....	111
Nernst .....	111
Parabola, Standard .....	111
Parabola, Variance Mean .....	111
Poisson .....	111
Polynomial .....	112
Straight Line, Origin at Zero .....	112
Straight Line, Standard .....	112
Voltage Dependent Relaxation .....	112
Constants .....	112
<b>Glossary .....</b>	<b>113</b>



There are several sample macros provided within Clampfit Batch Analysis Software. There is also corresponding sample data that you can import.

The following sections provide a brief explanation about using macros. To see the configuration details of a macro in the program, double-click on the macro steps to view their configuration dialogs.

## Configuring MDC File Server

The MDC File Server runs on Windows 7 and 10 computer operating systems. The computer it runs on requires enough hard drive space to store your data files. The storage location can be changed if it becomes full, and the database keeps track of data files in multiple storage locations. If you set a new storage location, keep the existing data storage location.

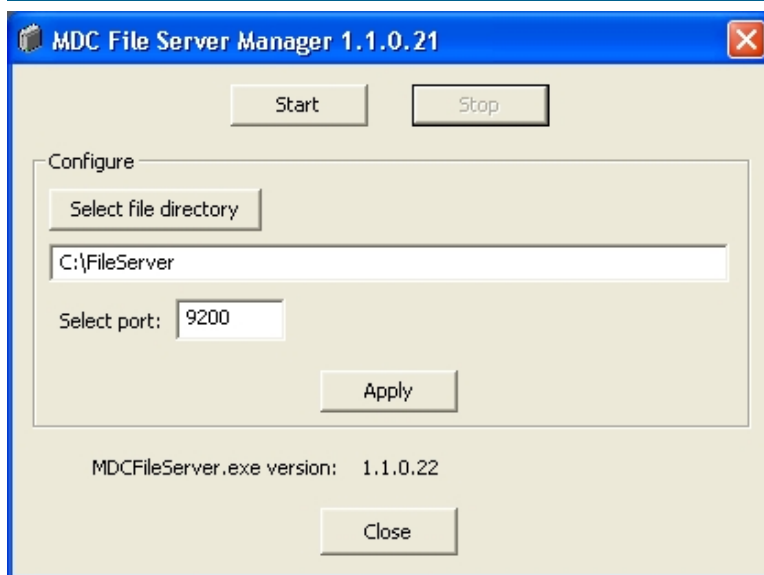
Before starting Clampfit software to run Batch Analysis the first time, you must first configure MDC File Server. MDC File Server is required for managing imported data. It runs quietly in the background while you use Batch Analysis. MDC File Server must be running for Clampfit software Batch Analysis to run.

To configure MDC File Server:

1. After you install the pCLAMP software, open the **MDC File Server Manager** from **Start > All Programs > Molecular Devices > MDC File Server > MDC File Server**, right-click and **Run as administrator**.
2. In the **MDC File Server Manager** dialog, click **Select file directory**.



**CAUTION!** If you select a computer other than the local one, you must have access permissions to the folder location at all times.



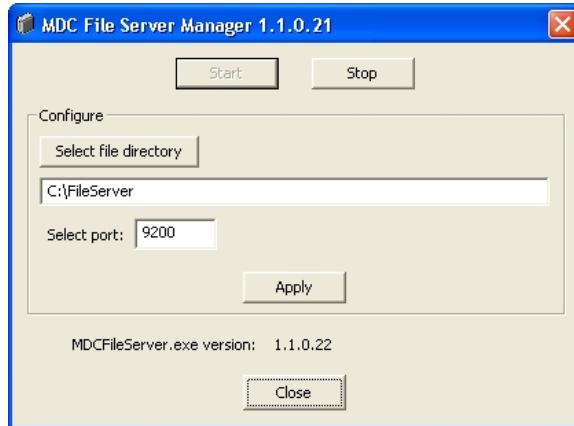
3. Click **Create new folder**, type the name **FILESERVER**, and click **OK**.



**Tip:** You can name the new folder something other than **FILESERVER** if needed.

4. In the **Select port** field, type **9200**, and click **Apply**.

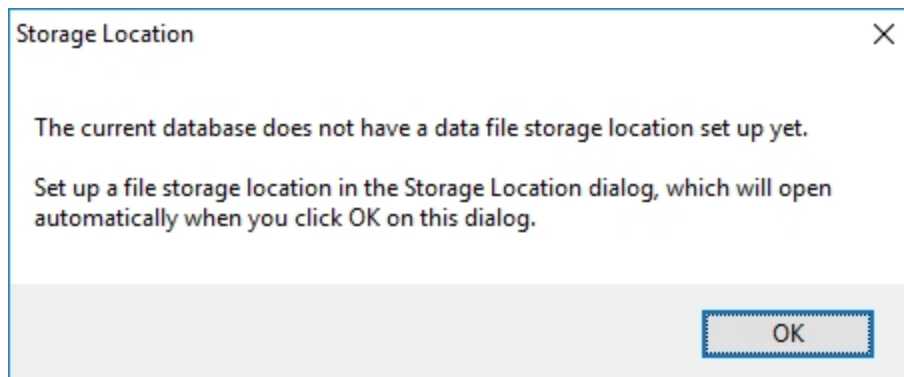
5. Click the **Start** button and wait for the **Start** button to disable.
6. When the **Start** button is inactive, click **Close**.



## First Time Use Database Configuration

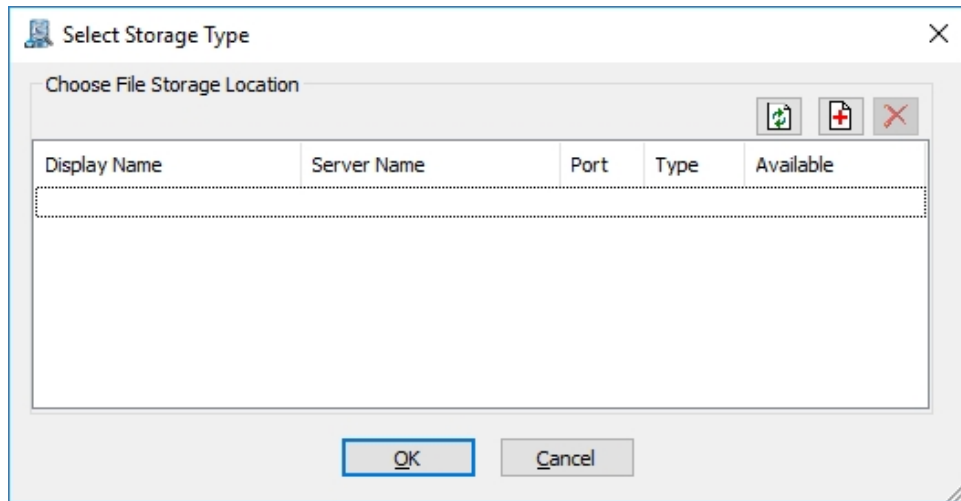
If this is the first time using Clampfit software Batch Analysis, you must do the following to configure your data storage folder:

1. Ensure that your Clampfit software license key dongle is installed on the same computer that the software is installed.
2. Start Clampfit software.
3. Select **Batch Analysis > Run**.
4. After the **Batch Analysis** window appears, when you see the **Storage Location** message, click **OK**.

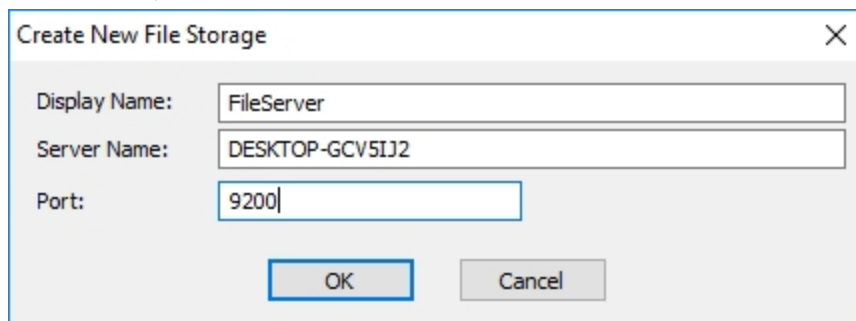




5. When the **Storage Location Type** dialog appears, click the + (add) button.



6. In the **Create New File Storage** dialog, do the following:
- In the **Display Name** field, type a name for your data file storage folder. See [Configuring MDC File Server on page 7](#).
  - In the **Server Name** field, verify that the name of your computer (local computer name) is correct.
  - In the **Port** field, leave the default setting as **9200**.
  - To finish, click **OK**.



- In the **Storage Location Type** dialog, for your file storage entry, confirm that under the **Available** heading, **Yes** appears.
- Click the **Refresh** button to update for **Available** if needed.
- To finish, select your file storage entry row and click **OK**.

## Importing Data Files

The Clampfit Advanced Analysis Software installer includes a few sample macros to use to familiarize yourself with the Batch Analysis functionality. To use these sample macros the first time, you must import the provided sample data files. When you import your own data files they must be .abf files from Clampex software.

The following procedure uses the imported sample data **Peak Data** to be used with the installed sample macro **DemoPeak**. These same procedures apply to any other imported sample data used with any of the other corresponding sample macros. Where you see **Peak Data** in the following procedure you can select an alternative data folder.

To import data files:

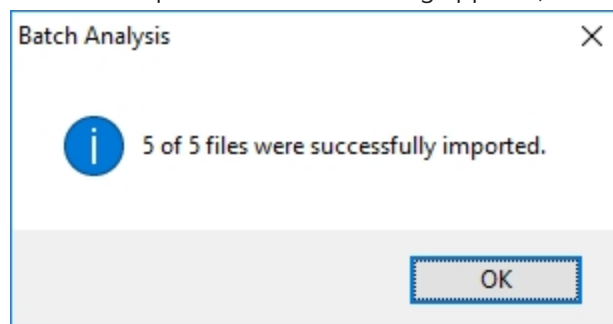
1. In the **Batch Analysis** dialog, select **File > Import Data > Entire Folder**.
2. Navigate to and select **C:\Program Files\Molecular Devices\pCLAMP 11\Sample Macros\Peak Data**.
3. Click **Open**.



**Tip:** Windows 10 users, if clicking **Open** fails, double-click on **Peak Data**.

---

4. When the import confirmation dialog appears, click **OK**.

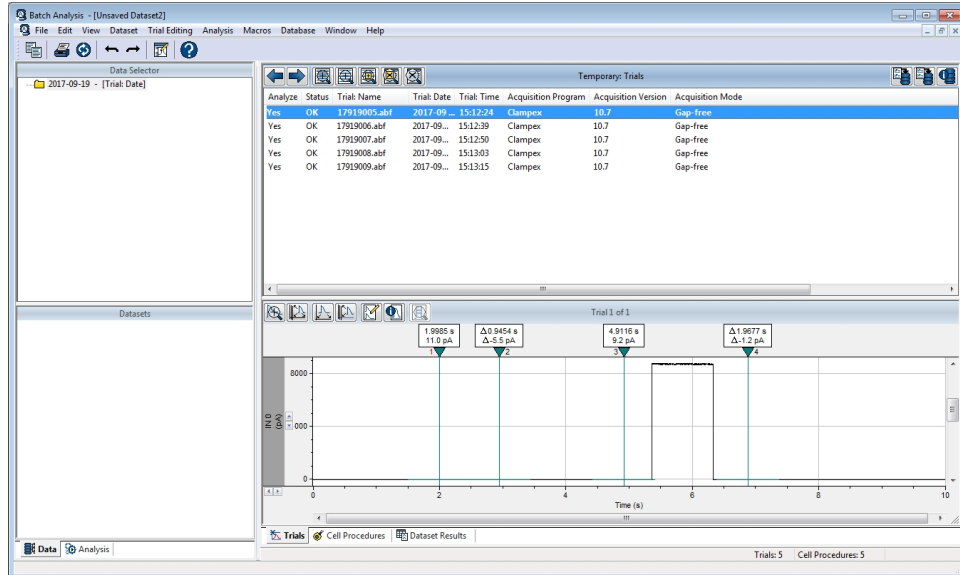


## Running a Sample Macro

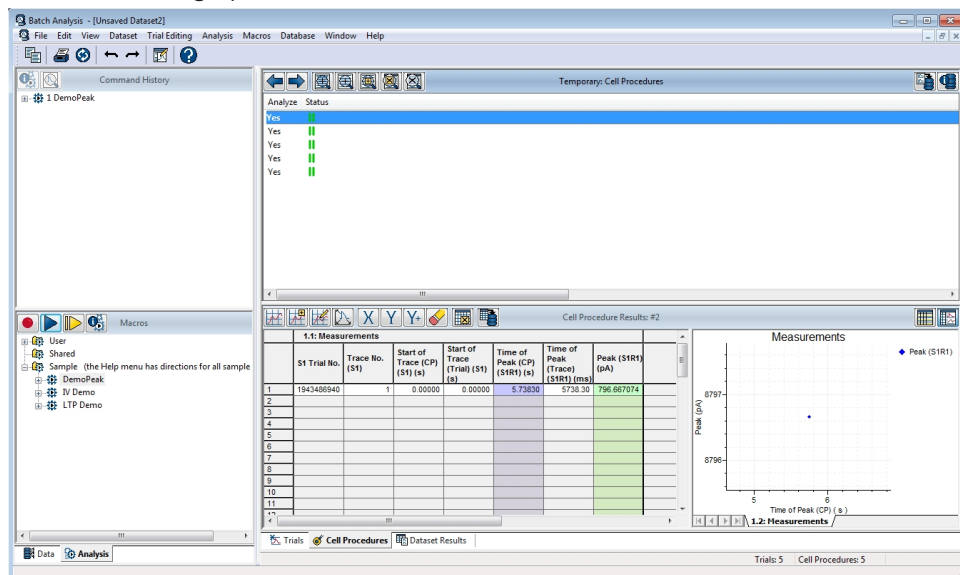
The following procedure uses the imported sample data **Peak Data** with the installed sample macro **DemoPeak**. See [Importing Data Files on page 10](#). These same procedures apply to any other imported sample data used with any of the other corresponding sample macros. Where you see **DemoPeak** in the following procedure select an alternative macro.

To run a sample macro:

1. In the top left quadrant of the **Batch Analysis** window under **Data Selector**, double-click on the data folder you imported to load the **Trials**.



2. In the bottom left, click the **Analysis** tab.
3. In the **Macros** panel, in the **Sample** tree, right-click on **DemoPeak** and select **Run Macro**. When a macro runs successfully, the lower right panel displays the measurements table and associated graph.



## Creating a Macro

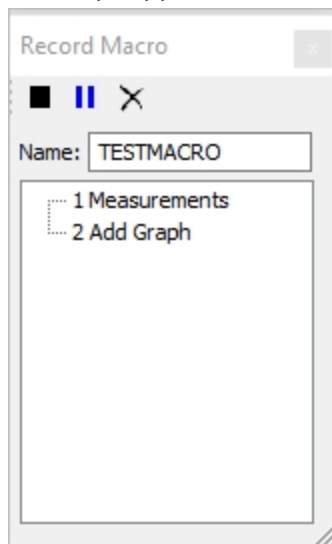
Any number of commands from the **Analysis** menu can be tied together as a single analysis sequence and stored in a macro. These new macros get stored in the **Macros** pane of the **Analysis** tab.

To create a macro:

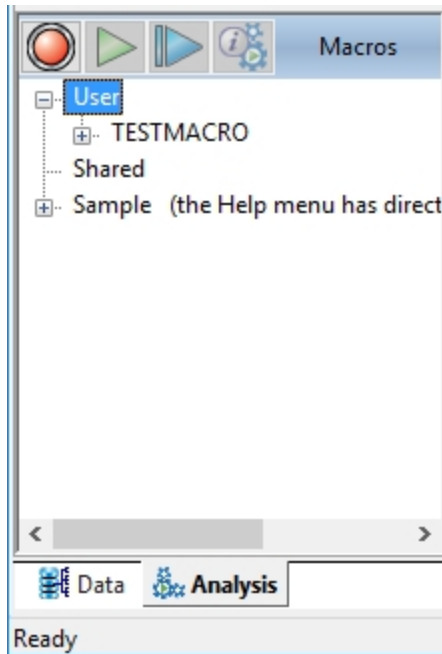
1. In the **Batch Analysis** dialog, select **Macros > Record**.
2. When the **Record Macro** dialog appears, type your new macro name in the **Name** field. Do not press <Enter>.



3. Go to the **Analysis** menu and select options as needed and specify settings as needed. The steps appear in the **Record Macro** dialog as they are added.



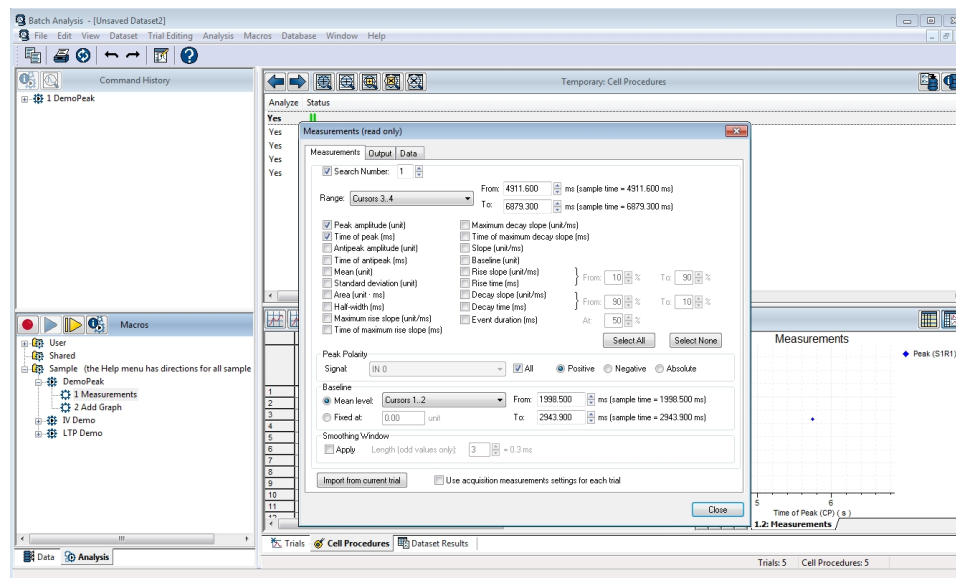
4. When you are finished adding steps, in the **Record Macro** dialog, click the **Stop** button. Your new macro saves and appears in the **Macros** pane of the **Analysis** tab under **User**.



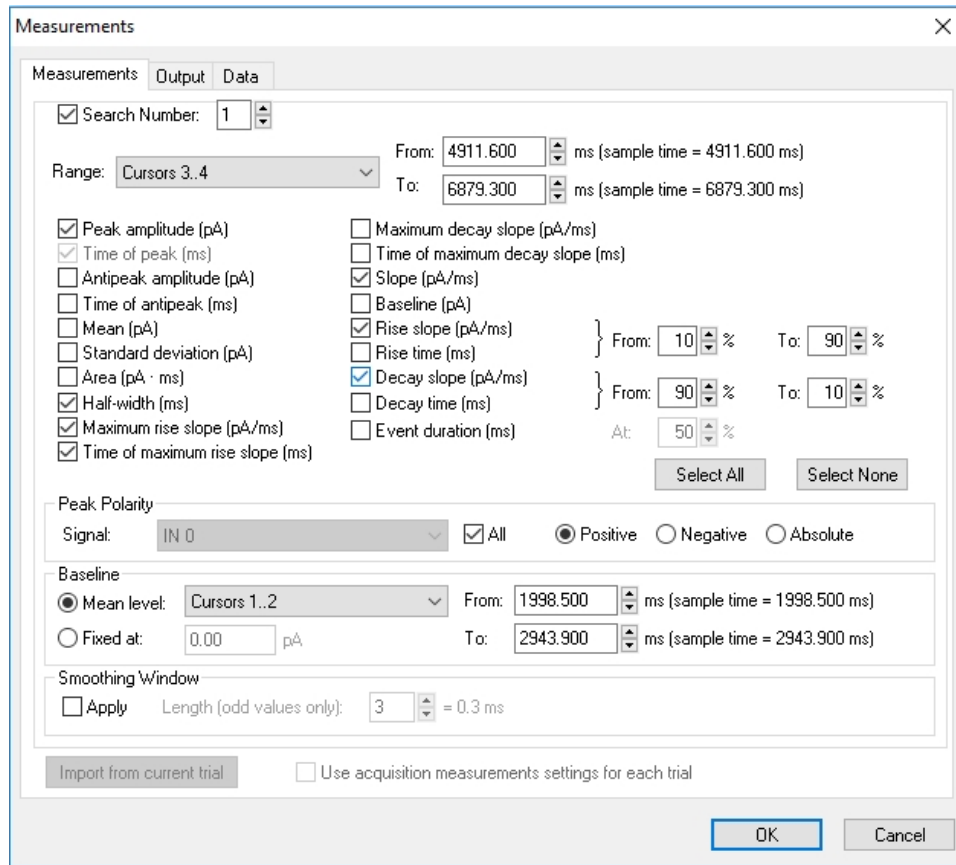
 **Tip:** Macros in the **User** tree are editable.

## Modifying the Sample Macros

Macros in the **Sample** tree are read-only. To view macro step settings, in the **Macros** pane of the **Analysis** tab, expand the macro name and double-click on the step name.



To edit macros, right-click on the sample macro name and select **Make a User copy** . It is copied to the **User** macro tree, where you can open the **Analysis** steps within the macro to reconfigure them. Click **OK** to save changes.



You can rename the macros and change their comments from the right-click menu as well.

You cannot add or remove steps to a macro when you edit it. To do this, record a new macro and use **Run Step** to copy the steps from the existing macro into the new one. Skip steps that are not wanted, or add new steps from the dialogs in the **Analysis** menu.

To delete a macro from the **User** tree, select the name and press **Delete**.

This chapter provides details of Clampfit Batch Analysis features and usage, following typical data flow from data importation, querying and preparation for analysis, through analysis and results handling.

### Import Data Files

The files you import into Batch Analysis are Clampex software .ABF data files. When you select to **Import Data Files**, you can import up to 100 **Individual Files** or an **Entire Folder** of files.

To import data files:

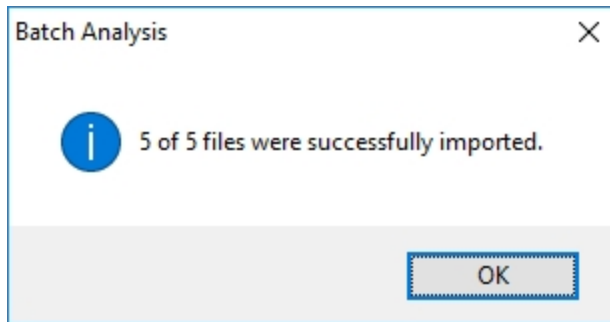
1. In the **Batch Analysis** dialog, select **File > Import Data Files > (Individual Files or Entire Folder)**.
2. Navigate to and select the needed .ABF files.
3. Click **Open**, or **Select** depending on your operating system.

---

 **Tip:** If clicking **Open** fails, double click on the folder.

---

4. When the import confirmation dialog appears, click **OK**.



### Interface Overview

The following is an overview of the various parts of the software interface.

- [Main Menu on page 15](#)
- [Data Tab on page 16](#)
- [Analysis Tab on page 17](#)
- [Dataset Windows on page 17](#)
- [Program Options on page 18](#)
- [Layout Window on page 19](#)
- [Copy and Print on page 19](#)

#### Main Menu

The main menu bar in Clampfit Batch Analysis includes various menus when a dataset window is open. Most of the options in the menus are also available as keyboard shortcuts, buttons, or are in popup menus.

The following are short descriptions of each menu.

**File:** This menu has file import and export options, file information, and printing setup.

The first set of commands run database queries for the current selection in the **Data Selector**.

**Edit:** The most important commands are **Undo** and **Redo** which undo and reapply trial editing and analysis commands.

Select **Copy** to copy results data, graphs, and lists to the Windows clipboard.

**View:** This menu has options to set the way datasets appear.

Open the **Attributes** dialog from this menu.

Open the **Layout Window** dialog from this menu.

**Dataset:** All commands for altering the content of datasets and saving datasets are included in this menu. It includes the **Analyze** commands that determine which items in the dataset will have analysis applied to them.

**Trial Editing:** The commands in this menu all change data recorded in trials, creating modified trials as output. Some commands (left hand list) apply to all the trials in the dataset selected for analysis, at once. These commands can be included in macros. The right hand commands only ever apply to one trial at a time, and cannot be included in macros.

**Analysis:** All commands that generate results sheet output, or graphs, are in this menu. Some commands apply to trials and others take results sheet columns as input. All these commands can be included in macros.

**Macros:** All commands for recording, running, and managing macros are in this menu.

**Configure:** The **Program Options** dialog has many general program settings. Also options to configure dataset lists and tree hierarchies.

**Database:** General database options to create new databases, select a database, and configure file storage locations.

**Window:** Standard MS Windows window management, select dataset windows here if you have multiple datasets at once.

**Help:** Access the manual in PDF, web links, including software updates, and the **About** dialog.

## Data Tab

The **Data** tab gives access to data in the database. There are two panes, **Data Selector** and **Datasets**.

### Data Selector

The **Data Selector** pane is a user configurable hierarchy of database attributes used to create queries to return data from the database, into temporary datasets. Double click on nodes to retrieve data from the database.

Most database attributes can be included in the hierarchy, in any order (**Configure > Data Selector Hierarchy**, or **Data Selector** popup menu). See [Create Datasets on page 20](#) for more information.

### Datasets

The **Datasets** pane displays saved datasets under a short, user configurable hierarchy.

Typically, data returned from a query with the **Data Selector** is tidied, removing bad files and preprocessing some trials (for example, removing traces) to make the data ready for analysis. Then it can be saved in a dataset. Analysis results are always saved with the dataset they were derived from. See [Saving Datasets on page 25](#) for more information.



## Analysis Tab

There are two panes. The top pane is **Command History**, which records the analysis commands run on the current dataset. The bottom pane is **Macros**.

### Command History

In the **Command History**, each new trial editing or analysis command for the current dataset is listed. Macros are added as single nodes that open to show their constituent steps.

Open steps with a double click or the **Properties** button to show the dialog for the analysis (read only) populated with the settings that were used.

**View Analysis Set**, also in the **Command History** popup menu, opens a dialog that reports the trials or data on the **Dataset Results** tab, that the selected analysis step was carried out on. The destinations of the analysis (**Trials** or **Dataset Results**) are also reported.

**Show Cursor Positions**, in the same popup menu, shifts the cursors in the currently open trial to the boundaries of the search and baseline regions used in the selected analysis step. This command only applies to analysis steps that used search regions, and within this, to analyses where the regions were defined as cursor regions (in contrast to using epochs to define the regions).

If an analysis step is undone, the node is removed from the bottom of the **Command History**.

### Macros

The **Macros** pane has buttons for some common macro commands, and lists available macros in three folders. See [Macros on page 30](#) for information about creating and using macros.

## Dataset Windows

Clampfit Batch Analysis can have multiple datasets open at once, each in a separate dataset window. If a new query is run from the **Data Selector** pane with **File > Open**, or by double-clicking on a **Data Selector** node, the new dataset replaces the existing one in the dataset window that last had focus. Alternatively, a new dataset can be created in a new window with **File > Open in New Window**.

Data can be added to a dataset at any stage by clicking a node in the **Data Selector** pane and using **File > Add Tree Selection to Current Dataset**.

Dataset windows can be minimized, tiled, and otherwise arranged within the central area of the interface.

Each new dataset window is numbered in the top left corner of the title bar. The name of saved datasets is reported at the top of the window, or **Temporary** shown for temporary datasets. The **Window** menu has a list of all open dataset windows that you can use to give focus to one of them, bringing it to the front.

The number of trials in each dataset is reported in the dataset **Status Bar** at the bottom of the dataset window. If any trial editing or analysis steps have been run on a dataset, these are displayed in the **Command History** pane when the dataset is selected.

Each dataset window has three tabs:

- **Trials:** Lists all the trials in the dataset in the top half, and has a graphical display of one trial at a time in the bottom half. Columns in the trials list are configurable to show database attributes for the trials.  
There are a number of display options, open popup menus from the main data area, axis areas or cursors for these. The **Properties** dialog opens from the main area popup menu with many additional configuration options.  
Additionally, set axis scaling and other options to apply when you open new trials into the trace display with the **Configure** menu **Program Options > Open Trial Options**. For example, you might want to always show the entire X axis range when you open new trials, or you can configure the **Open Trial Options** to retain the current X axis range so you can zoom in to a particular region, then scroll through all the trials looking at just that region.
- **Analysis Results:** Results of analyses applied to the trials are written to a spreadsheet in the left half of the pane. Graphs of the spreadsheet data are shown in the right half of the pane.
- **Dataset Results:** This tab has only one sheet for results from across all the trials in the dataset. The sheet has the same format as the analysis results, half spreadsheet and half graph.

## Program Options

The **Configure > Program Options** dialog has a range of display and program behavior options that affect your interaction with the Clampfit Batch Analysis program. Some of these are cosmetic, but others are more significant.

You can return all or selected settings to their defaults with buttons at the bottom of the dialog.

### Open Trial Options

The settings in this section define how new trials are opened into the **Trials** tab.

For example, the **X Axis Scaling** of each new trial can open as either:

- **Full scale**
- **Keep existing**, if, for example you zoomed in to view a particular region and want to see this same region in new trials.
- **Use acquisition**, with the same X axis range used during acquisition.

There are similar options for **Y Axis Scaling**, **Sweep Selection**, and **Signal Selection**.

### Display Options

This section has a number of options that affect how information is displayed in Clampfit Batch Analysis, as well as settings that allow you to change the 'look and feel' of the program, for example, title bar colors.

Beyond the coloration options, there are settings for:

- **Time Precision:** Set the number of decimal places used in the display of time values throughout the program, from zero to three. This option is for display purposes only—full time information is always kept and used for analysis.
- **Data Selector Chronological Ordering:** Set so that time and date attributes in the Data Selector have the most recent nodes at the top or the bottom.
- **Data Selector Sub Item Display:** Nodes in the Data Selector can show the number of nodes at the level below. This field sets a threshold such that if there are more nodes at the level below than the number you enter, the number of these is displayed, but otherwise not.

- **Data Selector Empty Items:** Normally this setting should stay at Hidden. When the Visible option is shown, nodes with value <none> are displayed. These nodes should generally have no trials under them, but with faulty importation it is possible to get trials here. Enable this option only if trouble shooting to find missing data.
- **User-Defined Columns:** If you have set user defined columns (for example, columns that show more than one attribute) in the trials name, these columns appear here. See [User-Defined Columns on page 22](#).

### Graph Options

This section has options that control the creation of graphs and the linkage between graphs and the spreadsheet data they plot.

- **Synchronize Selected X/Y Columns:** When enabled, clicking on a point in a graph plot selects the X and Y columns for the plot, on the spreadsheet. Within the selected columns, the X and Y coordinates of the individual point are further highlighted.
- **Plot Selection Distance:** The number of pixels from a graph point within which you must click to select the point.
- **Selected Graph Point Cursor:** When enabled the selected point in a graph is shown with dashed crosshairs. Disabling removes the crosshairs.
- **Automatic Graphing:** When enabled Clampfit Batch Analysis automatically generates a graph when an analysis is run, selecting the X and Y parameters from the analysis output. If **Skip Dialog** is enabled, the graphs are generated without opening the **Graph** dialog for configuration.

With **Merge Graphs** on, output from new analysis steps are plotted on an existing graph, if possible. Otherwise each analysis creates a new graph.

### General Options

This section has some general settings for downloading options:

- **Maximum Cache Size:** Sets the size of the cache folder (in MB) on the local computer:  
**C:\Users\[UserName]\AppData\Local\Temp\BatchAnalysis Cache**  
 where data files are stored when they are downloaded from the database. Once the cache is full, the least recently used files are removed to allow the download of new files.

### Layout Window

The **Layout Window** dialog enables simple layout functionality for the generation of printed reports, for example, for printed laboratory records. It is not intended for publication quality layouts.

Open the **Layout Window** dialog from the View menu or the toolbar button. Keep the window open while you use Clampfit Batch Analysis to move to different locations in the client and copy data to the window. As well as being able to copy data and graphs to the window, you can write text and create simple drawings.

Layouts are saved as **.alf** files.

All pages printed from the **Layout Window** dialog have a footer with the date and time of printing and the full user name. Page headers are user configurable.

### Copy and Print

Many elements of the Clampfit Batch Analysis client interface can be copied onto the Windows clipboard for pasting into other applications, or printed directly from the window where they are contained.

Alternatively, data from the results spreadsheets, graphs, lists and trials can be copied to the Layout Window dialog and printed from there.

Printing and copying apply just to the window and sheet in view, for example, if you have results graphs for multiple trials you can only copy one graph at a time. If you want to get data for all the graphs in one step, consider using **File > Export Results**.

If you copy a graph from the Results window you may need to use Paste Special in the application you copy it into in order to have the graph, rather than its raw data, appear.

All printouts have the date and time of printing on every page. Additional header and footer information can be included for printouts from the **Layout Window** dialog.

You can preview pages you are preparing to print in the **File > Print Preview** dialog.

The printable items in Clampfit Batch Analysis are:

- **Trials lists:** these lists print showing the columns configured in the interface, using the same column widths. It is good to use **View > Column Widths > AutoFit All** before printing to ensure all data will be in view.
- **Trials**, as configured in the **View > Command History > Properties** dialog for display in the **Trials** tab, and then in the **File > Page Setup** dialog to select which elements to print.
- **Results sheets:** prints the graph or spreadsheet, whichever has focus. The spreadsheet is printed as 'what you see is what you get' (WYSIWYG), so ensure the columns are widened to show all data before printing.
- **Layout Window:** this dialog has **Page Setup** and **Print** commands. Headers and footers can be configured for printouts.

## Create Datasets

After files are imported into Clampfit Batch Analysis, use the **Data Selector** to query the database for data.

The Data Selector is a user configurable tree hierarchy of database attributes. To configure the hierarchy use **Configure > Data Selector Hierarchy > attributes**, or open the same dialog from the **Data Selector** popup menu (see [Attributes on page 21](#) for further information about attributes). You can set:

- The attributes used in the tree (any number).
- The order of the attributes.

For example, the default configuration orders the tree by protocol, trial date, trial name and trial time, but you can quickly elevate the trial date to the top of the hierarchy if this is more convenient, or select any other trial attributes to include.

Double clicking on a node in the Data Selector sends a query to the database to return all data files with the selected node's attribute as well as attributes at higher levels on the same branch of the tree.

The **File > Add Tree Selection to Current Dataset** command (also in the popup menu) allows you to add data from new, different nodes in the Data Selector to the current dataset.

The result of a database query is the creation of a temporary 'dataset', of all matching data files, in a dataset window.

By using the **Data Selector** popup menu option **Open in New Window**, you can create numbers of distinct datasets in separate windows at the same time.

Datasets consist of **Trials**, which are **.abf** raw data recordings.

## Attributes

Attributes are the properties used by the Clampfit Batch Analysis database to store and identify data files. Clampfit Batch Analysis recognizes many attributes. Most can be used to query the database from the **Data Selector**, and these same attributes are available for display in columns in the trials lists.

All the attributes for a given trial can be viewed in the **View > Attributes** dialog (or use the toolbutton in the top right corner of the dataset window) when the item is selected in its list.

To output attribute values in results, or use them in analysis, write them to the results sheets with **Analysis > Get Attributes**.

Attributes are categorized in all the dialogs where they can be browsed, to make it easier to locate them. All the attributes are included in the **All Categories** grouping, and a subset of these also appears in the **Frequently Used** group. In addition, they all appear in one of the remaining categories. Trial level attributes are divided between two categories: **Trial Data** and **Trial Other**.

### Trial Data

The **Trial Data** class has attributes more directly related to the data recorded in a trial.

**Acq. Sweeps** reports the number of sweeps in the trial. 'Acq.' is short for 'acquisition', which is important because this value does not change if a trial is edited and sweeps removed or added.

**Acquisition Mode** refers to the protocol that generated the trial. Possible values are:

- Gap free
- Variable length events
- Fixed length events
- High speed oscilloscope
- Episodic stimulation

Clampex software uses only the last two modes, referred to as **Compound triggered** (for high speed oscilloscope), and **Internally triggered** (for episodic stimulation) in the **Edit Protocol** dialog.

**Amplifier Mode** is the clamping mode, voltage clamp or current clamp, under which the data were acquired.

**Trial: Ra**, **Trial: Rm** and **Trial: Cm** are Clampex software only attributes, reporting the last single recorded value of access resistance, membrane resistance and membrane capacitance prior to the trial.

**Cm Compensation** reports the membrane capacitance compensated for in amplifier settings when the trial was recorded.

**Analog Bandwidth** reports the net lowpass cutoff frequency at which the trial was filtered.

**Hardware Filter** is the lowpass cutoff frequency set on the amplifier, prior to any additional software filtering.

**Stopwatch**, in Clampex software, is the time in seconds from the start of the experiment.

**Trial: ID** is the number of the trial.

## Trial Other

This category has trial level attributes not immediately relevant to the trial data. Many of the attributes here are inherited from the pCLAMP **Data File Index (DFI)**, and report whether certain protocol options were enabled or not. For example, **Digital Output** and **External Tags** are in this category.

**Trial: Time** is the clock time that the trial began recording.

**Trial: Name** is the trial file name.

## Browse Datasets

After you create a dataset by running a query on the database, you can view trials and access information about the data contained in the dataset.

When a trial is selected in its list, it automatically opens in the lower section of the **Trials** tab. If multiple trials are selected, the top most of the selected trials in the list opens (unless a different one of the selected trials was already open).

Since selections on a tab can change while the tab is out of view by changing the selection on the tab in view, the trial or results on view can change while a tab is out of view.

## Viewing Trials

In the default settings, as soon as a dataset is created, the first trial automatically appears. After a trial is open in the **Trials** tab, you can view trial header information from **File > File Properties** (or use the tab button).

You can open one or more trials into Clampfit software (**Analysis > View in Clampfit**). Trials cannot be returned to the database from Clampfit software.

## Viewing Attributes

All attributes are available for inclusion as columns in the trials, lists (**Configure > Select Columns**, or popup menu).

After you configure the list columns with selected attributes you can order the lists on one attribute by clicking its column title. Click again to reverse the order.

You can order on more than one column, to order one attribute within a higher level ordering of another. When a list is refreshed, ordering is persisted on a maximum of two columns, so if rows were ordered on three columns, it is possible to lose the ordering at the lowest level.

All the attributes for individual selected trials can be viewed in **View > Attributes**, or use the toolbar button in the top right of the dataset window.

All of the lists can be copied and printed.

## User-Defined Columns

The dataset lists all allow a group of selected attributes to be reported in a single column, in order to help save space in the lists. One user-defined column can be created for each list. For example, rather than showing the two attributes **Trial: ID** and **Acq. Sweeps** in separate columns you can report both values in one column, separated by a comma. You can also name the column.

To configure a user defined column:

1. Go to the list that you want to configure on the **Trials** tab.
2. Open the **Configure > Select Columns** dialog.

3. Select the **Available attributes** that you want to appear, and move them into the **Contents of <User defined> column** field.
4. In the **Select attributes from category** field, select **All Categories**.
5. Select **<User defined>** from the main left hand field, and shift into the top right hand field.
6. Position the **<User defined>** column where you want it to appear relative to other columns with the up and down arrows.
7. Click **OK**, and the new column displays in the list with title **<User defined>**.

## Sorting Data

A number of attributes can be displayed in columns in the trial lists to help keep track of where and when Clampex software data were generated.

## Refining and Saving Datasets

Following the initial creation of a dataset from a query, you may want to check the data before analyzing it, removing poor recordings and ‘tidying up’ recordings with spikes or anomalous traces.

There are two ways to remove items from a dataset:

- Directly, with **Dataset > Remove Selected Trials from Dataset**, or use the **<Control + Delete>** keys or the list popup menu command.
- With the **Analyze** parameter; this has the advantage of allowing you to go through the entire dataset marking the items you want to remove, then removing them in one step. This is described in the following subsection.

As an alternative to removing items from a dataset, you can select a subset of the trials and create a new dataset with these, with **Dataset > Create Dataset from Selected Items in New Window**.

## Using Analyze Buttons to Review Datasets

A convenient way to review a dataset is to use the arrow and **Analyze** buttons at the top left of the **Trials** tab.

1. Scroll through the trials using the arrow buttons.
2. If you want to exclude a trial, click the **Analyze – Exclude Selected** button. The trial now shows **No** in the **Analyze** column.
3. Continue scrolling through all the trials in the dataset, clicking **Analyze – Exclude Selected** for all the trials you want to remove from the dataset.

When you have looked at all the trials, click the **Remove Analyze “No” Items from Dataset** button. All the **No** trials are removed from the dataset.

## Trial Editing

Trials are modified using the commands in the **Trial Editing** menu. See [Trial Editing Menu on page 33](#) for a description of individual trial-editing commands.

Most of the trial editing commands apply to all the trials in the dataset marked for analysis, for example, that have value **Yes** in the leftmost **Analyze** list column. When a dataset is first opened, all the trials are marked for analysis. You can change this with the **Analyze** buttons in the left hand side of the dataset window title bar, adding or removing specific trials from the following trial editing commands.

When you open the **Trial Editing** menu, the commands that apply to all of the **Analyze Yes** trials are listed in the left of the menu, marked **Selected for Analysis**.

In contrast, two commands on the right of the Trial Editing menu are marked **Selected Trial Only**. These commands—manual baseline adjustment and trace removal—require user input specific to individual trials, and so are automatically applied only to the trial currently open in the **Trials** tab.

As well as applying just to the open trial, the two **Selected Trial Only** commands cannot be included in macros. Thus, if you are recording a macro when you run one of these commands, the command is applied as usual, modifying the trial and being recorded in the **Command History**, but the step is excluded from the macro recording.

### Trial Versions: Unsaved (with Asterisk) and Saved

After a trial is modified the dataset window refreshes to show the modified version in the trace display. Although the modification is applied and the result can be viewed, the modifications are not yet saved. This status is indicated with an asterisk in the **Analyze** column.

Trials with unsaved changes, for example, those marked with an asterisk, can be used in further analysis steps. For example, after filtering a trial, you can take measurements from it, for example, from the filtered data, even though the filtered version is not yet saved. This allows you to explore the consequences of different trial modifications without having commit to them.

However, if you generate results from unsaved modified trials and you then want to save the results, you must save the trial modifications first, results are always saved with the data from which they were generated.

When you finish with your trial modifications, you can save them before running further analysis. Save trial modifications with **Dataset > Save Modified Trials**, or use the toolbutton in the top right group in the dataset window.

**Save Modified Trials** applies to all modified trials in the dataset, saving all these in one operation. You can enter a comment to be recorded with the new trial versions, or select from previous comments you have used. If you want to comment on each modified trial individually, you must save each trial after modifying it before modifying another.

When you save a new trial version, the comment that you record is written into the file header of the modified trial. This is most conveniently viewed from the **Revision History** dialog, opened from the **File** menu or trials list popup menu.

When a new trial version is saved, the original version remains unchanged in the database. However, the revised version now becomes the version available for new queries on the database. It is possible to revert to the original version. If you create further versions, copies of each version are always retained, and reversion to any earlier version is always possible.

### Revision History

Management of trial versions is all done in the **Revision History** dialog, **File** menu, or **Trials** list popup menu. In this dialog you can view all the versions of a trial, and open any of them into Clampfit software. The **Make Current** button returns you to an earlier version of a trial, making it the version returned from new queries.

When you use **Make Current** to 'revert' to an earlier version of a trial, the 'reversion' actually creates a new version that is a copy of the 'reverted' to trial.

The number of versions of any trial and the number of the current version can be displayed in trial list columns like any other attribute, called **Version Total** and **Version Number** respectively.



When a new version of a file is saved, it inherits all the attributes of the original. In almost all cases these remain correct, but if sweeps are removed, this is not reflected in the **Acq. Sweeps** attribute value, which continues to report the original number of sweeps.

If a dataset is saved with a particular version of a trial and that trial is later replaced by a newer version, the version in the saved dataset does not change.

## Saving Datasets

When you are happy with the contents of your dataset, you can save it with **Dataset > Save Dataset**, or use the tool button in the right of the main dataset window title bar.

Saved datasets are displayed on the **Datasets** section of the **Data** tab, below the **Data Selector**. Like the **Data Selector**, the datasets are placed into a configurable hierarchy, however with only two possible levels above the datasets:

- Project
- Creation Date (creation date of the dataset, not when the data contained in the dataset were generated)

At least one of these must be selected for the hierarchy.

Saved analysis results are always generated for a particular (saved) dataset, hence result set nodes appear beneath the dataset they belong to on the **Datasets** tree.

After a dataset is saved it cannot be deleted or altered.

## Analysis

Analysis is carried out on the trials in a dataset, and on results of previous analysis steps in the **Analysis Results** and **Dataset Results** tabs, using dialogs in the **Analysis** menu. Sequences of analysis steps can be tied together in macros. If you use a particular procedure to generate data in your acquisition system it should be possible to create a macro to perform your complete analysis on all the data generated by that procedure.

Analysis begins with data traces in trials on the **Trials** tab. These might be raw traces, or modified (saved or unsaved) trial versions (see [Trial Editing on page 23](#)). Results from analyses carried out on trials are written to the **Analysis Results** sheets. Further analysis steps can take these results as input, always writing results back to the same sheets that the input came from.

At some point of your analysis you might need to bring together data from the different analysis results in the dataset, for example to average the responses from replicates.

Use either of the following options:

- **Copy to Dataset Results**—copies selected columns from each analysis results sheet to the **Dataset Results** sheet concatenated into one column on the **Dataset Results** tab.
- **Copy to Dataset Results (group by trial)**—copies selected columns from each analysis results sheet to the **Dataset Results** sheet broken out into separate columns by trial on the **Dataset Results** tab.

You can average the values or apply other analyses to the combined data.

Any number of graphs can be created to plot results data columns.

Sections below describe how analysis is applied in Clampfit Batch Analysis in more detail. See [Analysis Menu on page 36](#) for a description of individual analysis commands.

## Analyze All / Some

By default, all the trials in a dataset are acted on when a trial-directed analysis is performed, for example, when peak currents are recorded with the **Measurements** dialog. This is indicated by the **Yes** in the left-most **Analyze** column in the trials list. You can, however, select particular trials to apply the analysis to, by using the **Analyze** buttons to the right of the arrow buttons in the dataset window title bar.

Once you have results data written to the spreadsheets, further analysis steps take these output columns as their input. As for analysis performed on trials, analysis steps that take spreadsheet data as input can be applied selectively, using the **Analyze** buttons on the **Analysis Results** tab.

So, for example, you can apply Measurements to all the trials in the dataset and plot peaks over time. You could then review the graphs and mark the cells with poor results with **Analyze** value **No**, with the **Analyze – Excluded Selected** button. These trials, and their results so far, remain in the dataset, but further analysis steps do not apply to them.

## Results Sheets

The output of analysis applied to trials is written to the **Analysis Results** tab results sheets. There is one results sheet per trial, each sheet consisting of a spreadsheet and a graph section. Multiple graphs can be configured for each sheet. You can scroll through analysis results sheets by clicking on the trials in the list at the top of the tab, or use the arrow buttons there.

The **Datasets Results** tab, where selected results from the analysis results sheets can be brought together for averaging or other analysis across cells, has a single results sheet with the same features as the analysis results sheets.

Use either of the following options:

- **Copy to Dataset Results**—copies selected columns from each analysis results sheet to the **Dataset Results** sheet concatenated into one column on the **Dataset Results** tab.
- **Copy to Dataset Results (group by trial)**—copies selected columns from each analysis results sheet to the **Dataset Results** sheet broken out into separate columns by trial on the **Dataset Results** tab.

In the **Measurements** dialog you have the option to place results from multiple trials in new columns, or to add the results from the later trials beneath the earlier ones using the same columns. All other analysis types that apply to trials, for example, Current/Voltage, output new columns for each trial.

Results of each new analysis are always written in the next free columns to the right of existing columns—in no case does analysis change values in place. Output columns are always written to the same sheet as their input, with the exception of the **Copy to Dataset Results** command, which transfers selected columns from the **Analysis Results** to the **Dataset Results** tab.

The results spreadsheets allow no direct editing; for example you cannot delete data or type in new values. The only way to change the contents of the results sheets is to apply analyses from the **Analysis** menu.

Results columns have two title cells at the top. The top title spans all the output columns from the particular analysis that generated the results. It numbers the analysis step within the dataset, using the same numbering as appears in the **Command History**, and names the analysis type, for example, **Measurements, Curve Fit Columns**, after the analysis dialog names. The title in the second row names the output in the particular column.

In the case of the **Arithmetic** analysis, you can name the output column yourself, but in every other case column names are preset and cannot be changed.

In many cases the source of the results in a column are identified in the second column header, with a sequence like 'T1S1R2'. These use the following abbreviations:

Abbreviation	Meaning
T	Trial: The 'Trial: Number' attribute that identifies the trial in the analysis results.
S	Signal: The signal within the trial that the result was generated from. Signals are numbered top-down within a trial, as viewed in the Trials tab. N.B. Signals in Clampfit Batch Analysis are numbered from 1, whereas in Clampfit they are numbered from 0, so S0 in Clampfit is S1 in Clampfit Batch Analysis.
R	Search Region: The time region within a trial that the value was generated from. Most analyses allow only one search region, but the Measurements dialog can have up to eight.
t	Trace: Where values are generated per sweep within a trial (for example, Histogram) each sweep, or trace, is identified with this number.

To quickly locate the output of a particular step within an analysis sequence you can click on the node for the step in the Command History. This brings the appropriate tab to the front, with the output columns (or graph) in view.

For most analyses you have the option to 'Write data source information'. This writes two additional columns with identifying information for the input to the analysis.

Where possible, if the output of an analysis contains numbers of columns with only one value, the columns are transposed and displayed as two columns of parameter and value pairs. This saves space on the spreadsheet, but each parameter is still available as a separate 'column' for input into further analysis steps.

### Analysis Output: File Export

Spreadsheet results data can be exported as either comma-separated variable (.csv) files, or Microsoft Excel (.xls) files. The **Export Results** dialog is in both the File and Analysis menus.

**Export Results** is included in the **Analysis** menu because file export can be configured as a step in a macro, so that your results are automatically output as files when the analysis is complete.

You can choose to export results from both the **Analysis Results** and **Dataset Results** tabs, or from either of these. If you select to export in CSV format, one file is written for each results sheet, for example, one file for the **Dataset Results** tab and as many files as there are trials for the **Analysis Results** tab.

If XLS export is selected, a single file with one sheet is generated for the **Dataset Results** tab, and a single file with one sheet per trials for the **Analysis Results** tab results.

### Select Input Columns

Analysis that takes results columns as input have list boxes within the dialogs for selection of the appropriate columns.

Where it is possible to make a reasonable assumption about the input for a given analysis, Clampfit Batch Analysis automatically fills the input columns. When such an assumption cannot be made, however, or when the automatically selected column is wrong, you must open the input column list box and select the correct column. Columns are identified by the two-level column titles used on the results spreadsheet.

Easier than selecting input columns from the list boxes in the dialogs, however, is to preselect the columns directly on the spreadsheet before opening the dialog. Use the graph X-axis and Y-axis column identifiers for this.

For example, if you are going to fit two columns of data, click in the first column to select it and then click the **X** button in the results title bar. Select the second column and click the **Y** button. When you open the **Curve Fit** dialog the two columns you selected are already set in the **Data** tab of the dialog, so you can go directly to other configuration settings.

**X** and **Y** columns can be selected in the spreadsheet from the graph, if the columns have already been plotted. For example, if you are fitting plotted points, click on the plot and then immediately open **Curve Fit Columns**. The **X** and **Y** columns for the plot are selected when you click on the graph, and so these are fed to the **Curve Fit** dialog when it opens.

## Command History

As you apply trial editing or analysis steps, each step is recorded in the **Command History** on the **Analysis** tab. If you apply a macro, this is reported as a single node that can be opened to show its component steps.

Each command step is numbered. These numbers are used in the column titles of results data written to the analysis results and dataset results sheets.

When a macro is run, the macro is numbered as usual following the main numbering sequence, while the steps within it are given a subsidiary number, for example, if the macro is the second command applied to a dataset, the macro is number 2, and its component steps 2.1, 2.2, etc. If the macro contains macros within it, the numbering goes to three levels, and so on.

A single click on a **Command History** node brings the results from that step into view on the results sheet, or shows the graph if it was an **Add Graph** or **Edit Graph** step.

Each **Command History** node stores the configuration settings for the analysis or editing step that it represents. The settings can be viewed in the dialog used to run the step by selecting the node and calling the **Properties** command (double-click, popup menu or **Command History** toolbutton). It is not possible to make changes to the analysis step from a dialog opened from the **Command History**. If you want to make changes to a previous analysis step you must undo back to it and then call the analysis dialog from the main menu to reconfigure the step.

The **View Analysis Set** option in the **Command History** popup menu identifies the trials or the **Dataset Results** tab for which the selected analysis step was done. The database ID attributes are used for this.

When you run a macro the cursors in the open trial in the Trials tab stay where they are, possibly in positions other than ones marking the baseline and search regions used in the macro. The **Show Cursor Positions** command in the popup menu moves the cursors to the baseline and search region positions used in the selected analysis step, provided that these regions were defined using cursors rather than, for example, epochs.

## Undo and Redo

Any number of analysis steps can be undone and, if no changes are made, redone, with the **Edit > Undo** and **Redo** commands.

**Undo** undoes the last step displayed in the **Command History**. If that step is a macro, then the entire macro is undone.

If a series of analyses have accumulated, it is possible, with the use of successive **Undo**, to return back to the start of the analysis session, before the first step. You can then **Redo** all the way back to where you were again. This might be done, if, for example, you have configured an analysis sequence and want to record it as a macro, undo all the way back to the start, begin the macro, and then redo the entire sequence, recording it the second time.

Use **Undo** when experimenting with analysis configurations. It allows you to be able to test a configuration, see the results, and then step back if you want to make changes.

**Redo** is only available while no changes have been made to the analysis; for example, it can only be applied to restore an **Undo** command if no existing analysis step has been changed, or no other analyses have been added.

## Graphs

Any spreadsheet columns containing numeric data can be graphed. Histograms, point, or line graphs can be created. Multiple Y-axis columns can be plotted in a single graph, even when these have different units. Vertical tags can be added to mark X-axis values.

Many analyses automatically generate a graph of their output. If not, the easiest way to create a graph is to select the X and Y columns directly in the spreadsheet, marking these as X and Y with the red and green **X**, **Y** and **Y+** buttons in the results sheet title bar (use the **Y+** button when more than one Y column is required). Then click the **Add Graph** button on the left of the title bar. The **Graph** dialog opens with the axis columns already selected, and the plot for the current trial displayed.

With the axes defined, you can take advantage of the many graph configuration options, to name the graph, axes and plots, change axis settings, or add tags, or click **OK** to generate the plots for the entire dataset.

If you want to add a plot to an existing graph, select columns in the same way but select the **Append to Graph** button.

### Linkage to the Spreadsheet

Graphs remain dynamically linked to the spreadsheet columns represented in them. If you click on a point in a graph this highlights the **X** and **Y** columns for the plot that the point belongs to, and the specific x and y coordinates of the point are further highlighted within this. The point itself is picked out with a dashed crosshair in the graph.

After the X and Y columns of a plot are selected, other points in the plot can be selected in the graph by clicking on their cells in the spreadsheet.

### Automatic Graphing

The output of many analyses can be automatically graphed. When one of these analyses is applied, a following **Add Graph** or **Edit Graph** step is automatically applied, plotting the output on a new graph or appending it to an existing graph.

## Save Analysis Results

To save the results of an analysis click the Save Results button in the Analysis Results tab results title bar, or the Dataset Results tab title bar. All results, in both tabs, are saved, irrespective of which tab the save command is called from. The Save Results command starts a sequence of save steps:

1. If the dataset contains trials with unsaved changes, the **Save Modified Trials** dialog opens for you to enter a comment to save all the modified trials as new trial versions. You must save the modified trials in order to be able to save the results.
2. If the trials in the dataset have not been saved in a dataset, the **Save Dataset dialog** opens. You must enter a dataset name, and can optionally add a comment.
3. After you name the new results set, it is stored under the dataset the analysis was run on in the **Datasets** tab.

All spreadsheet results and graphs are saved in results sets and the command history so that you have a complete record of the data that analysis was run on, the analysis steps that were applied, the configuration, and the results themselves.

After a results set is saved it cannot be edited or deleted. It can, however, be opened and additional analysis commands run on it. The combined results can then be saved as a new results set.

Any number of results sets can be saved with a dataset. They all share the dataset permission settings.

It is possible to save files of any type into the database, associated with a particular results set. Use the **Attach File to Results Set** command for this.

## Export Results

You can export analysis results as comma separated variable (.csv) or Excel (.xls) files, with the **Export Results** command in both the **File** and **Analysis** menus. This command can be included as a step in a macro, so results are automatically exported. See [Analysis Output: File Export on page 27](#) for more details.

## Macros

Any number of commands from the **Trial Editing** and **Analysis** menus can be tied together as a single analysis sequence and stored in a macro. Some **Trial Editing** commands, however, cannot be included in macros. These are then accessed to apply to new datasets, and to copy, edit, and otherwise manage from the **Macros** pane of the **Analysis** tab.

### Create Macros

Macros are created by recording commands as they are applied to the current dataset. Start recording a macro from the **Macro** menu or **Macro** pane **Record** button.

When recording is started, a macro recording window opens. The window has stop, pause and cancel buttons, and a field for the macro name, which must be entered before recording is stopped and the window closed. The main field in the window displays the commands recorded in the macro.

Command steps build up in the macro window in tree format, in the same way they do in the **Command History**. Importantly, the macro recorder is sensitive to **Undo** and **Redo** commands, so you can safely record while still experimenting with the details of your analysis—if you apply an analysis step you are unhappy with you can undo it, removing it from the macro recording just as it is from the **Command History**.

Frequently, the macro and **Command History** trees will be identical, but this is not necessarily so. Plainly, macro recording might start after some analyses have already been applied. Once recording begins, however, the **Remove Traces** and **Adjust Baseline Manual** commands are not recorded in macros, though they can still be applied (and hence recorded in the **Command History**). These commands are only suitably applied to individual trials, for each of which they need to be independently configured, and so they are not recorded in macros where they would be applied blindly to new trials.

It is possible to run a macro while recording another, incorporating the one as a step in the parent macro. A single macro, then, could consist of a number of sub macros (which could themselves contain further macros). This can be a good way to build up a store of macros; first create small macros for short, commonly used command sequences then use these as components of other macros.

## Run Macros

To run a macro, select it and click the **Run** button. The application of the macro is recorded as a single step in the **Command History**, but you can open this to see the component steps.

### Run Step

You can run individual steps of a macro with the **Run Step** button. If you select the first step and click **Run Step**, the following step is selected after the first command is applied so you can run through the entire macro one step at a time. However, you can also select any individual macro step to run at any time, wherever it falls within the macro.

**Run Step** records the commands in the **Command History** as if they had been opened from the main menu, for example, they are displayed as independent commands not belonging to a macro. The advantage is that the macro step is already configured how you want it.

### Show Cursor Positions

The **Show Cursor Positions** command in the macro popup menu shifts the cursors in the currently open trial to the baseline, and search region boundaries defined in the selected macro step, providing those regions are defined by cursor positions. If the regions are defined by, for example, epoch, the cursors do not move.

## Manage Macros

When a macro is first created it is shown in the **User** folder in the **Macros** pane. The current Clampfit Batch Analysis user, who created it, and administrators, are able to see it here, but no other users. The macro itself is stored in the database, hence its availability to administrators and the current user if she logs in on another computer.

To share a macro more widely, use the **Macros** menu or popup menu option to copy it to the **Shared** folder. **Macros** in this folder are available to all Clampfit Batch Analysis users. It is not possible to share macros selectively as it is with projects, screens, and datasets.

**Macros**, except for those in the sample folder, can be edited insofar as the command steps they contain can be reconfigured. Open the dialog for any of the component steps (popup menu **Properties** command or **Macros** toolbar button) where you can change any of the settings. When you click **OK**, the dialog is saved back into the macro with the new settings. Be aware of the types of changes you make when editing macros in this way because changes that alter the values that are output should not affect other steps in the macro, but any changes that affect the output format, for example, creating additional columns, could break the macro, if later steps rely on the current output format.

It is not possible to add new analysis steps to an existing macro, or to remove steps. If you want to add or remove command steps from a macro the easiest way is to record a new macro, and use **Run Step** to copy in the steps of the existing macro that you want to keep. Leave out steps that you do not want to include, and add new steps by opening the appropriate dialog from the **Trial Editing** or **Analysis** menu.

Macro **Export** and **Import** functionality allows you to share macros with others. Macros can be exported to disk as XML files and sent to a colleague, who can equally easily import the macro into her own database. Access these options from the main **Macros** menu or **Macros** pane popup menu.

## Database

The default database is BatchAnlysis.db. You can create new database files if you want more granular organization. For example, you can create and use databases for individual users or projects. To use an alternate database, you must create the new database and then select it for use. The selected active database remains active until you select a different database. The status bar specifies the name of the active database.

### Create Database

To create a database:

1. Select **Database > Create Database**.
2. In the **New database name** field, type the name for your new database then click **Create**.
3. In the **Save As** dialog, specify a folder location for the file then click **Save**.
4. To use the new database you created, you must select it.

### Select Database

When you want to change the active database, you must select it. Creating a new database does not automatically select it as the active database.

To select a database:

1. Select **Database > Select Database**.
2. In the **Select database to use** field, type the path of or click ... to locate and Open the database file you want to select, then click **OK**.
3. When you see the **Storage Location** message, click **OK**.
4. When the **Storage Location Type** dialog appears, click the **+** (add) button.
5. In the **Create New File Storage** dialog, in the **Display Name** field, type a name for your data file storage folder then click **OK**.
6. In the **Storage Location Type** dialog, for your file storage entry, confirm that under the **Available** heading, **Yes** appears.
7. Click the **Refresh** button to update for **Available** if needed.
8. To finish, select your file storage entry row and click **OK**.



In Clampfit Batch Analysis, the commands that generate new data are all contained in the **Trial Editing** and **Analysis** menus. Commands in the **Trial Editing** menu modify raw trace data in trials, producing new trial versions. Commands in the **Analysis** menu generate spreadsheet data as output, taking either trials or spreadsheet data as input.

This chapter lists the commands available, describing the main features of each.

### Trial Editing Menu

Commands in the Trial Editing menu are divided into two lists:

- **Selected for Analysis:** These commands can reasonably be applied, with the same configuration, to numbers of trials in a dataset, and to new trials in other datasets, in a macro.
- **Selected Trial Only:** These commands require independent configuration for each trial from which they are applied.

Because of these differences, different rules are used for the application of the commands:

- **Selected for Analysis**
  - Applied to all the trials in the dataset with Analyze value 'Yes'.
  - Can be recorded in macros.
- **Selected Trial Only:**
  - Only ever applied to one trial at a time, the trial that is currently open.
  - Cannot be recorded in macros.

**Adjust Baseline Manual**, and **Remove Traces** are the two **Selected Trial Only** commands.

After you edit a trial, it is updated in the dataset but not yet saved to the database. You can view the changes you made and analyze the modified versions. When you finish the modifications you can save them with **Dataset** menu **Save Modified Trials**.

If you save any results while you have unsaved modified trials, you are required to save the modified trials as well.

### Data

Many of the **Trial Editing** commands allow selection of the signal and traces that will be edited (some automatically apply to all signals and traces, or to all the traces in one signal). The options include:

- **Active signal:** Takes the number of the selected signal in the current trial, and then uses that number signal in all trials, such as the first signal in each, irrespective of name.
- **Specify a signal:** Specify a signal name from the current trial, then the trial of that number, not necessarily the same name, in other trials is used.
- **All visible signals:** All the visible signals in the current trial, and signals of the same numbers in other trials.
- **All signals (including hidden):** All the signals in all the trials.
- **All visible traces:** The numbers of the visible traces in the current trial, in all of the trials. If later trials have more traces than the current trial, the additional traces are not included.
- **All traces (including hidden):** All the traces in all the trials.
- **Active trace:** The number of the active trace in the current trial, in all trials.

- **Select from list:** The same selected trace numbers in all trials where they exist.
- **Specify a list:** As for 'Select from list'.

### Adjust Baseline Specified

Use this command to adjust selected traces by adding or removing a component. The most common application is to remove an unwanted DC component or a steady baseline drift prior to further analysis. Entire traces are corrected.

**Subtract mean of** computes the mean of the specified region and subtracts this. This method is useful for removing the DC component.

**Subtract slope of** computes a slope of the specified region and subtracts this. The slope is computed using linear regression. This option is useful for compensating for a steady baseline drift, in which case the specified region for computing the slope is generally the full trace.

### Arithmetic (Trials)

Use **Arithmetic** to perform arithmetical operations on selected data traces. Operations can be also applied to sections of traces.

Enter an expression in the top field to define the arithmetic operation. First enter the trace or traces to be modified, followed by an equals sign and then the operation to be performed on the traces, typically referring to the same and/or other traces in the right side of the equation.

Traces are identified with 't' prefixes. You can type these in directly, or select them with the options below and use the **Assign** button. Operators and functions can be typed in or selected from their lists.

Descriptions of the functions and constants available for **Arithmetic** expressions are given in the **Analysis** menu **Arithmetic** section. See [Arithmetic on page 41](#).

### Average Traces

The **Average Traces** command allows you to select traces to average in a trace display or by trace number in a list. The averaged trace is added as a further trace at the end of the trial, immediately following the last acquired trace.

You can optionally output a further trace of the standard deviation.

The currently open trial is displayed in the dialog, where you can choose which, or all, signals to view. However, when the command is applied, the selected traces in all signals are averaged.

The same trace numbers are averaged in each trial selected for analysis across the dataset.

To help with trace selection you can drag along the graph axes in the graphical display to zoom in, and use buttons there for other scaling options. Additionally, to help select closely packed traces, use the vertical separation option to space traces apart. If you select an incorrect trace, you can undo the selection. Only one selection can be undone.

### Filter

The **Filter** dialog has options for **Highpass**, **Lowpass**, **Bandpass**, electrical interference and notch filtering. A diagram representing the cutoff characteristics of the selected filter is displayed in the lower right hand corner of the dialog.

You can also filter a selected region of the traces, or entire traces.

The sampling frequency, sampling interval and number of data points per trace are reported at the bottom of the dialog.

See [Digital Filters on page 53](#) for full explanation of the filtering options in Clampfit Batch Analysis.

## Electrical Interference

The **Electrical Interference** filter identifies and removes complex 50 or 60 Hz power line waveforms, composed of multiple harmonics.

The filter is designed for reasonably long continuous data files of several seconds or more, and will generally not work well with short files.

The EI filter will be disturbed and take time to recover after a transient signal that has harmonics overlapping the harmonics of the steady state interference signal. The disturbance will take the form of an introduced cyclical noise that lasts for the duration of the 'cycles to average' parameter.

You can specify the highest harmonic of the interference waveform to be removed. It is often sufficient to remove only the first harmonic, or the first few harmonics, resulting in faster execution speed.

The cycles to average control specifies the time constant of the filter, expressed in line interference cycles (periods). In case of 50 Hz line voltage, one unit is 20 ms, while in case of 60 Hz it is 17 ms. The filter removes interference more effectively with a higher number of averaging cycles, at the expense of execution speed.

The reference frequency must be the same as the power line interference frequency. If Auto is selected the EI filter automatically determines the frequency so long as it is either 50 Hz or 60 Hz. Use **Auto** when you are not sure of the interference frequency.

## Force Values

This command modifies the contents of a selected region of the selected traces. It is typically used to eliminate the effect of transients before lowpass filtering.

The analysis applies to all **Analyze 'Yes'** trials in the dataset. If you want to remove a transient in one trial only be sure to select it alone for analysis.

The region selected in the **Region to force** takes the values configured in the main section of the dialog. You will often need to position cursors in the current trial for use in the configuration options in the dialog.

In typical usage to remove a transient peak, position cursors 1 and 2 on either side of the peak and then in the dialog select cursors 1 and 2 as the region to force, and straight line between cursors 1..2 as the new value. A straight line is drawn between the cursors, eliminating the peak.

## Subtract Last N Trace Average

This command takes the last N traces in each trial and averages these. Then the averaged trace is subtracted from each trace in the trial. The averaged trace is not itself written, so the resulting trial has the same number of sweeps as the original.

**Subtract Last N Trace Average** applies to all traces in all signals.

## Time Shift

Use **Time Shift** to change the alignment of the data with respect to the time base. The operations of this option either align the data peaks or shift the data traces by a specified amount along the time axis. You can rotate points when traces are shifted, or replace these points with zeros.

## Adjust Baseline Manual

Use **Adjust Baseline Manual** to adjust for baseline rundown manually. The dialog opens with one signal of the current trial displayed. You can change the signal in the top right list box. Use the cursor to drag the purple baseline marker as appropriate. All traces in the selected signal are adjusted to the new baseline when the dialog closes.

## Remove Traces

Use **Remove Traces** to select traces to remove a trace display, or by trace number in a list. It can only be applied to individual trials, and is not included in macros.

The currently opened trial is displayed in the dialog, where you can choose which (or all) signals to view. However, when the command is applied, the selected traces in all signals are removed.

Selected traces are highlighted, accumulating as you select additional traces. To update the view by removing those traces already selected, click **OK** and then reopen it.

To help with trace selection you can drag along the graph axes in the graphical display to zoom in, and use buttons there for other scaling options. Additionally, to help select closely packed traces, use the vertical separation option to space traces apart. If you select an incorrect trace you can undo the selection. Only one selection can be undone.

## Analysis Menu

Commands in the **Analysis** menu are all macroable, and output their results to spreadsheets in the **Analysis Results** or **Dataset Results** tabs. For a general discussion of the commands see [Analysis on page 25](#).

In the menu, various 'special ease' commands are shown in their own sections, while the remaining commands are divided into three sections:

- The group starting with Measurements all apply to trials. These commands all take some sort of measurement from data traces.  
You may be able to select particular signals and traces to apply these commands to (depending on the analysis); see [Data on page 33](#) for a description of these options.
- The group starting with Arithmetic takes spreadsheet columns as input.  
You can select the input columns from list boxes in the analysis dialogs when you open them, but more convenient is to select the columns in the spreadsheet first, marking them as X and Y columns with the spreadsheet buttons. These columns are then populated in the dialog when you open it.

The following describes individual Analysis menu commands.

### Get Attributes

This command retrieves attribute values from the database and writes them to results sheets.

If you call **Get Attributes** when the **Trials** tab is in view you must select whether to write the attributes you will select to the **Analysis Results** sheets (per trial) or to the **Dataset Results** tab.

The dialog closely resembles the **Data Selector** tree configuration dialog, and the list column configuration dialogs—listings of all database attributes appear in the left hand field, which you select by moving them to the right hand field.

Each selected attribute is written to its own column in the results sheets. Depending on the scope of the attribute, it will have a different number of values. For example, **Trial** attributes such as **Trial: Ra** or **Protocol** have as many values as there are trials.

By default, the **Repeat values to align rows** check box in the dialog is enabled. If single-value attributes are selected along with trial attributes (like **Protocol**, or **Acq. Sweeps**), of which there may be many, the single attribute is repeated to align with the rows of the trial attribute. If unchecked, then the single-value attribute is written once only. Single-value attributes can be repeated to align with multiple trial attributes.

When attributes are written to the **Dataset Results** tab, trial values are automatically written as well, so that you can identify where particular values came from within the columns.

## Add Graph/Edit Graph

The Clampfit Batch Analysis **Graph** dialog provides multiple plots, histograms, and tags, and offers many configuration options. As you configure a graph, a display in the bottom of the dialog shows how it will look.

For an overview of graph functionality, see [Graphs on page 29](#)—this section explains the main features in the dialog.

The most important selection of **X** and **Y** columns to plot is performed on the **Plots** tab. Very often, with automatic graphing, these are automatically selected, or it is most convenient to select **X** and **Y** columns directly in the results sheet before opening the dialog. Otherwise, select **X** and **Y** values from a listing of all results columns in the appropriate list boxes.

Add additional plots to a graph with the **Add** button under the **Plot** field on the left of the tab. Name it on the right, and then select the **X** and **Y** values.

Error bars are added to a graph simply by selecting an appropriate column, for example, of standard deviation values for an average value selected for the **Y** axis.

Once you have defined a plot use the **Plot Properties** section of the **Plot** tab to define its appearance.

The **General** tab has graph naming, legend, gridline and border configuration options. For graph names you can type in a name, to appear on all graphs, or select a results sheet column.

The **Axis** tab allows you to name graph axes, including units, and has many other configuration options including log and inverted scales, zero-crossing axes, and showing time in minutes.

You can define set axis ranges that will apply to all the graphs generated, or check **Show Zero** to always include zero in the axis range, whatever the data range on that axis may be.

Tags are defined by selecting the **X** axis values to be marked—as usual, by selecting the appropriate results sheet column. Any column can be selected to label the tags. If labels on successive tags are repeated, only the first is displayed.

Tags can be grouped using a third attribute. The tags are then color coded according to their group, and the grouping value used in the graph legend.

If no grouping is selected you can type in your own text for the legend entry for the tags.

Configuration options for the appearance of tags, which can be either full height lines down the graph, or arrows at the bottom, are on the right of the **Tags** tab.

## Measurements

The **Measurements** command can take different measurement types from up to eight different time regions within trials.

A full range of options to choose the signals and traces to measure from is available on the **Data** tab.

On the Measurements tab, select the number of different searches you want to configure with the Search Number spinner and check box. The full range of configuration options on the Measurements tab is available for each search that you enable, for example, for each search you can set:

- Search region (using cursors or epochs)
- Baseline region (or set a fixed baseline value)
- Measurement types
- Peak polarity
- Smoothing

Two options are included to input Measurements settings used during acquisition for example, in the Clampex Software) into the dialog:

- **Import from File:** This button copies the acquisition measurement settings from the currently open trial into the dialog. All options remain enabled so you can alter them if you like.
- **Use acquisition measurement settings:** This check box disables the Measurements tab. When the analysis is run, the acquisition measurements settings for each trial are used. So, if different acquisition measurements were taken from different trials in the dataset, different measurements will be taken from these.  
Use acquisition measurements settings does not populate the **Measurements** tab with the measurements taken on the currently open trial—to see these first click **Import from File**, and then check the acquisition measurements check box.

### Output Options

Since measurements from all the trials are written to the same results sheet, this option determines the placement of these relative to one another. Under the **Columns** option, measurements from each trial are placed in new columns beside one another. Under the **Rows** option, measurements in following trials are added under those from previous trials, in the same columns.

Write data source information writes a number of values to identify the input of the measurements step.

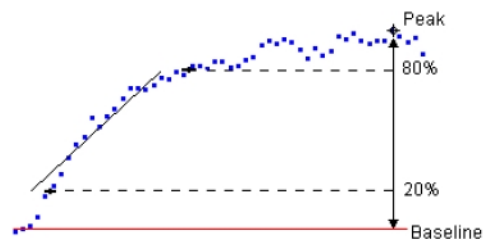
**Automatic Graphing Parameters** identifies the columns selected for plotting in an automatic graphing step following the application of the Measurements command. In most cases you should be able to accept default settings here.

### Measurement Types

Descriptions of the measurement types available in Clampfit Batch Analysis are given below.

- **Peak amplitude:** Finds the peak amplitude relative to the baseline, dependent on the peak polarity setting.
- **Time of peak:** Reports the time of the peak, from the start of the sweep, as well as from the start of the trial. Both measurements are automatically output, even if not requested, for automatic graphing purposes.
- **Anti peak amplitude:** The point furthest in amplitude from the peak.
- **Time of anti peak:** Reports the time of the anti peak.
- **Mean:** Average of the amplitude of all sample points in the search region.
- **Standard deviation:** The standard deviation of the amplitude of all the sample points in the search region.

- Area:** The area bounded by the trace and the baseline, in units of (amplitude × time). In the case of bipolar events the 'negative' area is added to the 'positive' area, so a total area of zero is possible.  
 Area is calculated using simple summation of points within the search region. For example, with a search region from sample ten to sample sixteen, the sample to baseline amplitudes of each of the seven sample points in the range are summed, and multiplied by the sampling interval.
- Half width:** To find the half width the points to the left and right of the peak are scanned until the points (nearest the peak) that are 50% of the peak amplitude relative to the baseline, are found. The time between these two points is the half width.  
 If the samples on one side of the peak never get down to 50% of the peak amplitude within the search region no half width is found.
- Maximum rise slope:** Finds the greatest rising slope (for example, moving towards the peak) between two sample points in the region bounded by the start of the search region and the peak. If two sections of trace have the same increase rate the section nearest the peak is reported.
- Time of maximum rise slope:** The midpoint between the two points during which the maximum rise occurred.
- Maximum decay slope:** The greatest falling (for example, returning to baseline) slope between two sample points in the region bounded by the peak and the end of the search region. If two sections of trace have the same decay rate the section nearest the peak is reported.
- Time of maximum decay slope:** The midpoint between the two points during which the maximum decay occurred.
- Slope:** The slope within the search region, as found by linear regression.
- Baseline:** If you specified a mean level for the baseline region, the averaged baseline value for the trace in the stipulated region is reported. Otherwise the fixed baseline is reported.
- Rise slope:** The slope, from within the region bounded by the start of the search region and the peak, between the points determined by the associated 'from' and 'to' percentiles. Points to the left of the peak are scanned until points at the nominated percentages of the peak amplitude (relative to the baseline) are found. Regression is applied to these and all intervening points to give the rise slope (see [Figure 3-1](#)).



**Figure 3-1: Rise slope from 20% to 80%.**

If the samples on the left side of the peak do not get down to either of the nominated percentages before the left boundary of the search region, the rise slope is not found.

- Rise time:** The time between the associated 'from' and 'to' percentile points (see Rise slope above for how these points are determined).

- **Decay slope:** The slope, from within the region bounded by the peak and the end of the search region, between the points determined by the associated ‘from’ and ‘to’ percentiles. Points to the right of the peak are scanned until points at the nominated percentages of the peak amplitude (relative to the baseline) are found. Regression is applied to these and all intervening points to give the decay slope (see [Figure 3-2](#)).

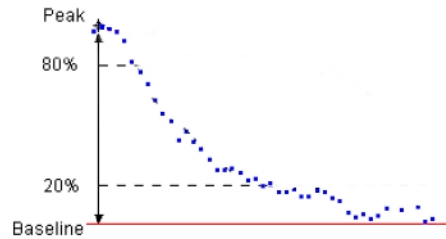


Figure 3-2: Decay slope from 80% to 20%.

- **Decay time:** The time between the associated ‘from’ and ‘to’ percentile points (see Decay slope above for how these points are determined).
- **Event duration (ms):** The time difference between the beginning and the end of the response determined by the specified percentage of the peak amplitude (see [Figure 3-3](#)).

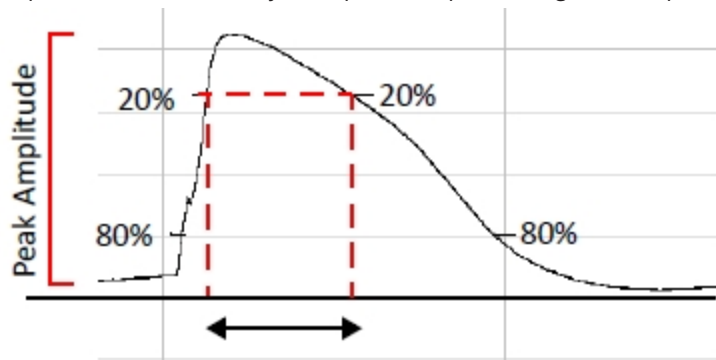


Figure 3-3: Event duration at 20%.

## Current/Voltage

**Current/Voltage** finds the stimulus waveform voltage at a specified point or region of the trials it is applied to, and measures the current at another (or the same) point or region.

Configuration of I V and H infinity plots is straightforward with the command—identification of voltage settings with the **X Axis** and current settings with the **Y Axis** reinforces this. As with other commands in Clampfit Batch Analysis, if you have automatic graphing enabled (**Program Options** in the **Configure** menu) the plot will be automatically generated after you apply **Command/Voltage**.

With the waveform signal identified, the voltage can be taken at a specific cursor position, or from an epoch. If the command signal is included as a signal within the trial, the voltage can be read from this as well.

The **X Axis** measurement is not limited to voltage only—the epoch duration can also be used, as well as any user list values you have set.

Current measurement from a selected signal can be taken at a single point, or be the mean or peak within a defined region. The command has an optional boxcar filter to smooth points in the traces before taking the current measurement. You can set the number of points smoothed.



Both the voltage and current measurements can be inverted if you wish. **Current/Voltage** measures current from one signal at a time.

### Curve Fit Trials

Use **Curve Fit Trials** to fit to selected regions of selected data traces in trials. There is a wide range of predefined fitting functions to choose from, or you can enter a function of your own.

Not all predefined functions are available for fitting trace data. Functions with more than one independent variable or negative X axis values can only be fitted to results sheet data. All configuration options in the dialog are disabled if a non-applicable function is selected.

The number of data points between the cursors cannot exceed 110,000.

**Curve Fit Trials** uses the same dialog as **Curve Fit Columns**, with the **Data** tab modified for selection of traces rather than columns.

See [Curve Fit Columns on page 43](#) for a description of options in the dialog.

### Histogram Trials

**Histogram Trials** bins individual data points by amplitude. You can apply the command to selected signals and traces, and to a specified region of the trace (set on the **Data** tab).

In other respects, **Histogram Trials** is the same as **Histogram Columns**. See [Histogram Columns on page 46](#) for full description of the command.

### Trace vs. Trace

**Trace vs. Trace** allows you to plot the points of a single selected trace against points at the same time of the sweep in another trace. Comparison can be made between traces in the same or different signals (within each trial). You can apply the analysis to the entire sweep length, or a specified region.

### Arithmetic

The **Arithmetic** command in the **Analysis** menu applies to results sheet data on the **Analysis Results** or **Dataset Results** tabs, unlike its close relative in the Trial Editing menu, which applies arithmetic to traces in trials.

In the **Analysis** menu command output is always to a single new column in the results sheet, therefore you only have to enter the right hand side of the arithmetic equation.

Use capital letters to identify results sheet columns, as shown in the main field in the dialog, or select a column you want and click the Assign button to write it into the expression. Type functions, operations or constants directly into the expression or select these from their lists.

Only columns with the same number of rows can be combined in **Arithmetic**. You can also limit the input to a specific row range. The same row range is applied to all the columns used in the expression.

The **Arithmetic** command is unique in Clampfit Batch Analysis in that it allows you to write the second row title for the output column. If you do not put anything here then the title defaults to the arithmetic expression. This feature is useful to write a meaningful description as the column title, making it easier to understand results when you later review them.

The following functions are available in **Arithmetic**:

**Table 3-1: Arithmetic Functions Descriptions**

Function	Description
arcos()	Arccosine: the inverse cosine of a number in the range of -1 to +1. The result is in radians.
arsin()	Arcsine: the inverse sine of a number in the range of -1 to +1. The result is in radians.
artan()	Arctangent: the inverse tangent of a number in the range of -1 to +1. The result is in radians. Cannot be used with odd multiples of $\pi/2$ .
cos()	Cosine: the cosine of a number in radians.
diff()	Differentiate: the derivative. The result is strongly affected by noise in the data; consider filtering before use.
exp()	Exponentiate: e raised to the value of the number.
intg()	Integrate: the continuous integral. The result is strongly affected by the baseline.
inv()	Inverse: the multiplicative inverse.
ln()	Log <sub>e</sub> : the logarithm to the base e. Cannot be used with zero or negative numbers.
log()	Log <sub>10</sub> : the logarithm to the base 10. Cannot be used with zero or negative numbers.
norm()	Normalization: the data are scaled and offset to a Y axis range of 0 to 1.
normax()	Normalization: the data are scaled by being divided by the largest value in the column.
normin()	Normalization: the data are scaled by being divided by the smallest value in the column.
ramp()	Ramp: the data are scaled by a steadily increasing factor from 0 to 1.
rect()	Rectify: replace numbers with their absolute values.
sin()	Sine: the sine of a number in radians.
sqrt()	Square root: the square root. Cannot be used with negative numbers.
tan()	Tangent: the tangent of a number in radians.

Also available in the **Constants** list are:

Constant	Description
time	the time value of the point.
maxtime	the maximum time of the destination trace
pi	pi

### Average Rows

**Average Rows** allows you to find the arithmetic or geometric mean, and/or the median, of a selected column. Standard error, sample and population standard deviations, and sample size can all also be derived.

You can isolate particular values in the column selected for averaging by reference to the values in one or more other columns.

## Curve Fit Columns

Use **Curve Fit Columns** to fit results sheet data or graph plots. There is a wide range of predefined fitting functions to choose from, or you can enter a function of your own.

The dialog has four tabs:

- **Function:** Select the fitting function, basic settings and model comparison to find the best number of terms.
- **Advanced:** Fitting methods, preprocessing and fitting strategy, as well as output information options and fit extrapolation. Many of these settings can be set automatically or left at defaults.
- **Seed Values:** Entry of seed values when required, as well as graphical seeding.
- **Data:** Selection of the columns to be fitted.

The formula of the selected function is displayed at the bottom of the dialog, visible from all tabs. The formula name and other information are reported in a **Status Bar** above the formula.

The following sections have information specific to options in the dialog. See [Curve Fitting on page 77](#), and [Fitting Functions on page 103](#), for further information about fitting in Clampfit Batch Analysis.

### Function Tab

Select the fitting function from the list of predefined functions, or configure your own custom function. The predefined functions are divided into groups to make it easier to find a specific function.

### Find Best Number of Terms

For functions that support multiple terms, the compare models option can be used to find the best number of terms at a specified confidence level. With automatic searching, models for up to 6 terms are tested. For least squares fitting the confidence level refers to the F-statistic. For maximum likelihood fitting it refers to the Chi-square statistic.

### Custom Function

A fitting function of up to 255 characters can be typed into the Custom Function field. An example showing the proper format is provided in the dialog. See [Define a Custom Function on page 97](#).


The following operators can be used in custom functions:

abs	absolute value	log	the logarithm to the base 10
acos	arccos	log10	the logarithm to the base 10
asin	arcsine	pow10	10 to the power
atan	arctangent	ramp	scale by a steadily increasing factor from 0 to 1

cos	cosine	random	random number generator (requires dummy variable)
diff	differentiate	rect	absolute value
exp	exponential	round	round up to the nearest integer
fact	factorial	sin	sine
intg	integrate	sqr	square
inv	reciprocal (1/x)	sqrt	the square root
ln	natural logarithm (for example, log to the base e)	tan	tangent

### Advanced Tab

Search, minimization and weighting methods are configured on the **Advanced** tab, as are other fit application and output options. Also see .

 **Tip:** Unexpected application of zero-shift is a common cause of failed or inappropriate fits. Always check this first, in the **Status Bar** above the formula, in these cases.

If you fix parameters and these are not at the fixed values when the fit is completed, check to see if **Auto-free fixed parameters and refit** is on.

The following Advanced tab options are available for enabling:

- **Zero-Shift X-axis Data**—a set of  $i$  data points  $x_i$ , each point  $x$  is offset such that  $x = x_i - x_0$  where  $x_0$  is the value of the first data point (the fitting origin). If data are to be offset, the text '(zero-shift)' appears in the options Status bar in the fitting dialog immediately above the equation window.
- **Auto-free fixed parameters and refit**—automatically frees fixed parameters following convergence and the fit is reapplied with all parameters free.

 **Tip:** Disable this option to keep fixed parameters fixed following convergence.

- **Attempt refit on failure**— seed values are automatically adjusted (up to six times in each direction) and the fit reapplied in the event of fit failure. This often results in a successful fit even if the original seed parameters were well off the mark. No indication of parameter readjustment is provided.
- **Extrapolate fit**—Fits can be extrapolated beyond the range of the data fitted. For trials, set cursors 1 and 2, or 3 and 4, to the region you want to extrapolate the fit to. For fits to results data, specify the X-axis limits for the extrapolated fit. The number of samples in the extrapolation range plus the fitting range cannot exceed 110,000.
- **Zero concentration replacement**—allows the Curve Fitting dialog to accept zero concentration as an input even though concentrations are in a log distribution, and hence zero strictly disallowed. In these cases zero is replaced by a number  $n$  times smaller than the least actual concentration.

## Search Method

The following search methods on the Advanced tab can be set automatically or manually:

- **Levenberg Marquardt** — uses an iterative approach to minimize the difference between the fitted curve and the data. See [Levenberg-Marquardt Method on page 79](#).
- **Variable Metric** — is also an iterative search method that relies on the evaluation of derivatives with respect to each parameter. The method can be slow when fitting to very large data sets but in most cases converges quickly with data sets that are well described by the fitting function. See [Variable Metric Method on page 82](#).
- **Simplex** — uses an iterative technique that does not require evaluation of derivatives. This method usually requires more iterations to converge than the Levenberg–Marquardt or variable metric methods, however, it is less prone to convergence on a local minimum. See [Simplex Method on page 80](#).
- **Chebyshev** — transforms the data to be fit to the Chebyshev polynomial domain before the errors between the fitting function and the data are minimized. It is a faster fitting method than the other options, but the data points must be equally spaced. See [Chebyshev Transform on page 83](#).

## Precision

The precision value is a tolerance level for convergence. The lower the value the more readily the fit converges. However, very low values might result in convergence even though the fit is poor. High values might result in very slow or no convergence.

The following default values provide a good compromise between speed of convergence and fitting accuracy:

- Levenberg Marquardt:  $10^{-6}$
- Variable metric:  $10^{-4}$
- Simplex: 10

## 4 pt. Smoothing Filter

A 4 point smoothing filter can be used with the Chebyshev fit. This filter reduces noise and nonlinear noise effects on the statistics of the fit. The averaging does not alter the accuracy of the fit but it can alter the value of the fitted parameters if the data are noisy. If in doubt as to whether or not to use this filter, fit both with and without the filter and accept the best fit on the basis of the fitting statistics.

## Seed Values Tab

The seed values in the **Seed Values** tab are used for the curves in the **Graphical Seeding** display window. If the function has multiple terms, the component curve for each term displays. The curve representing the complete function, the sum of the terms, is the function curve. Each individual component curve and the function curve are displayed along with the data. The function curve is always displayed, whereas the components curve display is optional.

Seed values are assigned to the parameters by typing their value directly in the appropriate field or by using the field entry spinner. There is a short delay before the graph updates with the new seed parameter values.

Enter all search methods, except the Chebyshev, linear, or polynomial regression, function parameter seed values before fitting. For the Chebyshev method, seed values are computed but must be fixed.

The fitting routine automatically generates seed values for all predefined functions and enters these values in the appropriate locations in the **Seed Values** tab. Seed values must be entered manually for custom functions.

Individual parameters can be fixed using the **Fix** option next to each seed value. When a parameter is fixed, the value is essentially a constant term in the fitting function.

---

**\* Tip:** Often a difficult fit can be made by fixing one or more of the function parameter values.

---

When using **Advanced > Auto free fixed parameters and refit**, the fixed parameters are automatically freed upon convergence, and the fit repeats. See [Advanced Tab on page 44](#).

For custom functions, none of the seed values can be zero. You must assign non-zero values to each parameter for fitting to continue.

---

**\* Tip:** For zero value parameters, either remove it from the function, or specify a very low value, such as 0.00001.

---

The following options are available:

- **Force Parameters Positive**—generates only positive amplitudes.
- **Force proportions positive**—generates only time constants.
- **Constant**—Most predefined functions have a constant Y offset term 'C'. This value is automatically estimated but can be modified or fixed like any other parameter. A value of zero can be assigned to this parameter.
- **Slope**—The **Exponential, sloping baseline** function is the only option that has a slope term. The seed value for this term is initially automatically estimated. A value of zero can be assigned to this parameter.
- **Proportion display** —displays the proportion terms either as absolute values or normalized proportions. This option is available only for exponential probability functions.
- **Multiple Fit Seeding**—when fitting multiple traces or columns in different trials, the initial seed values apply to the first trace or column only.
  - **Seed independently for each fit**—the seed estimates are automatically and independently generated for each trace. Use if the successive traces are very different from each other.
  - **Use results of current trace to seed next**—the fitted parameter values of the current fit are used to seed the next.

---

**\* Tip:** There can be instances where the automatic seed estimation is not sufficiently accurate to generate a successful fit, especially if the data are sparse, or not well described by the fitting function. Where seed estimates are critical, or where reasonable seed estimates are not automatically generated, graphically assisted seeding can be used to produce very accurate initial values.

---

## Histogram Columns

Bin selected columns of data for display in histograms with this command.

Select the columns to bin on the **Data** tab, where you can restrict the input to a specific row range.

If you select multiple columns these are binned separately by default, but you can pool all the values together if you wish with a **Combine histograms** option (similarly, data from multiple traces and/or signals can be pooled for **Histogram Trials**).

The dialog allows configuration of conventional, cumulative, logarithmic and variable width cumulative plots.

For conventional and cumulative plots you can set either the number of bins or their width, and the other value is automatically calculated. These values are derived using the full data range, which you can limit, if you wish, in the **Data Limits** section of the dialog (this is a further limitation, by data value, to the row limitation option on the **Data** tab).

**Conventional** and **Cumulative** plots can optionally be normalized by area.

**Logarithmic** histograms scale the parameter of interest on a base 10 log scale along the X-axis. All values must be greater than zero or logarithmic graphs. Bin width is set with the **Bins/Decade** value, which determines the number of bins you want between each whole number in the X-axis logarithmic scale.

With variable width cumulative graphs (also sometimes referred to as 'cumulative unbinned') the bin width is determined by the value of the data themselves. A scatter plot is created where each point represents, on the Y-axis, the number of data less than or equal to the X-axis value of the point.

## Normalize Columns

**Normalize Columns** normalizes the values in a selected column. You can normalize between 0 and 1, or if the data you are normalizing are negative (for example, negative-going peaks) and you want to retain this, you can normalize between  $-1$  and 0, the lowest value going to  $-1$  and the highest to 0.

Additional columns can be normalized at the same time as the principal column, in which case they are normalized to the same range as the principal column. Since values in additional columns might lie outside the range of the principal column values, the output columns for the additional columns might lie outside the range 0 to 1 or  $-1$  to 0.

## Copy to Dataset Results

The **Copy to Dataset Results** dialog copies data from the **Cell Procedures** tab results sheets to the **Dataset Results** tab.

Analysis on the **Cell Procedures** sheets proceeds per cell procedure, for example, all steps are applied within the respective cell procedure sheets. At some point of your analysis you may want to bring data from the individual cells together, for example to average the results from replicates, or simply to have a convenient report of final results from each cell, in one location.

To select columns to bring to the **Dataset Results** tab simply double-click on them in the left-hand field in the dialog. The selected columns are added to the right-hand field, where you can order them with the arrow buttons in the right margin.

The contents of a selected column are copied from the cell procedure sheets and written in one column, the values from each cell procedure placed underneath one another in succeeding rows. Columns with different numbers of rows on the cell procedures sheets, then, generate output columns with different numbers of rows, and the rows in different columns only align if they came from columns with the same number of rows.

If any of the cell procedures did not have a value in a selected column, the blank cell is written to the **Dataset Results** sheet.

The same cell procedure order is used down the columns for all columns that are copied.

As a quick illustration of these rules, if there are 10 cell procedure sheets and a column with one row is selected, a single column with 10 rows is output. If a column with four values per cell procedure is copied then a column with 40 rows is generated. The four values from the first cell procedure are at the top of the column, followed by the four values from the next, and so on.



Clampfit Batch Analysis functionality is part of the pCLAMP software Advanced Analysis license. After the pCLAMP software is installed, Clampfit software requires file server and data storage configuration to use Clampfit Batch Analysis functionality.



**Note:** To run Advanced Analysis Clampfit software, which includes Batch Analysis functionality, you must have the Advanced Analysis Clampfit software USB dongle installed on your computer.

Use the following configuration procedures.

## Configuring MDC File Server

The MDC File Server runs on Windows 7 and 10 computer operating systems. The computer it runs on requires enough hard drive space to store your data files. The storage location can be changed if it becomes full, and the database keeps track of data files in multiple storage locations. If you set a new storage location, keep the existing data storage location.

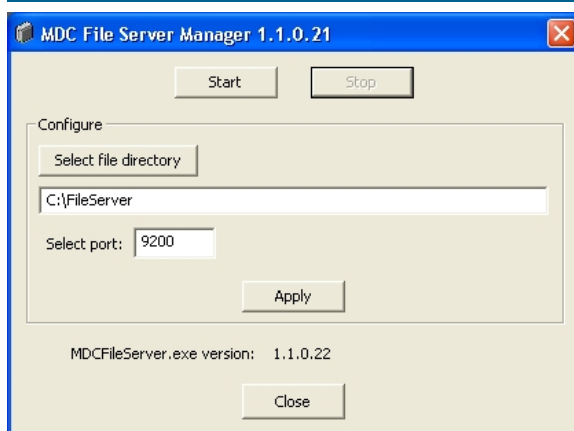
Before starting Clampfit software to run Batch Analysis the first time, you must first configure MDC File Server. MDC File Server is required for managing imported data. It runs quietly in the background while you use Batch Analysis. MDC File Server must be running for Clampfit software Batch Analysis to run.

To configure MDC File Server:

1. After you install the pCLAMP software, open the **MDC File Server Manager** from **Start > All Programs > Molecular Devices > MDC File Server > MDC File Server**, right-click and **Run as administrator**.
2. In the **MDC File Server Manager** dialog, click **Select file directory** to open a **Browse** dialog to select the folder where you want to store data files.



**CAUTION!** If you select a computer other than the local one, you must have access permissions to the folder location at all times.



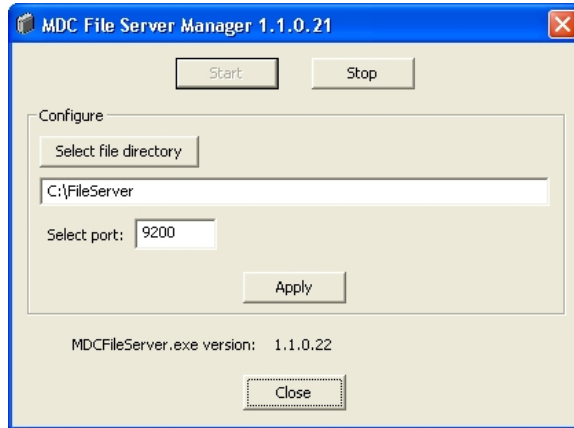
3. Click **Create new folder**, type the name **FILESERVER**, and click **OK**.



**Tip:** You can name the new folder something other than **FILESERVER** if needed.

4. In the **Select port** field, type **9200**, and click **Apply**.

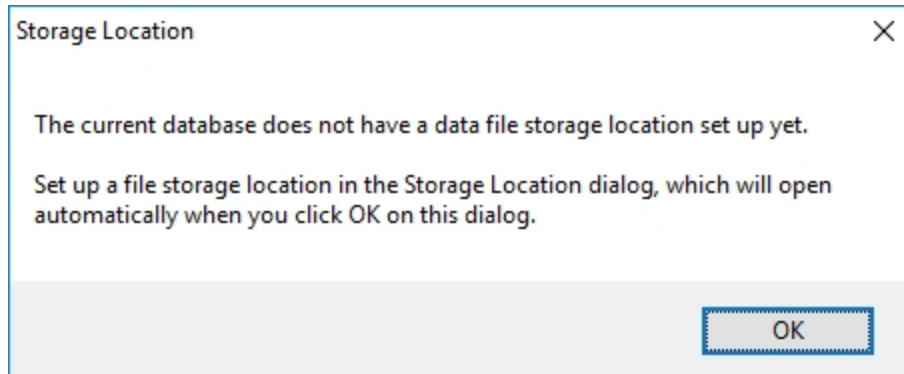
5. Click the **Start** button and wait for the **Start** button to disable.
6. When the **Start** button is inactive, click **Close**.



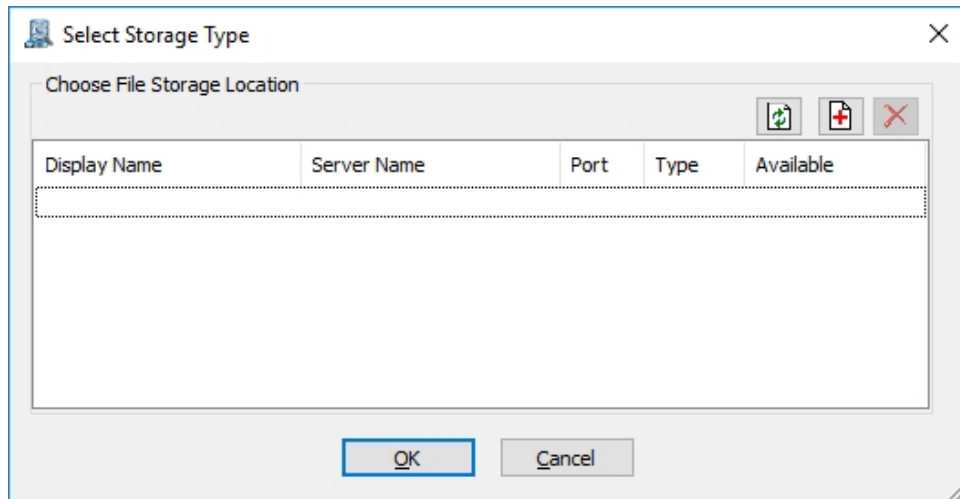
## First Time Use Database Configuration

The first time you use Clampfit software Batch Analysis, you must do the following to configure your data storage folder:

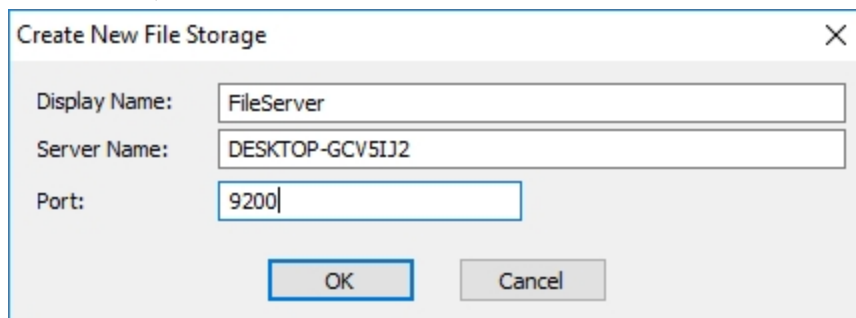
1. Ensure that your Clampfit software license key dongle is installed on the same computer that the software is installed.
2. Start Clampfit software.
3. Select **Analyze > Batch Analysis**.
4. After the **Batch Analysis** window appears, when you see the **Storage Location** message, click **OK**.



5. When the **Storage Location Type** dialog appears, click the + (add) button.



6. In the **Create New File Storage** dialog, do the following:
- In the **Display Name** field, type a name for your data file storage folder. See [Configuring MDC File Server on page 7](#).
  - In the **Server Name** field, verify that the name of your computer (local computer name) is correct.
  - In the **Port** field, leave the default setting as **9200**.
  - To finish, click **OK**.



- In the **Storage Location Type** dialog, for your file storage entry, confirm that under the **Available** heading, **Yes** appears.
- Click the **Refresh** button to update for **Available** if needed.
- To finish, select your file storage entry row and click **OK**.



In digital signal processing a system is something that operates on one or more inputs to produce one or more outputs. A digital filter is defined as a system (in the case of Clampfit Batch Analysis, a software algorithm) that operates on digitized data to either pass or reject a defined frequency range. The objective of digital filtering is to remove undesirable frequency components from a digitized signal with minimal distortion of the components of interest.

There will be instances when it is necessary to filter experimental data after they have been digitized. For example, you might want to remove random noise or line frequency interference from the signal of interest. To this end, Clampfit Batch Analysis offers several types of digital filters.

The lowpass filters include Bessel (8-pole), boxcar, Butterworth (8-pole), Chebyshev (8-pole), Gaussian, a single-pole RC and an 8-coincident-pole RC. The highpass filters include Bessel (8-pole) and 8-coincident-pole RC. The Gaussian and boxcar filters are finite impulse response (FIR) filters while the Bessel, Butterworth, Chebyshev and RC filters are infinite impulse response (IIR) filters.

A notch filter is available to reject a narrow band of frequencies and an electrical interference filter is provided to reject 50 or 60 Hz line frequencies and their harmonics.

## Finite vs. Infinite Impulse Response Filters

Digital filters can be broadly grouped into finite impulse response (FIR) filters and infinite impulse response (IIR) filters. FIR filters are also referred to as nonrecursive filters while IIR filters are referred to as recursive filters.

The output of FIR filters depends only on the present and previous inputs. The general 'recurrence formula' for an FIR filter, which is used repeatedly to find successive values of  $y$ , is given by:

$$y_n = \sum_{k=0}^M b_k x_{n-k}$$

where  $y_n$  is the output value for the  $n^{\text{th}}$  point  $x$  and  $b_k$  is the  $k^{\text{th}}$  of  $M$  filter coefficients. In the case of the Gaussian and boxcar filters in Clampfit Batch Analysis, the  $M$  points ahead of the current point are also used, giving a general recurrence formula of:

$$y_n = \sum_{k=-M}^M b_k x_{n-k}$$

where the filter width is  $2 \times M + 1$  points.

The disadvantage of FIR filters is that they can be computationally inefficient as they might require several tens, hundreds or even thousands of coefficients depending on the filter characteristics.

The advantages are that FIR filters are inherently stable because there is no feedback and they possess ideal linear phase characteristics, exhibiting no phase distortion. That is, all frequency components passing through the filter are subject to the same pure time delay.

On the other hand, the output of IIR filters depends on one or more of the previous output values as well as on the input values. That is, unlike FIR filters, IIR filters involve feedback. The general recurrence formula for an IIR filter is given by:

$$y_n = \sum_{j=1}^N a_j y_{n-j} + \sum_{k=0}^M b_k x_{n-k}$$

where  $a$  and  $b$  are the  $N$  and  $M$  filter coefficients, where  $a$  represents the feedback coefficients. The value of  $y$  for a given point  $n$  depends on the values of previous outputs  $y_{n-1}$  to  $y_{n-N}$  as well as the input values  $x$ .

The major advantage of IIR filters is that they are computationally more efficient, and therefore much faster, than FIR filters. The disadvantages are that IIR filters can become unstable if the feedback coefficients are unsuitable, and recursive filters cannot achieve the linear phase response that is characteristic of FIR filters. Therefore, all IIR filters introduce a phase delay to the filtered data.

The problem of potential instability of IIR filters is solved in Clampfit Batch Analysis by limiting the cutoff frequencies for all filter types to a range where the response is always be stable (see [Cutoff Frequency Limitations on page 74](#)). However, the phase delay is not corrected.

The Nyquist rate has important consequences for digital filtering in that the maximum analog frequency that a digital system can represent is given by:

$$f_h = \frac{1}{2T}$$

where  $T$  is the minimum sampling interval and  $f_h$  is the Nyquist frequency.

As a consequence of this, the maximum filter cutoff frequency of digital filters is limited to one-half the sampling rate. That is, the ratio of the cutoff frequency to the sampling rate ( $f_c/f_s$ ) cannot exceed 0.5. In fact, only the Gaussian and single pole RC filters can realize an  $f_c/f_s$  ratio as high as 0.5; the Bessel, Butterworth, Chebyshev and notch IIR filters are limited to values that are somewhat lower than this because of performance degradation at higher  $f_c/f_s$  ratios (see [Cutoff Frequency Limitations on page 74](#)).

The  $f_c/f_s$  ratio limitation should not present a problem if antialias filtering and oversampling are judiciously applied. For example, with a lowpass antialiasing filter cutoff of 4 kHz and a sampling rate of 40 kHz, an  $f_c/f_s$  ratio limited to as low as 0.1 allows a maximum cutoff frequency of 4 kHz, which is well above any useful cutoff frequency that might be applied to this particular digitized record.

## Digital Filter Characteristics

An ideal filter would have a rectangular magnitude response with no attenuation in the passband and full attenuation in the stopband. However, ideal filters are noncausal in that the present output depends on future values of the input. They are, therefore, not realizable. However, realizable digital filters can approximate ideal filters in that the output can be delayed for a finite interval until all of the required inputs have entered the system and become available for determination of the output.

Different filter types optimize different characteristics of the ideal filter that they are meant to approximate. Therefore, the application of a particular filter type should be carefully considered in view of the specific requirements at hand.

Most filters introduce a time lag between the input and output signals. Depending on the filter type, some frequencies are subjected to a greater lag than others. As a consequence the output signal is distorted to some degree. This distortion takes the form of 'ringing' and 'overshoot' in the filter output given a step function input (for example, a square pulse). Filters that introduce equal time lags for all frequencies are said to have a constant 'group delay'. Such filters exhibit minimal ringing and overshoot. A filter can be characterized by its cutoff frequency and steepness of response. In the Clampfit Batch Analysis filters, the cutoff frequency is defined as the frequency at which the signal amplitude decreases by a factor of 2. This corresponds to a drop in power of  $1/\sqrt{2}$ , or  $-3$  decibels (dB). The steepness of a filter, its 'roll off,' defines the rate at which a signal is attenuated beyond the cutoff frequency. It is desirable to have as steep a roll off as possible so that unwanted frequencies are maximally attenuated. However, filters that are designed for maximally steep rolloffs necessarily sacrifice constant group delay characteristics, and therefore exhibit ringing and overshoot. The steepness of a filter response is also a function of its order (number of poles): the higher the filter order, the steeper the response. Apart from the single pole RC filter, the IIR filters in Clampfit Batch Analysis are 8-pole realizations.

## End Effects

All software filters exhibit 'end effects' at the beginning or end of a data record. In the case of the boxcar and Gaussian filters, end effects occur at both ends of the record because these filters use both previous and succeeding points to filter the current point. Clearly, at the beginning of the record only succeeding points are available. These filters are, therefore, phased in progressively as previous points become available. Towards the end of the record fewer and fewer following points become available so the filter is progressively phased out. The filter coefficients are adjusted during these phases in accordance with the available number of points.

Filters with fewer coefficients exhibit shorter end effects as the full operating width overlaps a fewer number of points. The number of settling points required for the Gaussian and boxcar filters is  $(\text{filter length} - 1) / 2$ . In the case of the Gaussian the filter length is equal to the number of coefficients while in the case of the boxcar the filter length is equal to the number of averaging points.

IIR filters exhibit startup-transients only. Since these filters do not use points ahead of the current point for the filter output there is no phaseout transient. The output of these filters depends on previous outputs as well as the current input. Therefore, a certain number of points must be processed before these filters reach stability.

The rise time ( $T_r$ ) of a lowpass filter can be estimated from  $T_r = 0.35/f_c$ . As the filter settles to within about 10% of its final value in one rise time, a duration of  $3 \times T_r$  is sufficient to allow the filter to reach 0.1% of its final value, requiring about  $(3 \times 0.35 / f_c) \times f_s$  points.

## Bessel Lowpass Filter (8 pole) Specifications

10% to 90% step rise time:  $0.3396/f_c$

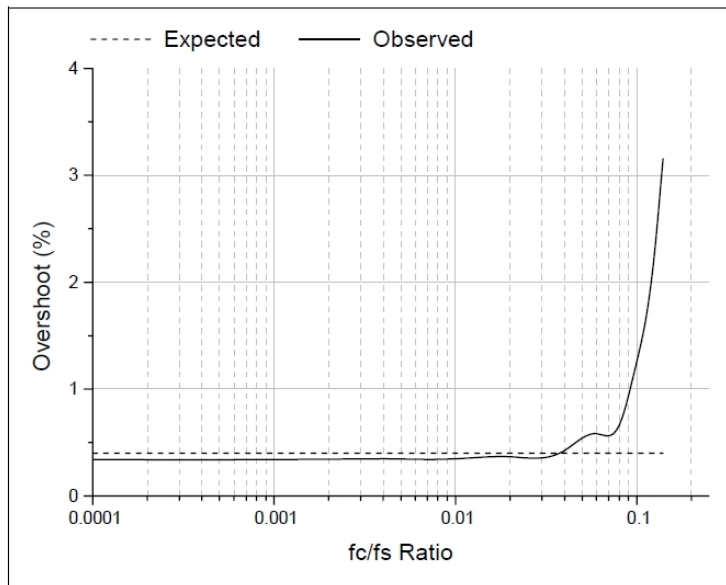
Maximum overshoot: 0.4%

Attenuation: 114 dB at  $f = 10 f_c$

A Bessel lowpass filter has a maximally flat response over the entire frequency range (constant group delay characteristics), exhibiting minimal overshoot and ringing in response to a step function. As all frequencies are delayed equally the shape of the original signal is preserved. Because of these characteristics, Bessel filters are most commonly used for time-domain analysis of biological data, where the preservation of the shape of the original signal is critical.

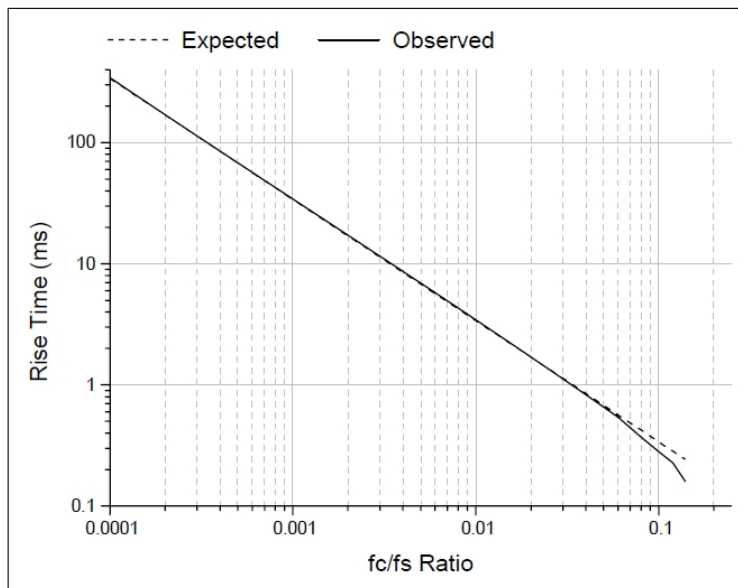
### Expected vs. Observed Overshoot

100 mV step pulse,  $f_s = 10$  kHz



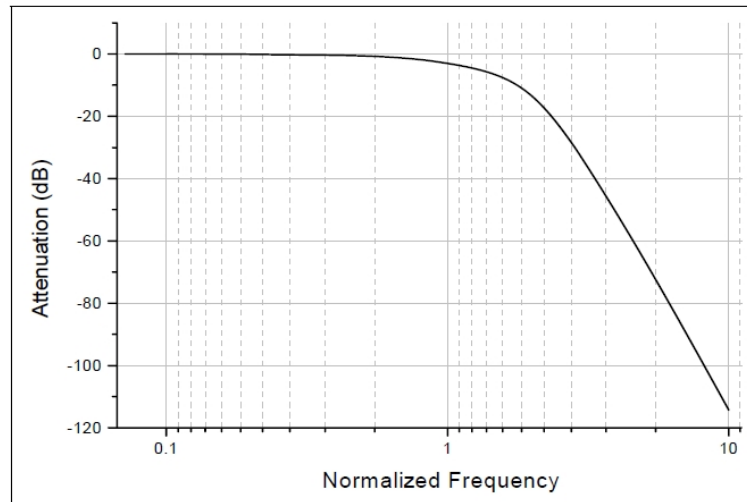
### Expected vs. Observed Rise Times

100 mV step pulse,  $f_s = 10$  kHz





## Normalized Frequency Response



## Boxcar Smoothing Filter Specifications

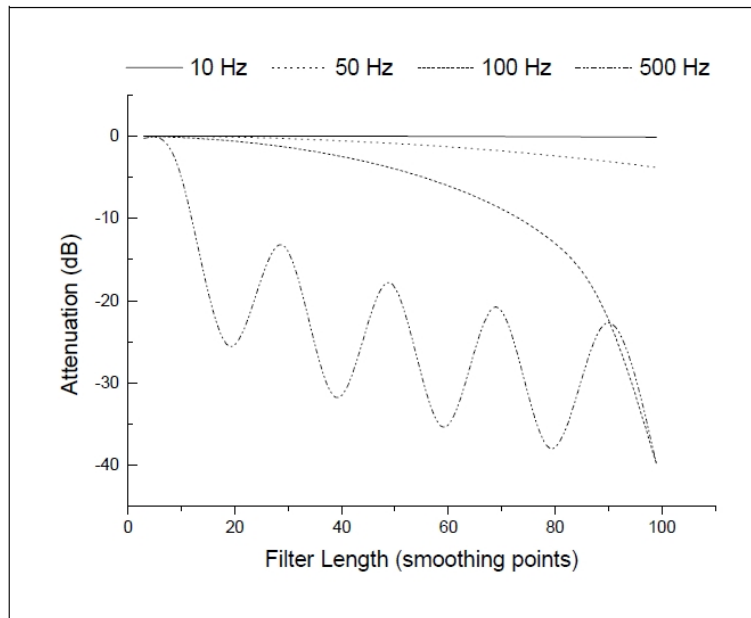
Smoothing filters are generally used to remove high-frequency components from slowly varying signals and are therefore lowpass filters. The boxcar-smoothing filter uses the average of the current point and a given number of previous and succeeding points to replace the value of the current point. The recurrence formula for a boxcar filter is:

$$y_n = \sum_{k=-M}^M \frac{x_{n-k}}{P}$$

where  $x_n$  is the  $n^{\text{th}}$  point to be filtered (at  $k = 0$ ),  $P$  is the number of smoothing points (the filter width) and  $M = (P - 1) / 2$ . The boxcar filter does not introduce a time lag.

Like the other filters, the boxcar filter also attenuates the signal; the degree of attenuation is directly proportional to the frequency of the signal and the number of smoothing points. The following figure compares the attenuation of 10, 50 and 100 and 500 Hz sine waves (sampled at 10 kHz) at various filter lengths:

## Boxcar Filter Attenuation vs. Number of Smoothing Points



That filtering periodic signals with the boxcar filter can introduce a periodic attenuation response, as seen with the 500 Hz signal. This occurs because the filter output for the current point is the mean of its value and the values of its immediate neighbors. The output therefore depends on the relative proportion of high and low data values within a given filter length.

## Butterworth Lowpass Filter (8 pole) Specifications

10% to 90% step rise time:  $0.46/f_c$

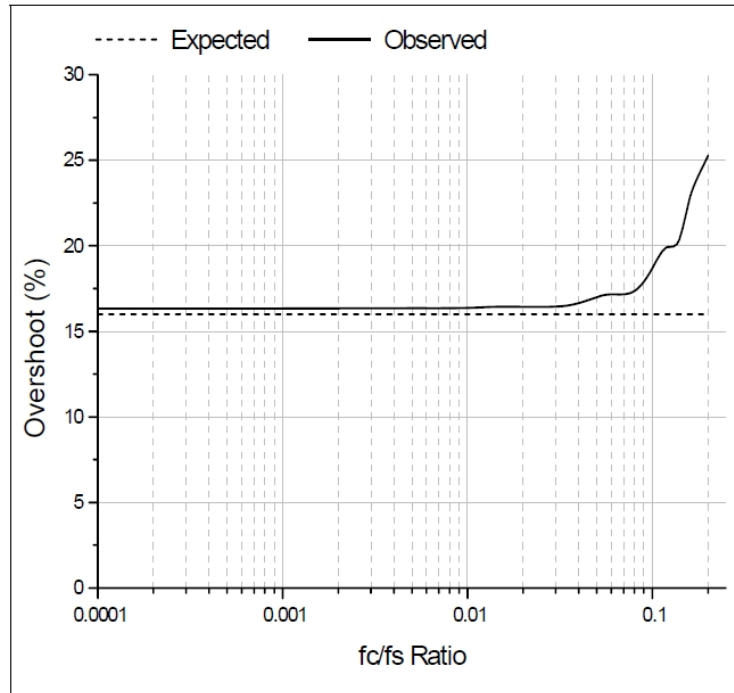
Maximum overshoot: 16.0 %

Attenuation: 160 dB at  $f = 10 f_c$

The Butterworth lowpass filter has a maximally flat response at low frequencies and a monotonically decreasing amplitude response with increasing frequency. The group delay is not constant so the Butterworth filter exhibits ringing and a substantial overshoot in response to a step function. This filter, however, has sharper roll-off characteristics than the Bessel filter. It is, therefore, better suited than the Bessel for frequency domain applications such as noise analysis. However, because of its nonconstant group delay characteristics this filter should generally not be used for time-domain analysis of biological data.

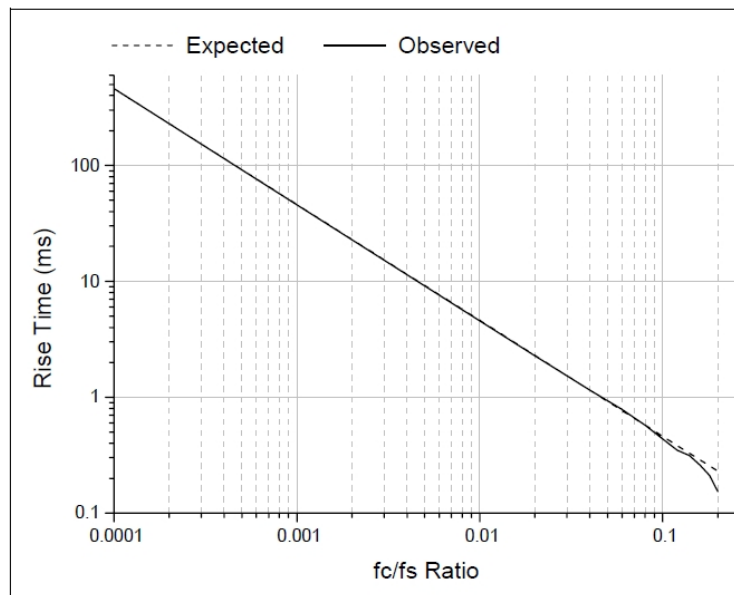
### Expected vs. Observed Overshoot

100 mV step pulse,  $f_s = 10$  kHz

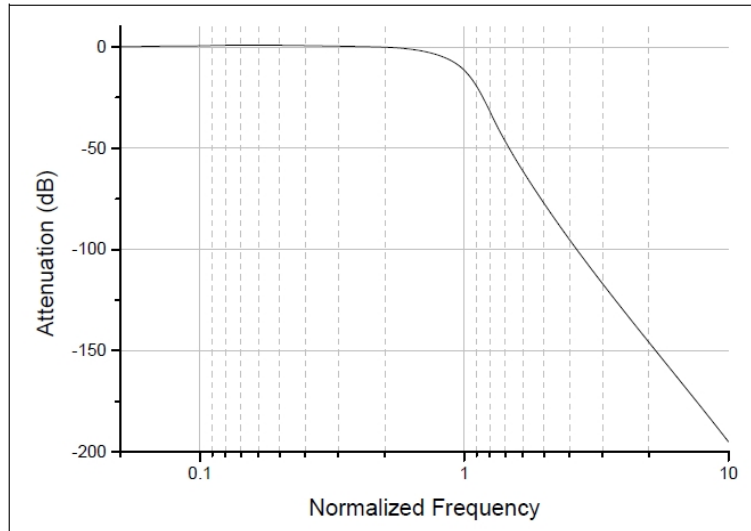


### Expected vs. Observed Rise Times

100 mV step pulse,  $f_s = 10$  kHz



## Normalized Frequency Response



## Chebyshev Lowpass Filter (8 pole) Specifications

10% to 90% step rise time:  $0.53/f_c$

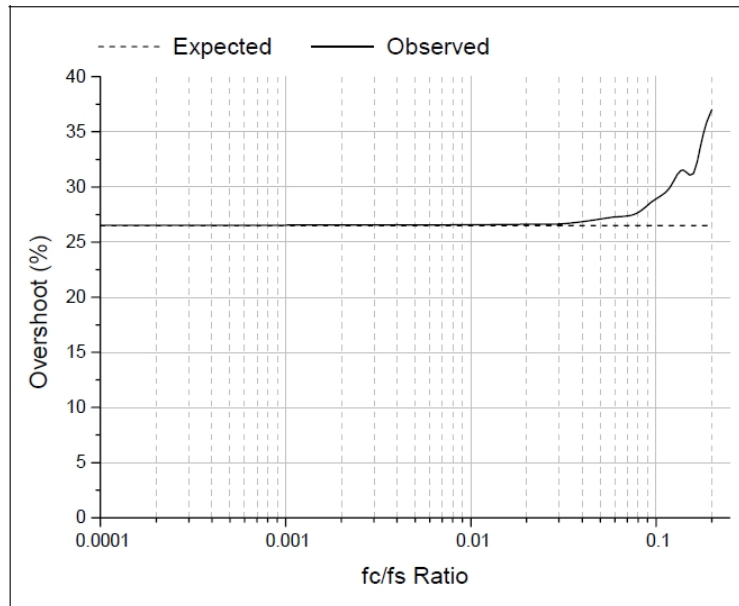
Maximum overshoot: 16.0%

Attenuation: 193 dB at  $f = 10 f_c$

The Chebyshev lowpass filter has a maximally sharp transition from the passband to the stopband. This sharp transition is accomplished at the expense of ripples that are introduced into the response. The Chebyshev filter in Clampfit Batch Analysis has a fixed ripple of 1 dB. Like the Butterworth, the sharp roll-off characteristics of the Chebyshev filter make it suitable for analysis of data in the frequency domain, such as noise analysis. Although the Chebyshev filter has a sharper roll-off than the Butterworth, it exhibits an even larger overshoot and more ringing. Therefore, it is also not generally suitable for time-domain analysis of biological data.

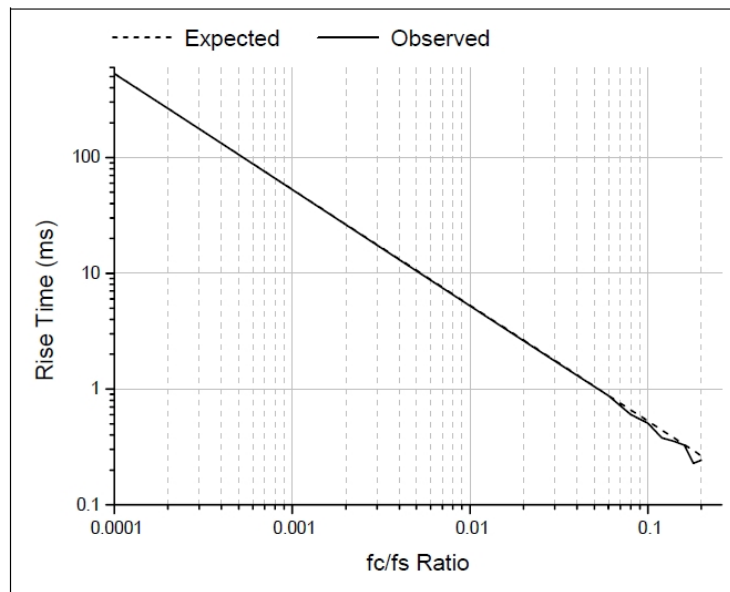
### Expected vs. Observed Overshoot

100 mV step pulse,  $f_s = 10$  kHz

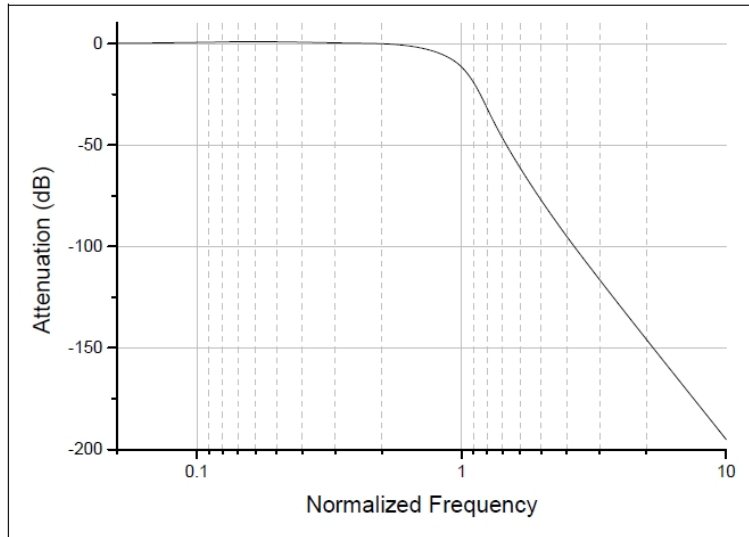


### Expected vs. Observed Rise Times

100 mV step pulse,  $f_s = 10$  kHz



## Normalized Frequency Response



## Gaussian Lowpass Filter Specifications

10% to 90% step rise time:  $0.3396/f_c$

Maximum overshoot: 0%

Attenuation: 90 dB at  $f = 10 f_c$

The Gaussian FIR filter forms a weighted sum of the input values to form an output value according to the following recurrence formula:

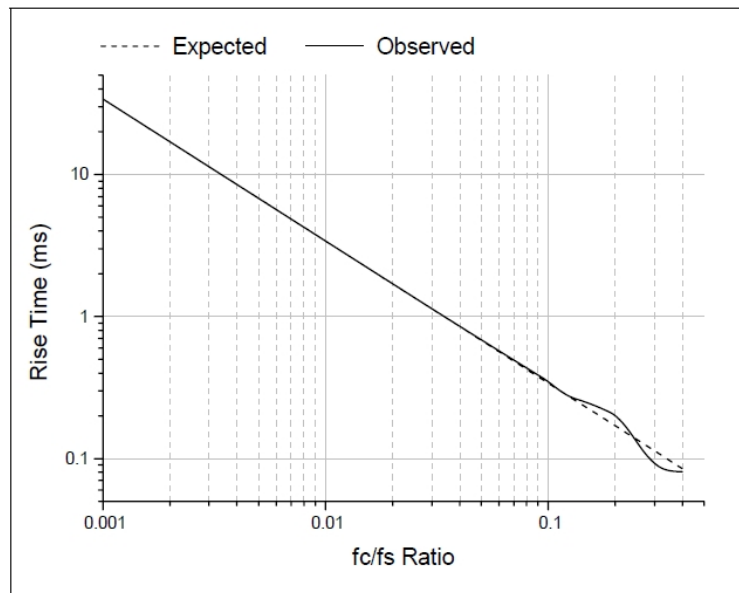
$$y_i = \sum_{j=-n}^n a_j x_{i-j}$$

where  $a_j$  are the Gaussian coefficients that sum to unity.

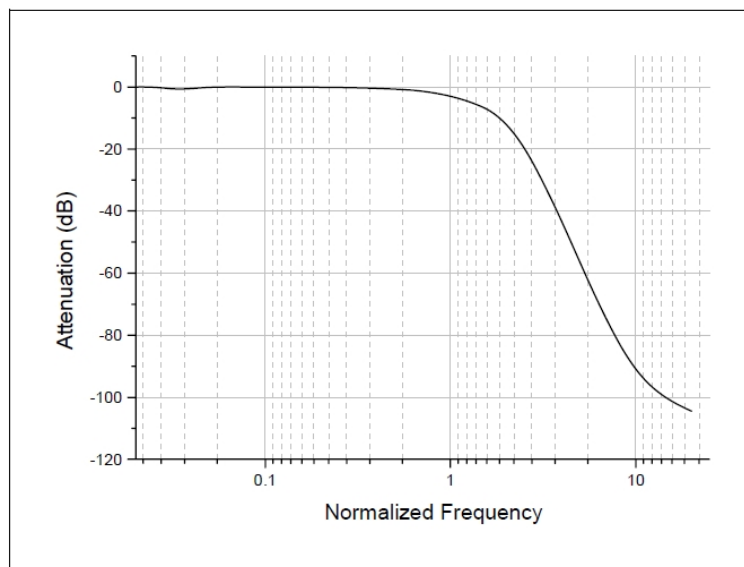
The Gaussian filter is particularly suited for filtering biological data for analysis in the time domain as it produces no overshoot or ringing and introduces no phase delay. The disadvantage is that it can be slow at high  $f_c/f_s$  ratios where the number of filter coefficients is large.

### Expected vs. Observed Rise Times

100 mV step pulse,  $f_s = 10$  kHz



### Normalized Frequency Response

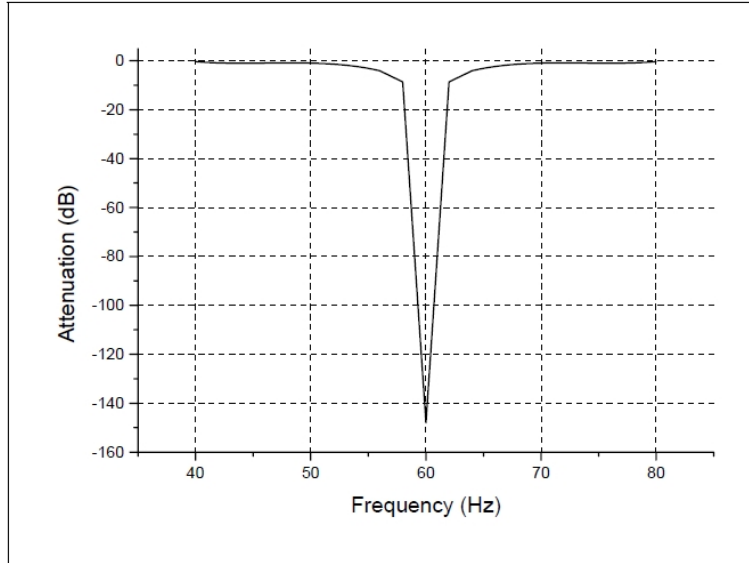


### Notch Filter (2 pole) Specifications

The number of poles of the notch filter is fixed at two. This filter has essentially zero gain ( $-\infty$  dB) at its center frequency and about unity gain (0 dB) elsewhere. The notch filter has approximately zero phase shift except at its center frequency, at which the phase shift is undefined (because the gain is zero). In both respects (magnitude and phase) the resonator behaves like a 'real' analog tuned circuit.

The following figure shows the frequency response of the notch filter with a 60 Hz center frequency with a 10 Hz –3 dB width.

### Notch Filter Frequency Response (60 Hz center frequency, 10 Hz –3 dB width)



### Settling Points

The –3 dB width has a significant influence on the number of points required for the notch filter to settle, the narrower the notch the greater the number of settling points. For example, for a 60 Hz sine wave sampled at a frequency of 1 kHz, applying a 60 Hz notch filter with a 1 Hz –3 dB width requires 2000 sampling points for the filter to reduce the amplitude to 0.1% of its original value (–60 dB). In contrast, a 60 Hz notch filter with a 10 Hz –3 dB width requires only 200 points.

The number of settling points also increases with increasing sampling frequency. For example, a notch filter with a –3 dB width of 10 Hz requires 2000 settling points for data sampled at 10 kHz compared to 200 for data sampled at 1 kHz.

The relationship between the number of settling points ( $P_s$ ) and the sampling frequency ( $f_s$ ) and –3 dB notch width ( $W_{-3dB}$ ) is given by:

$$P_s = \frac{2f_s}{W_{-3dB}}$$

for attenuation of the center frequency by 60 dB.

### RC Lowpass Filter (single pole) Specifications

10% to 90% step rise time:  $0.3501/f_c$

Maximum overshoot: 0%

Attenuation: 20 dB at  $f = 10 f_c$

The RC filter function is equivalent to that of a simple first-order electrical filter made up of a single resistor and a single capacitor. The RC filter introduces a phase delay to the output, but the group delay is constant so that there is no ringing or overshoot. The recurrence formula for this filter is:



$$Y(n) = X(n) + W + (Y(n-1) - X(n-1))$$

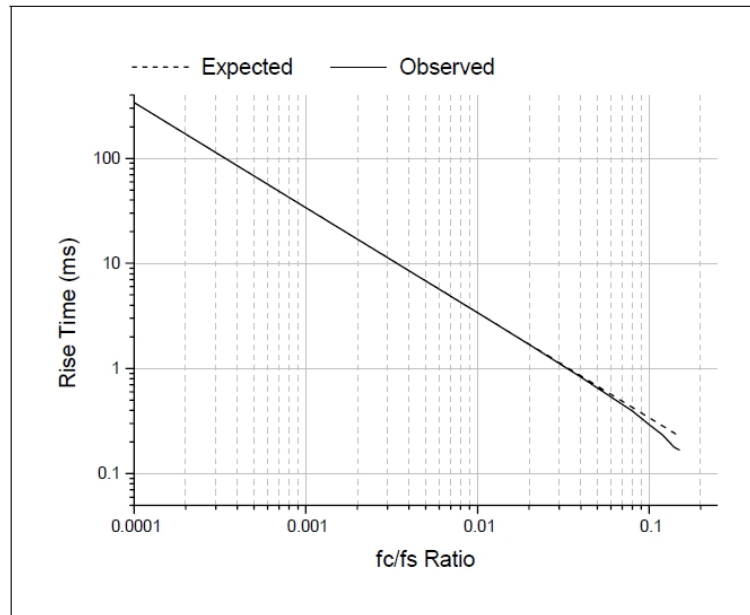
where  $Y(n)$  is the current output,  $X(n)$  is the current data point,  $Y(n-1)$  is the output for the previous point and:

$$W = e^{-dt/\tau} \text{ where } \tau = 1/2\pi f$$

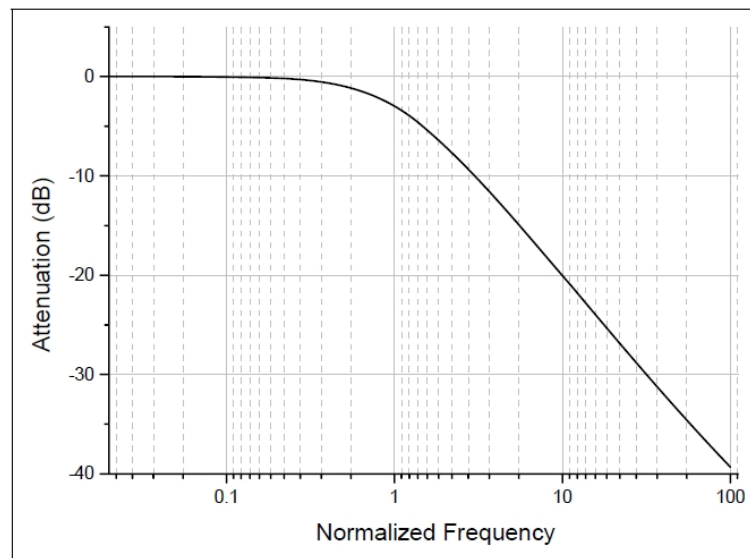
where  $dt$  is the sampling interval and  $f$  is the  $-3$  dB cutoff frequency.

### Expected vs. Observed Rise Times

100 mV step pulse,  $f_s = 10$  kHz



### Normalized Frequency Response



## RC Lowpass Filter (8 pole) Specifications

10% to 90% step rise time:  $0.34/f_c$

Maximum overshoot: 0%

Attenuation: 80 dB at  $f = 10 f_c$

The 8-pole RC filter is a 'multiple coincident pole' realization where the data points are filtered by applying 8 single pole RC sections in succession. The recurrence formula for this filter is, therefore, identical to that of the single pole RC filter except that the output from each previous pole is used as the input for the successive pole, where:

$$Y(n) = X(n_p) + W + (Y(n_p - 1) - X(n_p - 1))$$

for  $p = 1$  to 8, where  $Y(n)$  is the output from the current pole,  $X(n_p)$  is the filter output of the previous pole for the current point,  $Y(n_p-1)$  is the output from the previous pole for the previous point and:

$$W = e^{-dt/\tau} \text{ where } \tau = 1/2\pi(f/f_N)$$

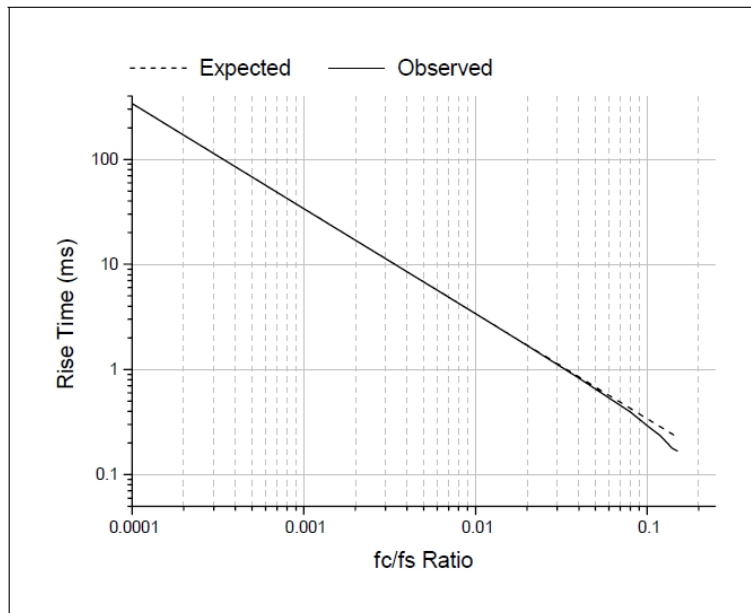
where  $f_N$  is the normalized cutoff frequency. With the coincident pole design the cutoff frequency rises as the order of the filter is increased. The normalized cutoff frequency,  $f_N$ , is given by:

$$f_N = 1/\sqrt{2^{1/n} - 1}$$

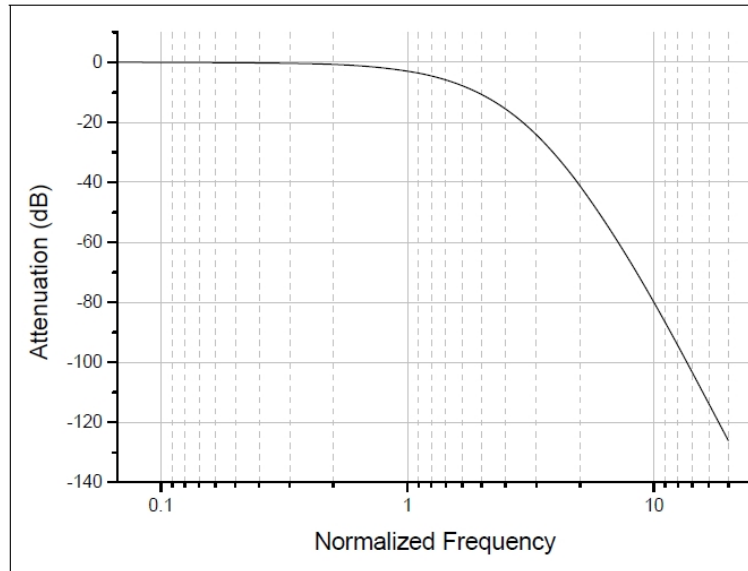
where  $n$  is the order (number of poles) of the filter. For an 8-pole filter this value is 3.32397. The specified cutoff frequency must be divided by the normalization factor in order to adjust the positions of the multiple poles. A consequence of this is that the maximum  $f_c/f_s$  ratio must be limited to the Nyquist frequency ( $f_c/f_s = 0.5$ ) divided by the normalized cutoff frequency, or  $0.5 / 3.32397 = 0.15$ .

### Expected vs. Observed Rise Times

100 mV step pulse,  $f_s = 10$  kHz



## Normalized Frequency Response



## RC Highpass Filter (single pole) Specifications

Attenuation: 20 dB at  $f = 0.1 f_c$

The single-pole RC highpass filter operates by subtracting the lowpass response from the data. This is valid in this case because of the constant group delay characteristics of the single pole RC filter. The recurrence formula for this filter is, therefore:

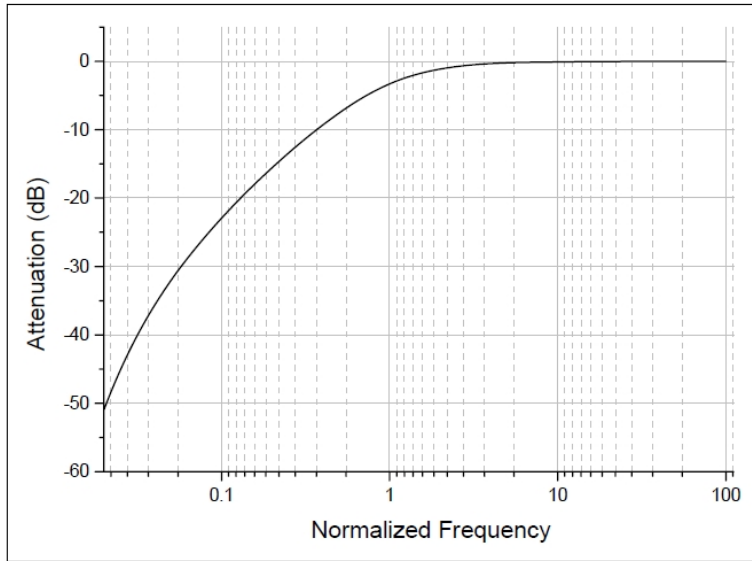
$$Y(n) = X(n) - [X(n) + W + (Y(n-1) - X(n-1))]$$

where  $Y(n)$  is the current output,  $X(n)$  is the current data point,  $Y(n-1)$  is the output for the previous point and:

$$W = e^{-dt/\tau} \text{ where } \tau = 1/2\pi f$$

where  $dt$  is the sampling interval and  $f$  is the  $-3$  dB cutoff frequency.

### Normalized Frequency Response



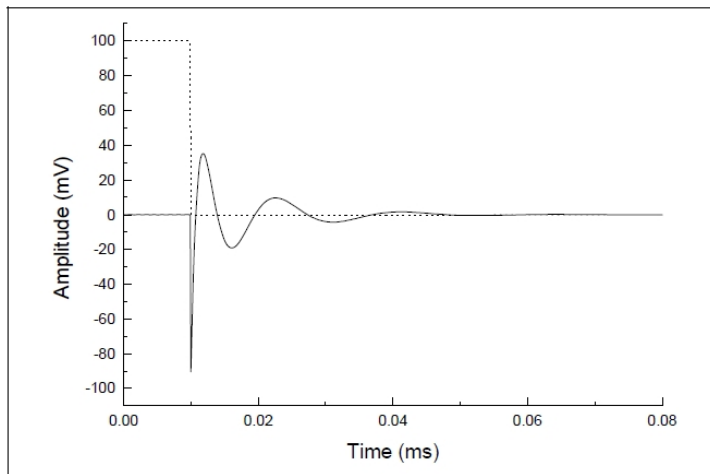
### Bessel Highpass Filter (8-pole analog) Specifications

Attenuation: 114 dB at  $f = 0.1 f_c$

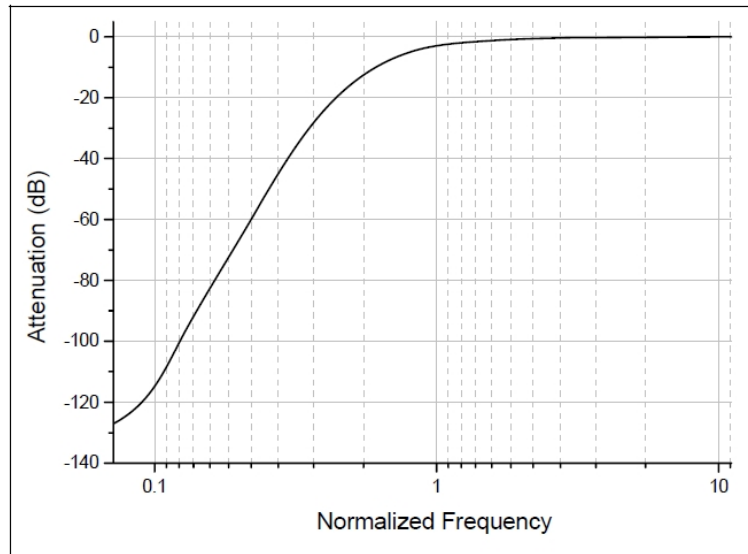
The highpass Bessel filter has a sharper roll-off than the highpass RC filter. However, the Bessel filter deviates from ideal behavior in that it introduces ringing in the response to a step function, as shown below.

### Highpass Bessel Filter Step Response

100 mV step pulse (dotted line),  $f_s = 10$  kHz,  $f_c = 100$  Hz



## Normalized Frequency Response



## Electrical Interference Filter

The electrical interference (EI) Filter removes power line interference from an acquired data signal. This filter identifies and removes complex power line waveforms composed of multiple harmonics. The interference detection method is adaptive; that is, the filter always matches the actual line signal even if its properties (frequency, phase and shape) change during measurement.

The core component of the EI filter is the sine detector that discriminates sinusoids from other additive signal components. The sine detector generates a reference sine wave and adjusts its phase until it locks to a specific line-interference harmonic. The EI filter uses an array of sine detectors, one for each harmonic of the interference waveform.

The sine detector basically operates as a digital Phase Locked Loop (PLL) tuned to line frequency (50/60 Hz) or its multiple. The correlator (phase detector) detects phase difference between the reference and actual line harmonic. This phase difference is used as a feedback signal to adjust the reference phase until a perfect match is achieved.

Reference signals, each locked to a specific interference harmonic, are subtracted from the original signal, thus canceling out the complete interference waveform.

### Assumptions

1. The line interference and the data signal are statistically independent, for example, uncorrelated.
2. The line signal  $x(n)$  is stationary across at least  $M$  of its periods. We can assume that the line signal does not significantly change when observed at any  $M$  of its periods.
3. The measured signal  $s(n)$  is the sum of data signal  $y(n)$  with scaled and phase shifted (delayed) version of the line signal:  $s(n) = y(n) + Ax(n-d)$ .

### Problem Statement

We want to identify line signal  $x(n)$  incorporated inside  $s(n)$ . We assume that line signal is composed of a certain number of sinusoids with harmonic frequencies. According to assumption 3, we need to determine  $A$  and  $d$  for each harmonic.

### Basic Theory

In order to detect the line signal we use the fact that data signal  $y(n)$  and line signal  $x(n)$  are uncorrelated. Practically, we say that cross correlation  $R_{xy}$  is equal to zero:

$$R_{xy}(d) = \frac{1}{M\Delta} \sum_{i=0}^{MA} y(i)x(i-d) = 0$$

for each  $d$ , where  $\Delta$  is the number of samples per period of the line signal. In order to keep the argument as simple as possible, we will use the term ‘correlation’  $R_{xy}$  meaning in most places actually ‘covariance’, implicitly assuming all signals to have zero DC component. The correlation of the line signal  $x(n)$  and the measured signal  $s(n)$  should equal to:

$$R_{xs}(d) = R_{xy}(d) + R_{xx}(d)$$

where  $R_{xx}$  is the autocorrelation of the line signal. Since  $R_{xy}$  is equal to zero, we conclude that if we correlate the line signal and measured signal, we will obtain the autocorrelation of the line signal. The autocorrelation function  $R_{xx}(d)$  is even and its maximum is at  $d = 0$ . Furthermore, since  $x(n)$  is periodic,  $R_{xx}$  is also periodic with the same period.

We use the above argument to determine both the delay  $d$  and the scale  $A$  of the line signal inside the measured signal (see assumption 3). The above discussion holds for our specific situation if we assume that  $x(n)$  is a sinusoidal reference signal.

### Electrical Interference Filter Block Diagram

The adaptive line interference filter block diagram for the fundamental harmonic is shown in Figure 5-1. The same procedure is repeated for each harmonic by multiplying the frequency of the reference generator.

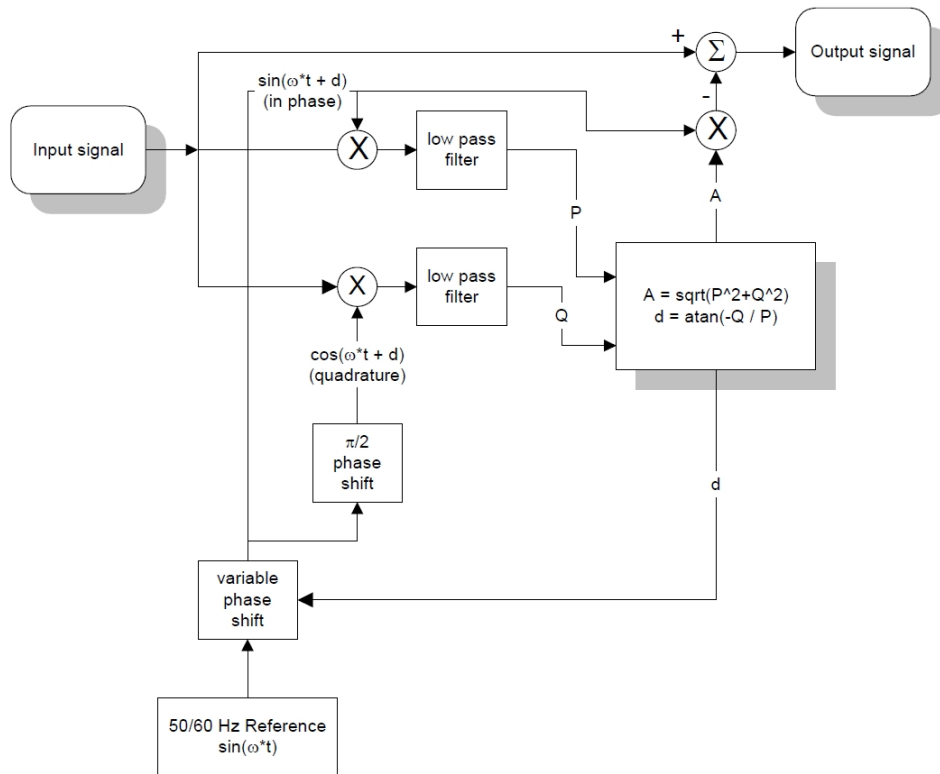


Figure 5-1: Adaptive line interference filter block diagram

## Implementation

The low pass filter is implemented as the simple average over  $M$  line periods. The average of the product of two signals, as in correlation definition, is equal to applying a low pass filter with box impulse response, or  $\sin(x)/x$  transfer function.

The critical parameter for correlator performance is the averaging period  $M$ . The higher the value of  $M$ , the lower the bandwidth of the equivalent low pass filter. By keeping the bandwidth as low as possible, we reduce the correlation estimate error. Since we ultimately need to estimate the autocorrelation of the periodic signal  $x(n)$ , the averaging period should be equal to a whole number of line signal periods  $M$ .

The EI filter generates a reference sine wave for each detected harmonic up to the maximum specified harmonic search number and subtracts reference waveforms from the signal, thereby canceling line interference. Ideally, each reference sinusoid exactly matches the corresponding interference harmonic both in phase and amplitude, resulting in the perfect cancellation of line interference.

A practical EI filter never exactly detects phase and amplitude of interference harmonics. After subtraction of the generated reference sinusoids any discrepancy in amplitude and phase will result in artifact sinusoids, that is, components that were not present in the original signal. After filtering, artifactual components actually replace the original line harmonics. When harmonic detection is good, the total power of artifact components after filtering is much lower than line interference power in the original signal.

Harmonic detection errors come from noise and data signal components at line harmonic frequencies (multiples of 50/60 Hz). Generally, noise errors are less significant and can be successfully reduced by increasing the number of cycles to average. A more significant source of EI-filter errors are original data signal components that fall at or close to 50/60 Hz multiples (data signal leak).

## Weak Harmonics

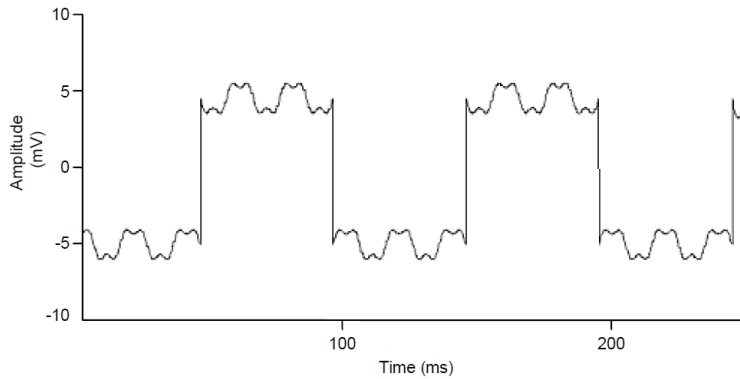
Weak line harmonics in the presence of noise cannot be accurately detected. In extreme cases EI-filtering artifact power for the single weak harmonic can be larger than the actual harmonic power. In such cases it might be better to reduce the harmonic search number in order to exclude weak harmonics. Also, increasing the number of cycles to average will always improve harmonic detection (if the noise is the main error source). However, excessively large number of cycles to average will negatively affect execution speed, tracking performance and startup transient compensation.

## Data Signal Components

### Periodic

Any periodic components in the data signal with frequencies at or very close to 50/60 Hz multiples will leak through the EI filter harmonic detectors and will produce false line interference harmonics.

Figure 5-2 shows the result of filtering a pure (no noise and no line interference) square wave at 10 Hz with the harmonic search number set to 3 and reference frequency set to auto.



**Figure 5-2: Results of filtering a square wave**

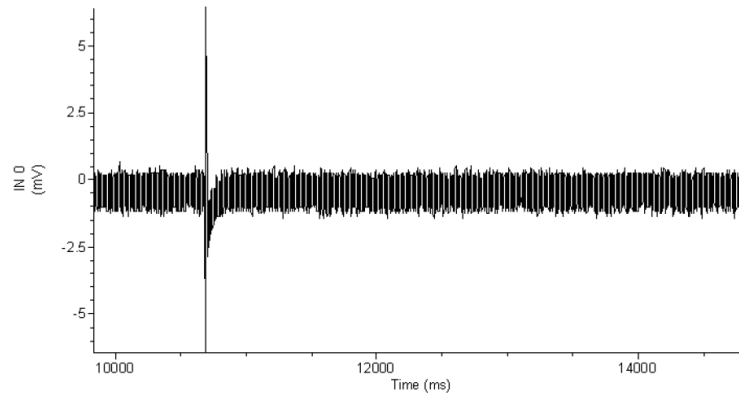
Since a 10 Hz square wave has strong components at multiples of 10 Hz, the EI filter locked on to the 5<sup>th</sup> (50 Hz), 10<sup>th</sup> (100 Hz) and 15<sup>th</sup> (150 Hz) harmonics, generating prominent artifacts.

### Aperiodic

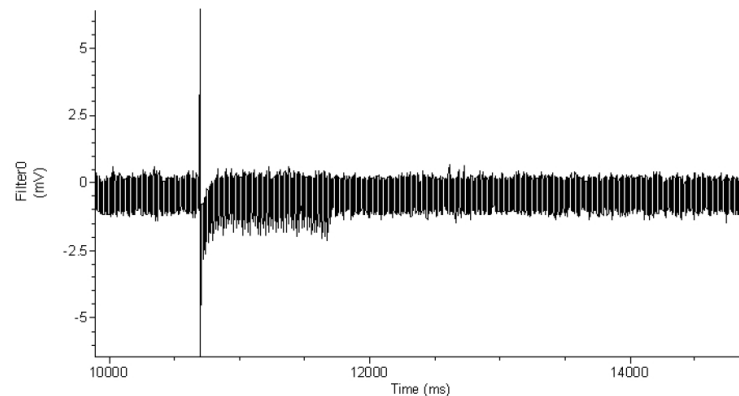
Strong and sharp pulses in the data signal may produce artifacts in the EI filtering process. Sharp pulses (spikes) have significant components at all frequencies including 50/60 Hz multiples. If the spike amplitude is two (or more) orders of magnitude larger than actual line interference amplitude, the EI filter will produce false line harmonics after the spike in the region whose size is equal to the number of cycles to average.

In the example below the EI filter was used at 10 harmonics with 50 cycles to average. Spike amplitude (not shown in full) was more than 200 mV. Notice the false line harmonics in [Figure 5-3](#).





(6a)



(6b)

Figure 5-3: EI filtering showing introduction of false line harmonics

### Start-up Transients

Start-up transients are spurious, rapidly changing false harmonics at the very beginning of the filtered signal. When processing samples at the beginning of the signal file EI filter does not have enough information to accurately detect line harmonics. With every new processed sample the detection becomes better and better and the start-up transient becomes smaller and smaller. The filter reaches its steady state after a time equal to number of cycles to average.

The EI filter automatically compensates for start-up transients by turning off reference subtraction until it reaches its steady state. When it reaches steady state after the specified number of cycles to average the EI filter assumes that line interference parameters are accurately detected and filters the signal backwards using current reference parameters.

### Potential Problems

#### The Filter is Too Slow

When dealing with large datasets and high sampling rates the filter might be slow. If filtering is unacceptably slow try the following:

- Check if it is necessary to remove all harmonics specified in the harmonics field. Try removing only the first harmonic, and if the result is not satisfactory, increase the harmonic number and try again. The removal of the first three harmonics will often sufficiently reduce the interference.

- Decrease the value in the Cycles to average field. Often, smaller averaging lengths (time constants) do not significantly affect the output signal quality.

### Interference is not Fully Removed

If the line interference is not fully removed try the following:

- If residual interference contains high frequencies then it might be necessary to increase the value of the upper harmonic to be removed.
- If the fundamental line harmonic is still visible in the output signal then the number of cycles to average should be increased.

### Cutoff Frequency Limitations

All digital filters have inherent limitations, and in some cases deviate significantly from their analog counterparts. The filters in Clampfit Batch Analysis have been restricted to an  $f_c/f_s$  ratio where the filter response is reasonably close to the theoretically expected response.

- The theoretical frequency range of digital filters is between 0 and the Nyquist frequency, which is one-half of the sampling frequency. This applies to all filters when filtering sampled data. However, the usable range of most software filters is considerably narrower than this theoretical range. The usable range depends on the nature of the filter (FIR or IIR) and the filter algorithm.
- The overshoot during a step response is a characteristic feature of Bessel, Butterworth and Chebyshev lowpass filters. For analog filters, the magnitude of the overshoot is constant over the full operating range. For digital IIR filters, however, the overshoot becomes increasingly larger as the ratio of  $f_c/f_s$  increases.
- The operating range of the Gaussian FIR filter is limited at the low end by a practical, rather than theoretical, limitation. Low ratios  $f_c/f_s$  result in the generation of a large number of filter coefficients. This creates two problems. The first is that smaller datasets cannot be accurately filtered because the filter length might be greater than the number of data points. The second is that the large number of coefficients is computationally inefficient. The number of Gaussian coefficients is inversely proportional to the  $f_c/f_s$  ratio where a lower cutoff frequency requires a greater number of coefficients for the filter realization.

The following table lists the numerical limitations for each filter type. The lower and upper cutoff frequencies are expressed as a factor times the sampling frequency ( $f_s$ ). These limits are internally set by the filter algorithm and cannot be altered.

Filter Type	Lower Cutoff Limit	Upper Cutoff Limit
Bessel (8-pole IIR)	$10^{-4} \times f_s$	$0.14 \times f_s$
Boxcar (FIR)	See note 1.	n/a
Butterworth (8-pole IIR)	$10^{-4} \times f_s$	$0.2 \times f_s$
Chebyshev (8-pole IIR)	$10^{-4} \times f_s$	$0.2 \times f_s$
Electrical Interference	See note 2.	n/a
Gaussian (FIR)	$10^{-4} \times f_s$ See note 3.	$0.5 \times f_s$
Notch (2-pole IIR)	$10^{-3} \times f_s$	$0.3 \times f_s$
RC (single-pole IIR)	$10^{-4} \times f_s$	$0.5 \times f_s$
RC (8-pole IIR)	$10^{-4} \times f_s$	$0.15 \times f_s$ . See note 4.

**Note: 1**

The boxcar filter requires that the number of smoothing points be specified. This must be an odd number in order for the filter to be symmetrical. The minimum number of smoothing points is 3. The maximum number of smoothing points is 99. However, the maximum number of smoothing points is also limited by the number of data points,  $n$ , such that the filter width is at least  $n/2$ . So if there are 50 data points the maximum number of smoothing points is  $50 / 2 = 25$ . If this formula generates an even number then the maximum number of smoothing points will be one less. For example, if there are 52 data points, then the maximum number of smoothing points will be  $52 / 2 - 1 = 25$ .

**Note: 2**

The electrical interference filter does not have a lower or upper cutoff frequency limit as it is preset to remove either 50 Hz or 60 Hz interference and the associated harmonics (see [Electrical Interference Filter on page 69](#)). However, there is a data point minimum, as this filter requires a specific number of points to reach steady state. This minimum is given by:

$$\text{minimum points} = \text{samples per period} \times \text{cycles to average}$$

where the samples per period is the sampling frequency divided by the reference frequency (50 Hz or 60 Hz) and cycles to average is the number of cycles of the reference frequency which are averaged in the response. For example, for a sampling rate of 1 kHz, a reference frequency of 60 Hz and 20 cycles to average the minimum number of data points required is  $1000 / 60 \times 20 = 334$  data points.

**Note: 3**

The Gaussian filter width (see [Finite vs. Infinite Impulse Response Filters on page 53](#)) depends on the  $f_c/f_s$  ratio; the lower this ratio the greater the number of Gaussian coefficients (see [Gaussian Lowpass Filter Specifications on page 62](#)). In view of this, two criteria are used to limit the lower cutoff frequency.

The first is that there must be enough data points to accommodate at least two Gaussian filter widths. That is, the minimum corner frequency will correspond to a filter width that is less than or equal to one-half the number of available data points.

The second is that the maximum number of Gaussian coefficients is limited to approximately 3500. This limit, which corresponds to an  $f_c/f_s$  ratio of about  $3 \times 10^{-4}$ , is not exact because the automatically computed minimum corner ratio is generally rounded up. Therefore, the minimum corner ratio might correspond to a number of coefficients that is somewhat more or less than the 3500 limit.

**Note: 4**

The 8-pole RC filter is a 'multiple coincident pole' design where the -3 dB cutoff frequency rises with each pole by an amount given by:

$$f_N = 1/\sqrt{2^{1/n} - 1}$$

where  $f_N$  is the normalized cutoff frequency and  $n$  is the number of poles. Therefore, for an 8-coincident-pole filter the normalized cutoff frequency is actually 3.32397 times the specified cutoff frequency (see [RC Lowpass Filter \(8 pole\) Specifications on page 66](#)). Consequently, the maximum  $f_c/f_s$  ratio must be limited to the Nyquist frequency ( $f_c/f_s = 0.5$ ) divided by the normalized cutoff frequency, or  $0.5 / 3.32397 = 0.15$ .

Clampfit Batch Analysis offers a powerful and flexible tool for fitting curves to data. An extensive set of options should satisfy the most demanding requirements. Four different search methods support a large number of predefined functions as well as user-defined (custom) functions. Automatic seeding of function ‘parameters’ is provided for the predefined functions. Graphically assisted seeding is also available for more demanding fits that require accurate determination of initial seed values.

A ‘fitting method’ is composed of a search method, a minimization method and an optional weighting method. The search method is the algorithm that reduces the difference between the data and the fitted function. The minimization method is used to specify the quantity that is to be minimized (or maximized in the case of maximum likelihood). The search methods include Levenberg-Marquardt, variable metric, Simplex and Chebyshev.

The minimization methods include sum of squared errors, maximum likelihood, mean absolute and minimax.

The weighting methods include function weighting, data weighting, bin width weighting or no weighting.

Linear and polynomial regression routines are also provided. These noniterative methods are automatically used when linear regression or polynomial regression is selected from the predefined function list. However, custom-defined linear or polynomial functions can only be fitted by means of one of the iterative search methods.

### Fitting Model

A ‘fitting model’, or simply, ‘model’ is defined as any function that is fitted to the data. Functions with different numbers of terms are considered to be different models. For example, a two-term exponential function and a three-term exponential function represent different models.

### Function Parameters

The term ‘parameters’ refers to those coefficients of the fitting function that are adjusted during fitting. For example, in the following function,  $A$ ,  $\tau$  and  $C$  are the function parameters. In all predefined functions the variable  $C$  is a constant offset in the  $y$  direction:

$$f(x) = Ae^{-\tau x} + C$$

Parameters are or are not adjusted by the fitting routine depending on whether they are ‘free’ or ‘fixed’. Fixed parameters, which are essentially constants, are always assigned a standard error of zero.

### Parameter Errors

All search methods report a standard error for each parameter in the fitting function. Parameter errors are estimated by evaluation of a covariance matrix using Gauss-Jordan elimination. The method of error evaluation is identical for all fitting methods, so the results of fitting by different methods can be compared directly.

In some cases parameter errors cannot be estimated because the covariance matrix cannot be evaluated. In this unlikely event the message ‘Could not compute parameter errors.’ is given in the Results window.

The parameter errors provide an estimate of the uncertainty in the determination of the parameters themselves. They do not necessarily provide information about the goodness of the fit. The correlation coefficient and the standard deviation of the fit are more reliable indicators of the quality of the fit. In fact, if the fit is poor the parameter errors are likely to be meaningless. In other words, the parameter errors are an indication of how reliably the parameters of a given model were determined for a particular dataset, where small errors suggest that the parameter estimates are reliable regardless of the quality of the fit. Therefore, the parameter errors can be quite small although the deviation between the fitted curve and the data might be quite large.

Alternatively, for small datasets the parameter error estimates can be quite large (perhaps as large or even larger than the parameter estimates themselves), but the fit, nevertheless, can still be quite good. Clearly, statistical parameters such as estimated errors cannot be as reliable with small datasets as with larger sets.

## Fitting Failure

The fitting algorithms in Clampfit Batch Analysis are very robust and will perform satisfactorily in most cases. However, there is always the possibility that a fit will fail. The most likely reasons for a fit failure are:

- The data are very poorly described by the fitting function.
- The initial seed values were very inaccurate.
- Sign restrictions were applied and the search algorithm cannot assign acceptable positive values to the function parameters, for example, the data cannot be reasonably well described by the sign restricted fitting function.

In the event of a fit failure possible solutions are:

- Ensure that the data are indeed reasonably well represented by the fitting function. If not, select or define a different fitting function. Also, try a different number of terms or run a model comparison.
- Assign more accurate seeds to the fitting function. Graphical seeding should be very helpful for this.
- Use the variable metric search method. This search method is the most reliable for forcing function parameters positive.
- Disable the Force parameters positive option.
- Reduce the tolerance. This could result in a poorer, although still acceptable, fit.
- Reduce the maximum number of iterations. Sometimes an acceptable fit can be achieved (as judged by the parameter errors and the quality of the fitted curve) even though the fit does not converge. This is especially true for Simplex, which can continue to search for many iterations even though it is very close to the function minimum.

In the event of a failed fit the error is reported in the Results window. When fitting multiple sweeps, errors do not cause execution to stop. If an error occurs while fitting a given sweep, the error is recorded in the Results window and fitting continues with the next sweep.

Therefore, if you have fitted a series of sweeps you should check the fitting results to ensure that all sweeps have been successfully fitted.

## Numerical Limitations

- The maximum number of data points that can be fitted is 110,000.
- The maximum number of function terms (where applicable) is 6.
- The maximum power (where applicable) is 6.

- The maximum number of parameters in a custom function is 24.
- The maximum number of independent variables in a custom function is 6.
- Only one dependent variable is allowed in a custom function.
- The maximum number of points for functions that contain a factorial term is 170.

## Units

The fitting routines in Clampfit Batch Analysis do not make any assumptions about the units of the data. The variety of data sources and the potential for various data transformations makes the automatic tracking or assignment of units virtually impossible. Consequently, units are not displayed along with the reported parameter values. It is up to the user to determine the units of a particular function parameter.

## Levenberg-Marquardt Method

The Levenberg-Marquardt method supports the least squares, mean absolute and minimax minimization methods. The explanation given here is for least squares minimization but the general principle is the same for all minimization functions.

The sum of squared errors (SSE) is first evaluated from initial estimates (seed values) for the function parameters. A new set of parameters is then determined by computing a change vector  $\Delta P$  that is added to the old parameter values and the function is reevaluated. The value of  $\Delta P$  will depend on the local curvature in the 'parameter space' that can be evaluated to determine the optimal rate and direction of descent toward the function minimum. This process continues until the SSE is 'minimized' at which time the fit is said to have converged. The criteria by which is judged to be at its minimum are different for the different search methods.

The Levenberg-Marquardt search method combines the properties of the steepest descent and the Gauss-Newton methods. This is accomplished by adding a constant  $\lambda$  to the diagonal elements of the Hessian matrix that is associated with the gradient on the parameter space. If  $\lambda$  is large the search algorithm approaches the method of steepest descent. When  $\lambda$  is small the algorithm approaches the Gauss-Newton direction.

The method of steepest descent can optimally find major changes in the data and thus works best in the early stages of the fit when the residual of the sum of squares is changing substantially with each iteration. The Gauss-Newton method is best for smoothing out the fit in later stages when these residuals are no longer changing substantially (see Schreiner et al. 1985). The Levenberg-Marquardt method requires that the first derivative of the function  $f(x,P)$  be evaluated with respect to each parameter for each data point. These derivatives are used to evaluate the 'curvature' in the local parameter space in order to move in the direction of the perceived minimum. For predefined functions the exact derivative is calculated. For custom functions a numerical derivative (central difference) is computed using a step size of  $10^{-7}$ .

As the fit progresses some steps may result in a poorer (larger) value of the SSE. However, the general trend is a reduction in the SSE.

The Levenberg-Marquardt method does not report SSE during the fitting process but rather reports the standard deviation ( $\sigma$ ). However,  $\sigma$  follows the same trend as the SSE, that is, if the SSE increases then  $\sigma$  also increases, and vice-versa. In fact, the standard deviation is reported by all search methods, providing a standard criterion for judging the fitting quality regardless of the search method. The standard deviation is given by:

$$\sigma = \frac{\sum_{i=1}^n (Obs - Exp)^2}{n-1}$$

where  $n$  is the number of data points,  $Obs$  is the observed value and  $Exp$  is the expected value as calculated using the fitting function.

### Levenberg-Marquardt Convergence

Convergence is reached when the parameter change vectors go to zero, which occurs when a minimum is reached in the local parameter space. Because of this the fitting function might converge to a minimum that is not necessarily the lowest (global) minimum in the entire parameter response surface. Convergence to a local minimum often results in a poorly fitted curve and so is easily recognized. If you suspect that the fit has converged on a local minimum, you should specify new fitting seed values (graphically-assisted seeding is very useful here) and retry the fit. Alternatively, use a different fitting method. For example, the Simplex search method is not as prone to convergence at a local minimum.

The iterations are also stopped (convergence is assumed) when the change in the minimization function (for example, the SSE) is less than a preset value. This value can be set in the 'Precision' field in the **Function/Method** tab of the fitting dialog. The default value is  $10^{-6}$ .

Normally, it is preferable to allow the parameters to converge naturally. Convergence on a precision criterion can result in a poorer fit especially if the precision criterion is reached before the individual parameters have converged. On the other hand, some 'difficult' fits might require hundreds or even thousands of iterations if only the change vector criterion is used for convergence. In order to favor convergence on the basis of change vectors but to also allow difficult fits to converge on the basis of an acceptable 'precision' value, the fitting routine converges on the precision criterion only if this criterion has been met over at least 100 successive iterations. Given this criterion it is not likely that further improvements in the minimization function will lead to a better fit, so the iterations will stop.

### Levenberg-Marquardt Precision

The default Levenberg-Marquardt precision is  $10^{-6}$ .

The Levenberg-Marquardt precision sets the minimum absolute change in the minimization function (for example, the SSE) that signifies convergence. This minimum difference must be satisfied over at least 100 successive iterations. That is, if:

$$\text{absolute}(\text{SSE}(\text{old}) - \text{SSE}(\text{new})) < \text{Precision}$$

over 100 consecutive iterations, convergence is assumed. A less stringent precision value could facilitate convergence for a particularly difficult dataset, but often at the expense of fitting accuracy. In any case, the statistics of the fit should always be carefully evaluated in order to determine whether or not the fit is acceptable.

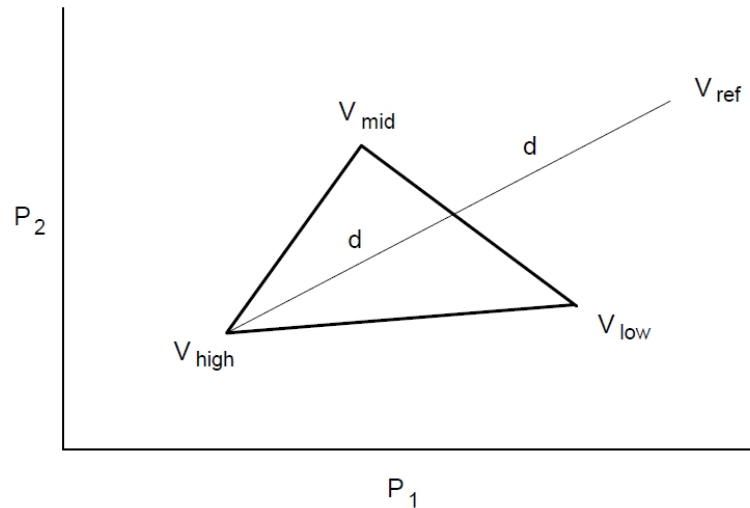
## Simplex Method

The Simplex method supports the least squares, mean absolute, maximum likelihood and minimax minimization methods. The explanation given here is for least squares minimization but the general principal is the same for all minimization functions.

The Simplex search method is based on the algorithm of Nedler and Mead (1965), and is an example of a direct search approach that relies only on the values of the function parameters. It does not consider either the rate or the direction by which the function is approaching the minimum on the parameter response surface. However, the direction in which the function parameters proceed is not purely random but rather relies on a clever strategy that takes advantage of the geometry of the response surface.



A simplex is a geometric figure that has one more dimension than the parameter space in which it is defined. The vertices of the simplex are first separated by adding an 'offset' to the initial seed values. The function to be minimized is then evaluated at each vertex to identify the lowest and highest response values. For example, a simplex on a two-dimensional space (corresponding to a two-parameter function) is a triangle that may have the following appearance:



**Figure 6-1: A simplex for a two-parameter function**

where  $P_1$  and  $P_2$  are the parameters,  $V_{high}$  is the vertex which has the highest (worst) function value,  $V_{low}$  is the vertex which has the lowest (best) function value and  $V_{mid}$  represents an intermediate function value. A 'downhill' direction is established by drawing a line from  $V_{high}$  through a point midway between  $V_{mid}$  and  $V_{low}$ . The algorithm then tries to find a point along this line that results in an SSE which is lower than the existing vertices.

The simplex changes shape by reflection, contraction, expansion or shrinkage. The first point tested is the reflected point  $V_{ref}$  which lies a distance of  $2d$  along the line from  $V_{high}$ . This reflected simplex is accepted if its response is neither worse than  $V_{high}$  nor better than  $V_{low}$ . If the response is better than  $V_{low}$  then the algorithm performs an expansion by moving a distance of  $4d$  along the line from  $V_{high}$ . The expansion is accepted if it has a lower (better) response than the previous best. Otherwise the reflection is accepted.

If the reflection results in a higher (worse) response than  $V_{high}$  then the algorithm tests a contraction by moving a distance of  $0.5d$  toward the midpoint on the line. If this produces a better response then the simplex is accepted; otherwise shrinkage occurs where all vertices except  $V_{low}$  move toward the midpoint by one-half of their original distance from it.

The advantage of the Simplex algorithm is that it is considerably less sensitive to the initial seed values than the gradient search algorithms. It will rapidly approach a minimum on the parameter response surface usually in the space of several tens of iterations for a multicomponent function.

The disadvantage of the Simplex fitting method is that its sensitivity does not increase when it is in the vicinity of a minimum on the parameter space.

Another problem can arise in that the Simplex algorithm may find what it perceives to be a local minimum but the fractional error is still greater than the convergence criterion (see below). In this case iterations may continue endlessly. To circumvent this problem, the fitting routine will stop when there is no change in the fractional error for 30 iterations. Even if the above error criterion is not met, the fit is assumed to have converged. This occurrence is reported as a 'Stable Fractional Error' in the Results Window. The displayed value of  $\sigma$  might be the same for many more than 30 iterations before the fitting routine is terminated. This is because  $\sigma$  is reported as a single precision value whereas the fractional error is a double precision value.

The weighting options are not available with Simplex fitting. This is because weighting interferes with the 'travel' of the simplex and greatly reduces the chances of convergence.

### Simplex Convergence

The Simplex algorithm can converge in one of three ways. The iterations are stopped when the fractional error is less than or equal to a preestablished 'precision' value.

The simplex is moved over the parameter space until the ratio of the response of the best and worst vertexes reaches a preset minimum fractional error, at which point the function is said to have converged:

$$\mathbf{FractionalError} = \frac{v_{high} - v_{low}}{v_{high}}$$

The quantities on the right-hand side of the equation are based on one of four minimization methods that Simplex can use, namely least squares, maximum likelihood, mean absolute or minimax.

The fractional error is computed for the simplex for each dimension (each having a  $V_{high}$  and  $V_{low}$  simplex). All simplexes must have a fractional error less than the value of precision for convergence.

### Simplex Precision

The default Simplex precision is  $10^{-5}$ .

The Simplex search is deemed to have converged when the fractional error is less than or equal to the Precision value. The fractional error can be based on one of four minimization methods, namely least squares, maximum likelihood, mean absolute or minimax.

## Variable Metric Method

The variable metric method supports the least squares and maximum likelihood minimization methods only.

Variable metric algorithms are designed to find either the minimum or the maximum of a multi-dimensional non-linear function  $f$ . The minimum is used in chi-squared or least squares applications and the maximum in maximum likelihood estimation.

Clampfit Batch Analysis typically uses minimization of least squares, which is asymptotically equivalent to likelihood maximization for most applications. The parameter values that determine the global minimum of  $f$  are optimal estimates of these parameters. The goal of the variable metric algorithm is to find these estimates rapidly and robustly by an iterative method.

Clampfit Batch Analysis uses the variable metric algorithm introduced by Powell (1978) and implemented by Dr. Kenneth Lange (UCLA). At each iteration the algorithm computes the exact partial derivative of  $f$  with respect to each parameter. The values of these derivatives are used in building an approximation of the Hessian matrix, which is the second partial derivative matrix of  $f$ . The inverse of the Hessian matrix is then used in determining the parameter values for the subsequent iteration. Variable metric algorithms have several desirable characteristics when compared with other methods of non-linear minimization. Like the simplex algorithm, variable metric algorithms are quite robust, meaning that they are adept at finding the global minimum of  $f$  even when using poor initial guesses for the parameters. Unlike simplex, however, convergence is very rapid near the global minimum.

### Variable Metric Convergence

The variable metric search method converges when the square of the residuals, or the maximum likelihood estimate if using likelihood maximization (see [Maximum Likelihood Comparison on page 97](#)), does not change by more than a preset value over at least four iterations. This value can be set in the 'Precision' field in the Function/Method tab of the fitting dialog. The default value is  $10^{-4}$ .

### Variable Metric Precision

The default variable metric precision is  $10^{-4}$ .

The variable metric search is assumed to have converged when the square of the residuals, or the maximum likelihood estimate if using likelihood maximization, does not change by more than the Precision value over at least four iterations.

## Chebyshev Transform

The Chebyshev technique is an extremely rapid, stable, noniterative fitting technique with a goodness of fit comparable to that of iterative techniques. It was developed by George C. Malachowski and licensed to Axon Instruments. The explanation in this section only describes how the Chebyshev transform is used to fit sums of exponentials.

The Chebyshev Transform transforms a dataset to the Chebyshev domain (in this case, domain refers to a functional basis, in which datasets are considered in terms of scaled sums of functions), using a method equivalent to transforming data to the frequency domain using Fourier transforms. Instead of representing the data using a sum of sines and cosines, the Chebyshev transform uses a sum of the discrete set of Chebyshev polynomials (described below).

Transforming the data allows it to be fitted with various functions of biological interest: sum of exponentials, Boltzmann distribution and the power expression (an exponential plus a constant raised to a power). For each of these functions, it is possible to derive an exact, linear, mathematical relationship between the fit parameters and the coefficients of the transformed data. If the dataset has noise present, as is almost always the case, these relationships provide estimates of the fit parameters. However, as the relationships are linear, high-speed regression techniques can be applied to find the parameters.

This method has the following properties: extremely fast fitting at a rate that is independent of noise, comparable goodness of fit to that of iterative techniques, and it always finds a fit.

At present this technique has one limitation: it can only be used on datasets that have equally spaced data points. This makes it inappropriate at present for fitting histogram data with variable bin widths.

## Orthogonal Polynomial Set

An  $N^{\text{th}}$  order polynomial is a function of the form:

$$P_N(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_Nx^N$$

where  $a_0, a_1, a_2, \dots, a_N$  are the coefficients of the polynomial, and  $a_N$  cannot be zero unless  $N$  is zero.

A set of polynomials consists of one polynomial from each order ( $P_0(x), P_1(x), P_2(x), P_3(x), \dots$ ), in which adjacent members of the set are related by a generating equation. Mathematicians usually represent a polynomial set with a single letter. In the case of the Chebyshev polynomials, the letter is 'T' (from previous anglicization of Chebyshev as Tchebysheff). A polynomial set is said to be orthogonal if every member in the set is orthogonal to every other member. In the case of the Chebyshev polynomials,  $T_0$  is orthogonal to  $T_1, T_2, T_3, T_4$ , and so on,  $T_1$  is orthogonal to  $T_2, T_3, T_4, T_5$ , and so on.

Orthogonal means at a right angle; synonyms are perpendicular and normal. Those familiar with vectors may recall that two vectors are tested for orthogonality using the dot product; two vectors are said to be orthogonal if their dot product equals zero. Similarly, two functions that are defined only at discrete points in time are said to be orthogonal if the sum of their product over all sampled points is zero. (This relation may only be true on a restricted range (for example,  $x \in [-1, 1]$ ), and with a weighting function present.)

## Transform Data to the Chebyshev Domain

All orthogonal function sets have the property of being able to represent a function in terms of a sum (or integral) of the members of the set. Depending on the function being represented, an appropriate scaling factor is chosen for each member of the set. A function  $f(t)$  can be represented as a sum over the Chebyshev polynomials by the relation:

$$f(t) = \sum_{j=0}^{\infty} d_j T_j(t)$$

where  $T_j(t)$  is the  $j^{\text{th}}$  member of the Chebyshev polynomial set and  $d_j$  is its scaling factor. In general, a sum of an infinite number of Chebyshev polynomials is required to represent a continuous function. In the case of a sampled dataset with  $N$  sampled points, a sum of only  $N$  Chebyshev polynomials, from order 0 to  $N-1$ , is required to exactly represent the dataset. In this case,  $t$  is not continuous, but is rather a set of data points  $t_j$ , where  $j$  runs from 0 to  $N-1$ . The above equation then becomes:

$$f(t) = \sum_{j=0}^{N-1} d_j T_j(t_i), \text{ for } i = 0, \dots, N-1. \quad (1)$$

This sum of polynomials exactly equals the function at all points in time, even though the individual members may only cross the function at a few points.

A function that has been represented this way is said to have been transformed to the Chebyshev domain, and the scaling factors (the  $d_j$ s) are usually referred to as the coefficients of the transform. Do not confuse these transform coefficients with the individual coefficients that make up each of the member Chebyshev polynomials! The member polynomials' coefficients never change; the coefficients of the transform change with  $f(t_j)$ .

## Calculate the Coefficients

The orthogonality property of the Chebyshev polynomials makes calculating the  $d_j$ s straightforward. Recall that every member  $T_k$  is orthogonal to every other member  $T_j$  for all  $k \neq j$ . To determine each coefficient ( $d_k$ ), both sides of the above equation are multiplied by  $T_k$ , then summed over all values of  $t$ ;

$$\sum_{i=0}^{N-1} T_k(t_i) f(t_i) = \sum_{i=0}^{N-1} T_k(t_i) \sum_{j=0}^{N-1} d_j T_j(t_i)$$

where  $T_k$  is the member whose coefficient is to be determined. Rearranging, and summing over all data points  $t_i$  eliminates all  $T_j$ s except  $T_k$ , leaving (after several steps):

$$\sum_{i=0}^{N-1} T_k(t_i) f(t_i) = d_k \sum_{i=0}^{N-1} T_k^2(t_i)$$

The summation on the right-hand side is usually written as a normalization factor ( $R_k$ ). Solving for  $d_k$ :

$$d_k = \sum_{i=0}^{N-1} T_k(t_i) f(t_i) / R_k. \quad (2)$$

## Discrete Chebyshev Polynomials

The generating equation for the Chebyshev polynomials is given by (Abramowitz and Stegun, page 792):

$$j(N-j) \cdot T_j(t) = (2j-1)(N-1-2t) \cdot T_{j-1}(t) - (j-1)(N-1+j) \cdot T_{j-2}(t)$$

where  $T_j(t)$  is the Chebyshev polynomial being generated, and  $N$  is the number of data points. It is clear that each  $T_j(t)$  depends on the previous two members in the set:  $T_{j-1}(t)$  and  $T_{j-2}(t)$ . The zero<sup>th</sup> member of this set is defined to be:  $T_0 = 1$ , from which all higher-order members may be derived.  $T_1$  and  $T_2$  are shown below:

$$T_1(t) = 1 - \frac{2}{(N-1)} t$$

$$T_2(t) = 1 - \frac{6}{(N-2)} t + \frac{6}{(N-1)(N-2)} t^2$$

Clearly,  $T_0$  is a horizontal line,  $T_1$  is a sloping line and  $T_2$  is a parabola.

## Isolate the Offset

The offset of a dataset can be isolated from the data in the Chebyshev domain. Consider the general case of a function containing an offset; we can rewrite this function in terms of its nonconstant and its constant parts:

$$f(t_i) = g(t_i) + K$$

where  $g(t_i)$  is the nonconstant part of the function. Using this function in Equation 2 we derive:

$$d_j = \left[ \sum_{i=0}^{N-1} T_j(t_i) g(t_i) / R_j \right] + \left[ \sum_{i=0}^{N-1} T_j(t_i) K / R_j \right].$$

The above equation is simply the sum of the Chebyshev transforms of  $g$  and  $K$ . This can be seen if we write the transform coefficients of  $g$  as  $d'_j(g)$ , and the transform coefficients of  $K$  as  $d''_j(K)$ :

$$d_j(g + K) = d'_j(g) + d''_j(K).$$

However,  $T_0 = 1$  implies that the Chebyshev transform of  $K$  is nonzero only for the zero<sup>th</sup> coefficient ( $d''_0(K)$ ). ( $K$  can be rewritten as  $KT_0$ , and  $T_0$  is orthogonal to all other Chebyshev polynomials.) We therefore have:

$$d_0 = d'_0(g) + d''_0(K),$$

$$d_j = d'_j(g), \quad \text{for } j > 0.$$

Once a dataset has been transformed to the Chebyshev domain, we can isolate the effect of the constant offset by not using  $d_0$  in our calculations of the other parameters.

### Transform an Exponential Dataset to the Chebyshev Domain

Suppose we wish to transform an exponentially decaying signal  $f(t_i)$  to the Chebyshev domain, where  $f(t)$  is defined as:

$$f(t_i) = a_0 + a_1 e^{-t_i/\tau}.$$

To exactly represent  $f(t_i)$  requires  $N$  coefficients, where  $N$  is the number of data points in the set. However, we can approximate  $f$  quite well with a sum of just a few polynomials. A pure, noiseless exponential with time constant 25 ms, sampled every 1 ms from  $t = 0$  to  $t = 255$  ms, when transformed to the Chebyshev domain has the following first 11 coefficients:

Chebyshev transform of $e^{-t_i/25ms}$					
$d_0$	0.0996188372	$d_1$	0.239563867	$d_2$	0.260135918
$d_3$	0.196609303	$d_4$	0.115307353	$d_5$	0.0552865378
$d_6$	0.0223827511	$d_7$	0.00782654434	$d_8$	0.00240423390
	$d_9$	0.000657552096	$d_{10}$	0.000161851377	

The magnitudes of the coefficients rise to a peak at  $d_2$ , then decline slowly as the coefficient index increases. Using these coefficients to approximate the exponentials by forming a sum of the first 11 Chebyshev polynomials approximates the original exponential to at worst  $4.5 \times 10^{-5}$ . Below is a graph of the exponential data, with the above Chebyshev fit superimposed upon it. Inset in this graph is a greatly magnified view of the residuals between the fit and the exponential. The polynomial-like oscillation of the residuals about the data.

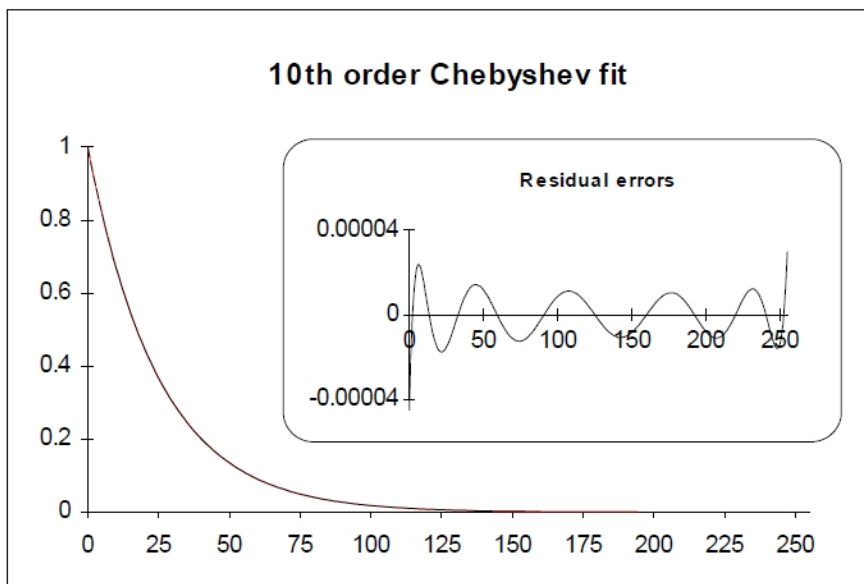


Figure 6-2: Graph of exponential data with Chebyshev fit superimposed.

Although 11 Chebyshev coefficients are sufficient to well approximate an exponential dataset, the presence of high frequency noise would require many more coefficients to accurately represent the data. If fewer coefficients are used, the approximation will appear filtered.

## Integrate an Exponential Function in the Chebyshev Domain

Let us say that we have a function  $f(t)$  that we wish to integrate, and that its integral is  $F(t)$ . In the discrete domain, where we only have a finite set of data points (usually evenly spaced), we write the discrete integral as:

$$\sum_0^{t-1} f(t') dt' = F(t).$$

The discrete integral  $F(t)$  is defined at each value  $t$  by the sum of the previous values from 0 to  $t-1$ . Using  $t-1$  as the end point of the integration serves to ensure that the forward difference equation is equal to  $f(t)$ :

$$F(t+1) - F(t) = f(t).$$

A difference equation in the discrete domain is analogous to a differential equation in the continuous domain. The equation shown above in the continuous domain would be expressed as  $dF/dt = f(t)$ . Forward difference refers to using  $t$  and  $t+1$  to form the difference.

How are the Chebyshev transforms of the integral  $F$  and  $f$  related? If we were to transform  $f$  to the Chebyshev domain, we would obtain a set of coefficients  $d_j(f)$ . Similarly, if we were to transform  $F$ , we would obtain a different set of coefficients  $d_j(F)$ . Comparing these two sets of coefficients,  $d_j(f)$  and  $d_j(F)$  we would find the relation:

$$D_j(F) = \frac{1}{2} \left[ \frac{(N+j+1)}{2j+3} d_{j+1}(f) - d_j(f) - \frac{N-j}{2j-1} d_{j-1}(f) \right], \quad (3)$$

**for all  $j > 1$ ,**

where  $D_j(F)$  is the  $j^{\text{th}}$  coefficient of  $F(t)$  and  $N$  is the number of points in the data. This equation cannot tell us the value of  $D_0(F)$ , as there is no  $d_{-1}(f)$  coefficient.

(This formula is derived in A Method that Linearizes the Fitting of Exponentials, G. C. Malachowski). This equation is critical to the use of this technique. Proof of this relation is long; those interested may refer to the appendix of the above paper. Briefly though, it can be described as follows: integrating the Chebyshev transform of a function is the same as the sum of the integrals of each of the Chebyshev polynomials making up the transformation. The integral of a polynomial is itself a polynomial. It turns out that after much simplification and rearrangement, each coefficient in the transform of the integral is a sum of the two adjacent coefficients in the transform of the original function.) If  $f$  is an exponential function, the following, very similar, relationship exists:

$$\frac{d_j(f)}{k} = \frac{1}{2} \left[ \frac{(N+j+1)}{2j+3} d_{j+1}(f) - d_j(f) - \frac{N-j}{2j-1} d_{j-1}(f) \right], \quad (4)$$

**for all  $j > 1$ ,**

where  $k$  is defined as:

$$k = e^{-1/\tau} - 1$$

or, solving for  $\tau$ :

$$\tau = \frac{-1}{\log_e(k+1)}. \quad (5)$$

Basically, these equations tell us that any adjacent triplet of Chebyshev coefficients forms an exact relationship that tells us the value of tau. Notice how Equation 4 further restricts the value of  $j$  to be greater than one: it is only true for those coefficients that do not contain the constant offset term.

Thus, integrating an exponential function in the Chebyshev domain allowed us to determine the value of tau. This is similar to the case of integrating an exponential function in the continuous domain:

$$\int e^{-t/\tau} dt = -\tau e^{-t/\tau} + C.$$

For reasons that will become clear in the section on the fitting of two exponentials, the right-hand side of Equation 4 can be written as  $d1_j(f)$ , which stands for the Chebyshev coefficients of the first integral of  $f$ . Equation 4 then becomes:

$$\frac{d_j(f)}{k} = d1_j(f), \text{ for } j > 1 \quad (6)$$

where  $k$  is as defined in Equation 5.

### Calculate Tau

Now we can calculate  $\tau$  using Equations 4 and 5. Choose any triplet of Chebyshev coefficients, and use those values in Equation 4 to get the value of  $k$ . Then use  $k$  in Equation 5 to calculate  $\tau$ . Every triplet has the same, redundant information built into it: triplet  $d_1, d_2, d_3$ , triplet  $d_2, d_3, d_4$ , and so on. The following is an example of the values of  $\tau$  predicted using the 8 triplets of the first eleven Chebyshev coefficients.



**Note:** Remember that  $d_0$  cannot be used as it contains information about the offset of the exponential, if present.

Chebyshev Calculations of $\tau$					
$d_1, d_2,$ $d_3$	25.00000003	$d_2, d_3,$ $d_4$	25.00000075	$d_3, d_4,$ $d_5$	24.99999915
$d_4, d_5,$ $d_6$	24.99999998	$d_5, d_6,$ $d_7$	25.00000096	$d_6, d_7,$ $d_8$	25.00000257
$d_7, d_8,$ $d_9$	25.00000464	$d_8, d_9,$ $d_{10}$	24.99999885		

The small differences between the calculated value and the actual value (25) are due to the limited precision of the coefficients used. In general, double precision numbers are used to calculate the fit parameters ( $\tau$ , the amplitude and the offset).

### Calculate the Amplitude

To determine the amplitude of the exponential, we must change directions completely, and generate an exponential dataset based on the value of tau just calculated. The generated dataset will have unity amplitude and zero offset:

$$g(t_i) = le^{-t_i/\tau} + 0.$$

Transforming this generated set to the Chebyshev domain will give us a different set of coefficients ( $d'_j(g)$ ). Now we can determine the value of the amplitude of our dataset by comparing the  $d_j(f)$ 's from our dataset to those of the generated set  $d'_j(g)$ 's. Recalling that the function that we are trying to fit is:

$$f(t_i) = a_0 + a_1 e^{-t_i/\tau}$$



we can transform this function to the Chebyshev domain as:

$$d_j(f) = \left[ \sum_{i=0}^N a_0 T_j(t_i)/R_j \right] + a \left[ \sum_{i=0}^N e^{-t_i/\tau} \cdot T_j(t_i)/R_j \right].$$

The Chebyshev coefficients for  $g(t_i)$  are very similar to those for  $f(t_i)$ :

$$d'_j(g) = \sum_{i=0}^N e^{-t_i/\tau} \cdot T_j(t_i)/R, \text{ for all } j.$$

Comparing these two equations yields the following relationship between the Chebyshev coefficients of  $f$  and those of  $g$ :

$$a_1 = \frac{d_j(f)}{d'_j(g)}, \text{ for } j > 0. \quad (7)$$

The amplitude is contained redundantly in all of the coefficients of the two transforms, excluding the zeroth coefficient. This redundancy is similar to that seen in the calculation of  $\tau$ .

$$a_1 = \frac{d_1(f)}{d'_1(g)} = \frac{d_2(f)}{d'_2(g)} = \frac{d_3(f)}{d'_3(g)} = \frac{d_4(f)}{d'_4(g)} = \dots$$

### Calculate the Offset

Calculating the offset is similar to calculating the amplitude: we compare the zeroth index coefficients from the two sets of transforms:

$$a_0 = d_0(f) - a_1 d'_0(g). \quad (8)$$

Unlike  $\tau$  and  $a_1$ , the offset information is not redundantly stored in the transform coefficients.

### Calculate the Fit Parameters in the Presence of Noise

If the dataset being fit contains noise, Equations 4, 5, 7 and 8 are no longer exactly true. For example, when calculating  $\tau$ , each triplet gives an estimate of  $\tau$ .

### Estimating $\tau$

Recall that for  $j > 1$ , Equation 4 shows a relationship between  $k$  and each triplet of Chebyshev coefficients. In the case of noise, this relationship is not strictly true;  $d_j(f)/k$  is an estimate of the right side of this equation. Equation 6 then becomes:

$$d_j(f) \cong k d_1_j(f).$$

The value of  $k$  that minimizes the following expression will be  $k'$ :

$$x^2 = \sum_{j=1}^n (d_j - k d_1_j(f))^2$$

where the sum generally does not include all  $N$  of the coefficients, but generally includes only those coefficients with a significant contribution to the transform; usually  $n$  is chosen to be 20. Expanding the squared term, differentiating with respect to  $k$ , setting the derivative to 0 and rearranging gives us our estimate of  $k$ :

$$k' = \sum_{j=1}^n d_j d_1_j(f) / \sum_{j=1}^n d_1_j^2(f). \quad (9)$$

Once  $k$  has been estimated, Equation 5 will give the corresponding best estimate of  $\tau$ .

## Estimating $a_1$

A similar technique is used to calculate the best estimate of the amplitude  $a_1$ . The ratios of the coefficients of our dataset ( $d_j(f)$ 's) to the coefficients of the pure exponential ( $d'_j(g)$ 's) now give estimates of the amplitude:

$$\frac{d_j(f)}{d'_j(g)} \cong a_1.$$

This can be rewritten as:

$$d_j(f) \cong a_1 d'_j(g).$$

As in the case of estimating tau, we form a linear regression equation to find the best estimate of  $a_1$  ( $a_1'$ ):

$$x^2 = \sum_{j=1}^n (d_j(f) - a_1 d'_j(g))^2.$$

Expanding the squared term, differentiating with respect to  $a_1$ , setting the derivative to 0 and rearranging gives us:

$$a_1' = \sum_{j=1}^n d_j(f) d'_j(g) / \sum_{j=1}^n (d'_j(g))^2. \quad (10)$$

## Estimating $a_0$

The estimate of  $a_0$  is calculated by substituting the above value of  $a_1'$  into Equation 8.

## Fit the Sum of Two Exponentials

In the two exponential case, two taus must be found. To do so, we shall integrate the function to be fit twice, and solve the resulting set of simultaneous equations for  $\tau_1$  and  $\tau_2$ . (This procedure is somewhat complicated.) Once we have the two taus, solving for the amplitudes and the offset is a simple extension of the procedure for fitting a single exponential.

$$f(t_i) = a_0 + a_1 e^{-t_i/\tau_1} + a_2 e^{-t_i/\tau_2}.$$

Let  $g_{\tau_1}(t_i)$  represent a unity amplitude exponential with time constant  $\tau_1$  and let  $g_{\tau_2}(t_i)$  represent a unity amplitude exponential with time constant  $\tau_2$ . Then we can write:

$$f(t_i) = a_0 + a_1 g_{\tau_1}(t_i) + a_2 g_{\tau_2}(t_i). \quad (11)$$

Transforming both sides of this equation gives us:

$$d_j(f) = \sum_{i=0}^n (a_0 + a_1 g_{\tau_1}(t_i) + a_2 g_{\tau_2}(t_i)) T_j(t_i) / R_j$$

or

$$d_j(f) = d_j^{offset}(a_0) + a_1 d_j^{\tau_1}(g_{\tau_1}) + a_2 d_j^{\tau_2}(g_{\tau_2})$$

where  $d_j^{offset}(a_0)$ ,  $d_j^{\tau_1}(g_{\tau_1})$  and  $d_j^{\tau_2}(g_{\tau_2})$  are the Chebyshev

coefficients of  $a_0$ ,  $g_{\tau_1}$  and  $g_{\tau_2}$ , respectively. To isolate the constant term from the calculations that follow, we shall only use coefficients where  $j > 0$ , yielding:

$$d_j(f) = a_1 d_j^{\tau_1}(g_{\tau_1}) + a_2 d_j^{\tau_2}(g_{\tau_2}), \quad \text{for } j > 0 \quad (12)$$

since the offset is only contained in the zero<sup>th</sup> coefficients.

### Use the Coefficients of $f$ and its Integrals to Determine the Taus

What if we were to integrate both sides of Equation 12? (since we are dealing with discrete data points, we use a summation). From Equations 4, 5 and 6, we know the coefficients of the integral of an exponential function, and we know how those coefficients are related to the coefficients of the exponential function itself. Applying those relationships to this sum of two exponentials case yields:

$$dl_j(f) = a_1 dl_j^{\tau_1}(g_{\tau_1}) + a_2 dl_j^{\tau_2}(g_{\tau_2}), \text{ for } j > 0 \quad (13)$$

or using Equation 6 to rewrite in terms of the coefficients of  $g_{\tau_1}$  and  $g_{\tau_2}$  themselves:

$$dl_j(f) = \frac{a_1}{k_1} d_j^{\tau_1}(g_{\tau_1}) + \frac{a_2}{k_2} d_j^{\tau_2}(g_{\tau_2}), \text{ for } j > 1 \quad (14)$$

where

$$\tau_1 = \frac{-1}{\log_e(k_1+1)} \text{ and } \tau_2 = \frac{-1}{\log_e(k_2+1)}. \quad (15a, b)$$

Integrating both sides of Equation 13 again:

$$d2_j(f) = \frac{a_1}{k_1} d1_j^{\tau_1}(g_{\tau_1}) + \frac{a_2}{k_2} d1_j^{\tau_2}(g_{\tau_2}), \text{ for } j > 2$$

where we write the Chebyshev coefficients of the second integral of  $f$  as  $d2_j(f)$ .  $j$  now must be greater than two. (The exact reason for this is beyond the scope of this description (see Malachowski's paper) however, briefly, it is required to isolate the effect of the offset from the calculation of the taus.) Substituting in Equation 6 again gives us our final relation that we need to determine  $\tau_1$  and  $\tau_2$ :

$$d2_j(f) = \frac{a_1}{k_1^2} d_j^{\tau_1}(g_{\tau_1}) + \frac{a_2}{k_2^2} d_j^{\tau_2}(g_{\tau_2}), \text{ for } j > 2. \quad (16)$$

### Solve a Set of Simultaneous Equations to Determine $k_1$ and $k_2$

Equations 12, 14 and 16 now form the three relations that we need in order to determine the taus. Rewriting them below we have three equations in three unknowns  $d_j$ ,  $d1_j$  and  $d2_j$ , and restricting  $j$  to be the same in all cases:

$$d_j(f) = a_1 d_j^{\tau_1}(g_{\tau_1}) + a_2 d_j^{\tau_2}(g_{\tau_2})$$

$$d1_j(f) = \frac{a_1}{k_1} d_j^{\tau_1}(g_{\tau_1}) + \frac{a_2}{k_2} d_j^{\tau_2}(g_{\tau_2})$$

$$d2_j(f) = \frac{a_1}{k_1^2} d_j^{\tau_1}(g_{\tau_1}) + \frac{a_2}{k_2^2} d_j^{\tau_2}(g_{\tau_2}), \text{ for } j > 2.$$

In order for these three equations to be simultaneously true, there must exist a pair of parameters  $x_1$  and  $x_2$  such that for all  $j > 2$ :

$$d_j + x_1 d1_j + x_2 d2_j = 0. \quad (17)$$

The solution to this equation is a straight line in  $x_1$ - $x_2$  coordinate space. To solve it, we add up the three simultaneous equations, and gather the like terms to find:

$$d_j + x_1 d1_j + x_2 d2_j = a_1 d_j^{\tau_1}(g_{\tau_1}) \left[ 1 + \frac{x_1}{k_1} + \frac{x_2}{k_1^2} \right] + a_2 d_j^{\tau_2}(g_{\tau_2}) \left[ 1 + \frac{x_1}{k_2} + \frac{x_2}{k_2^2} \right].$$

The values of  $x_1$  and  $x_2$  that satisfy this equation are:

$$x_1 = -(k_1 + k_2) \quad (18a, 18b)$$

$$x_2 = k_1 k_2,$$

as can be seen by substituting these values into the above equation.

Our strategy will be to solve for  $x_1$  and  $x_2$ , from them calculate the values of  $k_1$  and  $k_2$ , and finally use Equations 15a and b to calculate the corresponding values of  $\tau_1$  and  $\tau_2$ . To do so, we must first solve for  $k_1$  and  $k_2$  in terms of  $x_1$  and  $x_2$  (Equations 18a and b are the converse). There is no direct, algebraic method to do so, but we can recognize that Equations 18a and b are the roots of the quadratic polynomial:

$$k^2 + x_1 k + x_2 = 0 \quad (19)$$

as can be seen by factoring the polynomial into the product of its roots:

$$k^2 + x_1 k + x_2 = (k - k_1)(k - k_2).$$

This means that we can determine  $k_1$  and  $k_2$  by using  $x_1$  and  $x_2$  to form the above quadratic polynomial, and then solving for its roots. For a quadratic polynomial, we use the quadratic formula. For higher order polynomials, such as are used for fitting higher order exponentials, an iterative, root finding method is used.

What if there are not two, real roots? Recall that the quadratic formula either yields two real roots, one real root, or two complex roots. This corresponds geometrically to two crossings of the X axis, one tangential 'touch' of the axis, or no crossings of the axis.

If there is one real root, then the data being fit only consisted of a single exponential. In this case, this technique would yield two taus with the same value as that of the single exponential, and with amplitudes each one half of the amplitude of the single exponential.

If there are two complex roots, the data being fit is not a pure exponential. Rather, it is the product of an exponential and a harmonic function (for example, a cosine). This function is commonly called a ringing response, or an exponentially damped cosine. This can be seen by substituting a complex number into Equations 15a and b ( $a \pm bi$ ),

rewriting the resulting number in terms of a complex exponential ( $re^{i\theta}$ ),

where  $r$  is  $\sqrt{(a+1)^2 + b^2}$  and  $\theta$  is  $\tan^{-1}(b/(a+1))$ , taking the logarithm, substituting back into Equation 11, and simplifying.

substituting back into Equation 11, and simplifying.

### Finding the Taus in the Presence of Noise

In the presence of noise,  $f(t_j)$  is not an exact sum of exponentials, and therefore the Chebyshev coefficients  $d_j$ ,  $d1_j$  and  $d2_j$  do not lie along a straight line, but are scattered:

$$d_j + x_1 d1_j + x_2 d2_j \cong 0.$$

To find the best line through the data, we form the following regression equation, and minimize the  $\chi^2$  value:

$$\chi^2 = \sum_{j=2}^n (d_j - x_1 d1_j - x_2 d2_j)^2. \quad (20)$$

The best values for  $x_1$  and  $x_2$  are determined by expanding this relation, minimizing it first with respect to  $x_1$ , then with respect to  $x_2$ . After rearranging we have the following set of simultaneous equations:

$$\sum_{j=2}^n d_j d1_j = x_1 \sum_{j=2}^n d1_j^2 + x_2 \sum_{j=2}^n d1_j d2_j, \quad (21a, 21b)$$

$$\sum_{j=2}^n d_j d2_j = x_1 \sum_{j=2}^n d1_j d2_j + x_2 \sum_{j=2}^n d2_j^2.$$

Direct solution of simultaneous equations is a well known problem in mathematics; an iterative matrix technique is used here. This allows us to easily solve for the more difficult case of finding the solution to a set of  $m$  simultaneous equations, which must be used when fitting the sum of  $m$  exponentials.

### Find the Amplitudes of the Two Exponentials

Once the taus of the sum of exponentials are known, a technique similar to the single - exponential, noise present, case is used to find the amplitudes and the offset. (The corresponding two exponential case without noise is not shown.) We generate two exponential datasets based on the values of  $\tau_1$  and  $\tau_2$  just calculated; both datasets have unity amplitude and zero offset:

$$g_{\tau_1}(t_i) = 1e^{-t_i/\tau_1} + 0 \text{ and } g_{\tau_2}(t_i) = 1e^{-t_i/\tau_2} + 0.$$

Recalling from Equation 12 that the transform of  $f$  is the scaled transform of  $g_{\tau_1}$  and  $g_{\tau_2}$ , the resulting coefficients of each of these datasets is scaled and added together. In the presence of noise, however, this relationship is not

exactly true. Rather:

$$d_j(f) \cong a_1 d_j^{\tau_1}(g_{\tau_1}) + a_2 d_j^{\tau_2}(g_{\tau_2}), \text{ for } j > 0.$$

Linear regression of this equation yields the best possible values of  $a_1$  and  $a_2$  that satisfy:

$$\chi^2 = \sum_{j=1}^n (d_j(f) - a_1 d_j^{\tau_1}(g_{\tau_1}) - a_2 d_j^{\tau_2}(g_{\tau_2}))^2 \quad (22)$$

Solution of this equation is not shown, but involves expanding the square inside the summation, minimizing first with respect to  $a_1$  and then to  $a_2$ , and solving the resulting simultaneous set of equations for  $a_1$  and  $a_2$ .

### Find the Offset

Finally, to find the offset, a formula similar to Equation 8 is used; no regression need be done:

$$a_0 = d_0(f) - a_1 d_0^{\tau_1}(g_{\tau_1}) - a_2 d_0^{\tau_2}(g_{\tau_2}) \quad (23)$$

### Fit the Sum of Three or More Exponentials

Fitting the sum of three or more exponentials is a simple extension of the fitting the sum of two exponentials case. A full description is not given here.

### Speed of Fit

In tests the Chebyshev technique fit speed was completely unaffected by noise. There was a slight dependence on the number of exponentials being fit.

In contrast to these results, with iterative techniques, fitting a sum of two exponentials usually requires twice as much time, fitting a sum of three exponentials requires three times as much time, and so on.

## Goodness of Fit

In tests comparing the Chebyshev method to the Simplex iterative search method, both yielded the same values of the fit parameters in low noise and no noise conditions. The tests added varying amounts of noise to exponentials generated from known values. As the noise levels increased, these two methods produced slightly different values of the fit parameters for each test case, although the average of the parameters was the same. At extremely high levels of noise (the peak to peak noise reached 30% of the peak of the exponential), the Chebyshev search clearly did not fit as well as the Simplex method, in those times that the iterative Simplex converged at all.

Like all fitting methods, the Chebyshev method fits sums of exponentials data best if the dataset spans several times the largest time constant in the exponential. Although the Chebyshev method consistently outperforms other iterative techniques in this regard, even it can generate incorrect fits in this situation (for example, trying to fit an exponential function with a time constant of 2000 ms to a dataset spanning just 10 ms!). In particular, as the amount of noise increases, its ability to fit an insufficiently sampled dataset decreases.

The Chebyshev method performs most poorly when fitting data with extremely low frequency signals present. This may occur under the following circumstances: 60 Hz noise present (or other low frequency noise) or insufficiently averaged data (as may occur by forming a macroscopic current from an insufficient number of single channel records: for example, the single channel events may still be seen). This occurs since low frequency noise will appear most strongly in the low index Chebyshev coefficients, the same coefficients that contain most of the information of the exponential. Although iterative techniques do not perform well in this case either, when they converge they do so to the correct result more often than the Chebyshev technique.

## Success of Fitting

The Chebyshev technique very rarely fails to find a fit to the data when fitting exponentials or sums of exponentials, as it uses mathematical relationships and linear regression techniques. Experimentally though, it sometimes fails to find a good fit (see above conditions) and can even fail altogether with some kinds of data (for example, if the x and y data values are identical). In particular, since the Chebyshev technique always finds an answer so quickly, it is tempting to assume that this answer is always correct. Be sure to always compare the fitted data to the dataset.



**Note:** The Chebyshev method can fail when fitting shifted Boltzmann or exponential power functions, although the failure in these cases is expected to be very rare. In the event of a failure, all function parameters will reported as zero.

---

## Fit Sums of Exponentials to Non-Evenly Spaced Datasets

At present, the Chebyshev method only works with datasets having equally spaced points, since the relations derived here depend upon that assumption. With datasets that were sampled at two different rates (split clock acquisition), you must first use the **Analyze > Interpolation** command to fill in the missing data points using linear interpolation.



**Note:** At the time of this writing the Chebyshev search method had not been published in a peer reviewed journal. While Molecular Devices has conducted many empirical tests that have confirmed the accuracy of the fits, you should occasionally compare Chebyshev fitted results to those of one or more of the other fitting methods.

---

## Maximum Likelihood Estimation

Data are fitted by maximizing the logarithm of the likelihood with respect to a set of fitting parameters. Exponentially distributed data ( $t_i$ ) are described by one or more components ( $k$ ) of the form:

$$F(t_i) = \sum_{j=1}^k a_j e^{-t_i/\tau_j} \quad (1)$$

where  $t_1, t_2 \dots t_n$  are the  $n$  measured data points and  $\tau_j$  is the time constant of the  $j^{\text{th}}$  component. Each  $a_j$  is a fraction of the total number of events represented by the  $j^{\text{th}}$  component where:

$$\sum a_j = 1.0$$

The likelihood ( $L$ ) of obtaining a given set of observed data  $t_i$ , given the form of the distribution and the set of function parameters (denoted here by  $\theta$ ) is the product of the probabilities of making each of the  $N$  observations:

$$f(t_i) = \frac{d}{dt_i} F(t_i) = \sum_{j=1}^k a_j \tau_j^{-1} e^{-t_i/\tau_j} \quad (2)$$

As the likelihood typically takes on very small values the numerical evaluation of its logarithm is preferable. In practice, limited frequency resolution of the recording system makes it impossible to record events shorter than  $t_{\min}$ . To generalize this case it is also assumed that no events longer than  $t_{\max}$  can be measured. Taking these corrections into consideration, the conditional PDF is given by:

$$L = \prod_{j=1}^N f(t_i/\theta)$$

where:

$$L(\theta) = \sum_{i=1}^N \ln [f(t_i|\theta) / p(t_{\min}, t_{\max}|\theta)]$$

is the probability that the experimentally measurable dwell times fall within the range delimited by  $t_{\min}$  and  $t_{\max}$  given the probability distribution with parameters  $\theta$ . In Clampfit Batch Analysis  $t_{\min}$  and  $t_{\max}$  are defined by the lower and upper limits of the data's time base.

The fitting algorithm used here is not, strictly speaking, a maximum likelihood estimation. It is an iteratively reweighted least squares fit to the number of elements in a bin (which should converge on the maximum likelihood estimates of the parameters in this case). The weighting here has to do with the expected variance of the number of elements in a bin, which is Poisson distributed. So the weighting factor is the inverse of the variance (remember that mean = variance in a Poisson distribution). The iterative aspect has to do with the fact that the number of elements per bin gets moved around to correct for censoring (i.e., short events are not measurable). Because of the simple form of the 'weighted' sum of exponentials (that's a different 'weight'), the derivatives of the minimized function with respect to each parameter can be written directly and corrected for this Poisson distributed variance.

## Maximum Likelihood for Binned Data

The log likelihood  $L_b$  of observing a particular set of bin occupancies  $n_i$  for a given set of dwell - time data of times  $t_i$  is calculated by:

$$L_b(\theta) = \sum_{i=1}^k n_i \ln \left\{ \frac{F(t_{i+1}|\theta) - F(t_i|\theta)}{p(t_i, t_k|\theta)} \right\} \quad (1)$$

where  $F(t_i)$  and  $F(t_{i+1})$  are the probability distributions at the lower and upper bounds of the  $i^{\text{th}}$  bin using the parameter values  $\theta$ , and  $p(t_s, t_k)$  is the probability that the experimental dwell times fall within the range ( $k$ ) of the histogram where

$$p(t_s, t_k | \theta) = F(t_s | \theta) - F(t_k | \theta) \quad (2)$$

In Clampfit Batch Analysis, the probability distribution function (equation 1), rather than the probability density function (equation 2), is used in the calculations.  $F(t)$  is evaluated at each bin edge. The difference gives the probability that an event falls in the bin. This is equivalent to integrating the PDF over the width of the bin.

For a sum of  $m$  exponential components the probability distribution function is given by:

$$F(t | \theta) = 1 - \sum_{i=1}^m a_i e^{-t/\tau_i}$$

where  $\theta$  is the entire set of coefficients comprising the fraction of the total number of events in each  $i^{\text{th}}$  component  $a_i$  and the time constants  $\tau_i$ . The coefficients  $a_i$  sum to unity, where:

$$\sum_{i=1}^m a_i = 1$$

Therefore, the set of parameters  $\theta$  has  $2m-1$  degrees of freedom. Clampfit Batch Analysis can use either the Simplex or variable metric search methods to find the set of parameters that maximize  $L_b(\theta)$ . Only the variable metric method constrains the coefficients  $a_i$  to sum to unity. In the Simplex method, these parameters are not constrained and the set of parameters has  $2m$  degrees of freedom.

Maximum likelihood will operate on either logarithmically binned or conventionally binned histograms. It is, however, recommended that logarithmically binned data be used if MLE is the fitting method. With conventional binning, it is often not possible to select a bin width that can represent the data satisfactorily if the time constants are widely spaced. This problem can be avoided by binning the data into the variable width bins of a logarithmic histogram.

### EM Algorithm

The EM (Expectation step – Maximization step) algorithm computes maximum likelihood parameter estimates for binned observations from a mixture of exponentials. This algorithm is used in Clampfit Batch Analysis to estimate initial seed values for maximum likelihood estimates for fitting exponential probability functions.

## Model Comparison

In general, with non linear fitting routines, when the order of the fitting function is increased the fitted curve will appear to improve (up to a point, of course, until it becomes difficult to distinguish between successive order fits). Although it is often possible to choose one fit over another by visual inspection, this is not always the case. Moreover, visual inspection is not an adequate substitute for a statistical comparison, especially if two different models (for example, different orders) produce fits that are visually very similar. Clampfit Batch Analysis provides a means of statistically comparing different models that have been fitted with either a least squares routine or with maximum likelihood.

When model comparison is selected, the text ‘compare models’ appears next to the function name above the equation window. If the selected function does not support model comparison, the model comparison status text is blanked out.



**Note:** Fixed fitting function parameters are not allowed when comparing models.



## Maximum Likelihood Comparison

Suppose you wish to compare two models,  $F$  and  $G$ , which have been fitted using the maximum likelihood method. The probability densities for these models are given by  $f(x, \theta)$  and  $g(x, \beta)$  where  $x$  is the dataset, and  $\beta$  and  $\theta$  are the function parameters with dimensions  $k_f$  and  $k_g$  (where  $k_g > k_f$ ). The natural logarithm of the likelihood ratio (LLR) for  $F$  and  $G$  is defined as:

$$LLR = \log \left\{ \frac{\sup_{\beta} f(x, \beta)}{\sup_{\theta} f(x, \theta)} \right\} = \log \left\{ \frac{g(x, \beta)}{f(x, \theta)} \right\}$$

where  $\beta$  and  $\theta$  are the parameter values (maximum likelihood estimates) that maximize the likelihood for each probability density. The suprema of  $f(x, \theta)$  and  $g(x, \beta)$  are denoted by  $\sup_{\theta} f(x, \theta)$  and  $\sup_{\beta} g(x, \beta)$ , respectively. When model  $F$  is true,  $2LLR$  has a Chi square distribution with  $k_g - k_f$  degrees of freedom.;

For example at a confidence level of 0.95 ( $p < 0.05$ ) for  $k_g - k_f = 2$  degrees of freedom (as is always the case between successive models) Chi square is 5.991. Therefore, if  $2LLR < 5.991$  then it is assumed that model  $G$  does not represent a significant improvement over model  $F$  and model  $F$  is selected as the best fit.

## Least Squares Comparison

In the case of least squares fitting (Simplex or Levenberg Marquardt fitting methods) the SSE for models  $F$  and  $G$  is defined as:

$$SSE_f = \sum_{i=1}^n [x_i - f(x_i|\theta)]^2 \text{ for model } F$$

and:

$$SSE_g = \sum_{i=1}^n [x_i - g(x_i|\beta)]^2 \text{ for model } G$$

where  $x_i$  are the  $n$  data points,  $f(x_i|\theta)$  and  $g(x_i|\beta)$  are the values predicted by models  $F$  and  $G$ , respectively, and  $\theta$  and  $\beta$  are the set of function parameters that minimize SSE.

To compare models  $F$  and  $G$  a form of the likelihood ratio test is used. In the case of least squares fitting the statistic  $T$  is defined by:

$$T = \frac{SSE_f - SSE_g}{SSE_g}, \frac{n - k_g}{k_f}$$

where  $SSE_f$  and  $SSE_g$  are the sums of squared errors for models  $F$  and  $G$ , respectively.  $T$  has an  $F$  distribution which can be compared with a standard  $F$  distribution table, with  $k_f$  and  $n - k_g$  degrees of freedom. This statistic is denoted in Clampfit Batch Analysis by ' $F$ '.



**Note:** The degrees of freedom ( $k_f$  and  $n - k_g$ ) will be different for each successive model.

## Define a Custom Function

The rules for defining a custom function and associated issues are as follows:

Only one dependent variable is allowed. The dependent variable is already expressed, as is the equal sign, so these should not be entered as part of the equation. Only the right hand side of the equation should be specified.

When fitting from Analysis window data, only one independent variable is allowed. This variable must be expressed as  $x1$  (or  $X1$ ).

When fitting from a Results window, up to six independent variables are allowed. These variables must be expressed as  $x1...x6$  (or  $X1...X6$ ).

The maximum number of function parameters is 24. Parameter names must be expressed as p1...p24 (or P1...P24).

The maximum length of the function including parenthesis is 256 characters.

Parameter labels (p), independent variable labels (x) and mathematical operations such as log, sin, etc. are case insensitive, so you may use either lower or upper case when specifying these labels or operations.

Automatic seeding is not available for custom functions. You must specify all seed values before the fit will start. If you try to start fitting (by clicking **OK**) before all parameter seeds have been specified you will receive an error message.

Graphically assisted seeding is not currently available when fitting to data from a Results window. If you absolutely require graphical seeding for fitting Results window data, you can first save the data to an ATF file, then import the ATF data into an Analysis window for graphical seeding and fitting. However, the data can contain only positive, uniformly spaced independent variable values (X axis data) for proper display in the Analysis window. Also, keep in mind that only a single independent variable can be specified for fitting from the Analysis window.

The custom function is compiled when switching from the Function/Methods tab to either of the other tabs in the fitting dialog. Once compiled successfully, the equation will appear on the line above the equation window in the fitting dialog. If there is an error in the expression, compiler warnings will be issued.

Be careful with parentheses. For example, be aware that  $2*x1+p2$  is not the same as  $2*(x1+p2)$ . In the former case  $2*x1$  is evaluated before  $p2$  is added to the product, whereas in the latter case,  $x1+p2$  is evaluated before the multiplication is performed.

## Multiple-Term Fitting Models

A fitting model can be expressed as a sum of identical functions (terms). Each term in a multiple term model will be assigned a 'weight' or 'amplitude' that will reflect the contribution of that term to the fitted curve. For example, a two term standard exponential will be of the form

$$f(x) = A_1 e^{-t/\tau_1} + A_2 e^{-t/\tau_2} + C$$

where  $A_1$  and  $\tau_1$  are the amplitude and time constant, respectively, for the first term and  $A_2$  and  $\tau_2$  are the amplitude and time constant, respectively, for the second term. The variable  $C$  is a constant offset term along the Y axis.

Multiple terms for custom functions must be explicitly defined within the custom function itself, for example, the above two term exponential function would be specified as

$$f(x) = p1 * \exp(-x1/p2) + p3 * \exp(-x1/p4) + p5$$

where  $p1$  and  $p3$  are the amplitudes,  $p2$  and  $p4$  are the time constants and  $p5$  is the constant offset term.

## Minimization Functions

### Sum of Squared Errors

Levenberg Marquardt, variable metric and Simplex only. The function to be minimized is

$$SSE = \sum_{i=1}^N (y_i - y)^2 = \sum_{i=1}^N [y - f(x, P)]^2$$

where SSE (sum of squared errors) is the sum of the squares of the difference between the data  $y_i$  and the fitting function  $y = f(x, P)$  with a set of parameters  $P$  to be fitted over  $N$  data points. The optimal values for  $P$  are assumed to occur when SSE is at a minimum. Weighting may or may not be applied to modify this function.

### Maximum Likelihood Minimization

*Variable metric and Simplex only.*

Maximum likelihood estimation (MLE) is available only for the standard and log transformed probability exponential functions and only with the variable metric or Simplex search methods. Moreover, these functions are intended to be used with binned data. That is, the dependent variable values (X axis data) are assumed to be bin center values. Strictly speaking, the likelihood is maximized so the use of ‘minimization method’ might be deemed inappropriate here. However, the fitting method minimizes the negative of the log likelihood value, which is equivalent to maximizing the positive log likelihood.

See [Maximum Likelihood Estimation on page 95](#) for a description of the algorithm.

### Mean Absolute Minimization

*Levenberg Marquardt and Simplex only.*

Mean absolute minimization is a linear criterion that weights a badly-fitted data point proportionately to its distance from the fitted curve, rather than the square of that distance. If some points have substantially more error than others then the best sum of squares fit might deviate from the more reliable points in an attempt to fit the more reliable ones. The mean absolute error fit is influenced more by the majority behavior than by remote individual points. If the data are contaminated by brief, large perturbations, mean absolute error minimization might perform better than sum of squares minimization.

The function  $E$  to be minimized is

$$E = \text{abs} \left( \sum_{i=1}^N (y_i - y) \right) = \text{abs} \left( \sum_{i=1}^N [y_i - f(x, P)] \right).$$

### Minimax Minimization

*Levenberg Marquardt and Simplex only.*

Minimax minimization yields a fit in which the absolute value of the worst fitted data point residual is minimized. This might be useful for certain datasets when the fit must match the data within a certain tolerance. The function  $E$  to be minimized is

$$E = \text{abs}(\text{max}(y_i - y)) = \text{abs}(\text{max}([y_i - f(x, P)]))$$

where  $\text{max}(y_i - y)$  is the largest absolute difference between the data and the fit.

## Weighting

For the search methods that support least squares minimization (Levenberg Marquardt, variable metric and Simplex) the sum of the squares (SSE) of the difference between the data ( $f_i^{obs}$ ) and fitted curve ( $f_i(\theta)$ ) is minimized, where

$$SSE = \sum (f_i^{obs} - f_i(\theta))^2.$$

This works quite well when uncertainties are random and not related to the time or number of counts. However, the uncertainty observed is often a function of the measured value  $y$ , such that for larger values of  $y$  there are substantially greater deviations from the fit than for smaller values of  $y$ . To compensate for this, weighting becomes quite useful during the assessment of the fit, where:

$$SSE = \sum \left[ \left( \frac{f_i^{obs} - f_i(\theta)}{f_i(\theta)} \right)^2 \right].$$

### None

In this case the denominator  $f(\theta)$  is 1.

### Function

Weighting the sum of squared errors function generates the Chi square ( $\text{Chi}^2$ ) function. The  $\text{Chi}^2$  value for a function with a given set of parameters  $\theta$  is given by:

$$\text{Chi}^2 = \sum_{i=0}^m \frac{f_i^{obs} - f_i(\theta)}{f_i(\theta)}$$

where  $m$  is the number of points in the histogram fitting range,  $f_i(\theta)$  is the fit function calculated for the  $i^{\text{th}}$  data point, and  $f_i^{obs}$  is the observed value of the  $i^{\text{th}}$  point.

### Data

This is a modified form of the  $\text{Chi}^2$  function, where:

$$\text{Modified Chi}^2 = \sum_{i=0}^m \frac{f_i^{obs} - f_i(\theta)}{f_i^{obs}}$$

where  $m$  is the number of points in the histogram fitting range,  $f_i(\theta)$  is the fit function calculated for the  $i^{\text{th}}$  data point, and  $f_i^{obs}$  is the observed value of the  $i^{\text{th}}$  point.

### Bin Width

Weighting by the bin width weights the sum of squared errors by the width of each  $i^{\text{th}}$  histogram bin, such that:

$$SSE_b = \sum_{i=1}^m \frac{f_i^{obs} - f_i(\theta)}{x_i - x_s}$$

where  $x_i$  is the right bin edge value and  $x_s$  is the left bin edge value. Bin width weighting is allowed only for the predefined log transformed exponential function that assumes log binned data. It is not available for custom functions.

The selected weighting type will also be applied to mean absolute and minimax minimization.



**Note:** To apply different weighting criteria, you can export the data to a results sheet, modify the data using column arithmetic and subsequently fit the data directly from the results sheet.

## Normalized Proportions

The normalized proportion is the absolute value of that parameter divided by the sum total of the absolute values of all of the proportion terms, that is:

$$P_{norm_k} = \frac{P_{abs_k}}{\sum_{j=1}^n P_{abs_j}}$$

Normalized proportions are most likely to be specified with the exponential probability functions or the Exponential, weighted/constrained function. The variable metric method (and only this method) constrains the proportion terms to sum to 1.0 during fitting (see [Variable Metric Method on page 82](#)), but only for the standard or log transformed exponential probability functions when using maximum likelihood or the weighted/constrained exponential with least squares.

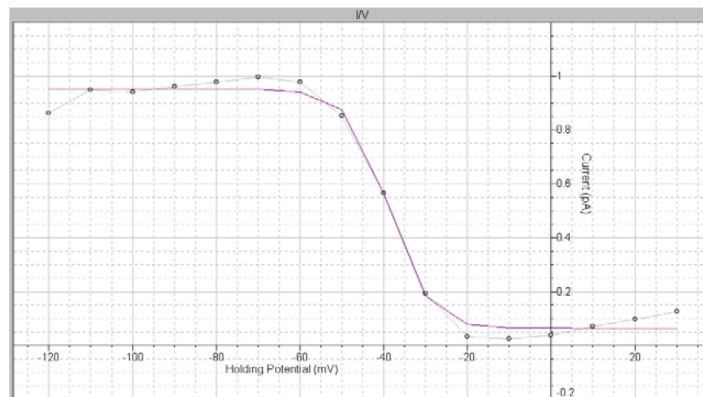
## Zero-shifting

In cases where the natural origin for a dataset is ambiguous or inappropriate, it would be desirable to shift the origin of the data that are to be fitted to zero. For example, if a time constant of an exponential curve is to be extracted from an arbitrary segment of a large dataset it would be reasonable to force the selected fitting range to start at zero. In the same vein, it might also be desirable to set a zero origin just after a stimulus artifact. To this end, the 'zero shift' option is provided. If `zeroshift` is enabled then for a set of  $i$  data points  $x_i$ , each point  $x$  is offset such that

$$x = x_i - x_0 \text{ where } x_0 \text{ is the value of the first data point.}$$

However, it is important to note that zero shifting can affect the values of some fitted parameters. Consequently, in some instances, zero shifting the data might not be appropriate and could lead to unexpected results. For example, when fitting a 'Z delta Boltzmann' to current/voltage data the parameter  $V_{mid}$  (the voltage at which the current is half maximal) will differ depending on whether or not zero shift is enabled. In the following example, the data were zero shifted prior to fitting.

[Figure 6-3](#) shows a fit of the Z delta Boltzmann function to current/voltage data. The fitted curve is the line without symbols.



**Figure 6-3: Fit of Z delta Boltzmann function to current/voltage data.**

In this fit  $V_{mid}$  is reported as +81.16 but from the data one would expect  $V_{mid}$  to be about -40 mV. Nevertheless, the value of +81.16 is in fact correct because the first point (originally at -120 mV) has been forced to zero such that the actual fitting range is from 0 to +140 mV. On this scale, the reported positive value is a reasonable estimate of  $V_{mid}$ .

If zero shift is disabled, the fitted curve looks exactly the same but  $V_{mid}$  is now reported as -38.84 mV. The other parameters of this function are identical in both cases. As the expected value for this dataset is indeed in the range of -40 mV, it is clear that in this particular case it would not be appropriate to use zero shifting. In fact, if zero shift is inadvertently enabled it would appear that the fitting routine is malfunctioning, which is not the case. It is, therefore, very important that the status of the `zeroshift` option be known at all times when fitting.



Clampfit Batch Analysis provides a large selection of predefined fitting functions to assist in data analysis. Some functions require that restrictions, such as forced positive parameters, are applied during the fit. Accordingly, some options in the Curve Fit dialog might be automatically set to comply with mandatory fitting requirements. Furthermore, some functions require automatic preprocessing of the data, such as normalization, prior to fitting. In the function descriptions below, any automatic data preprocessing and forced option requirements for a function are listed on the line above the function formula.

The references cited provide sources for more detailed explanation of the functions and examples of specific experimental applications.

### Beta Functions

$$f(x) = a^{a-1}(1-x)^{b-1}/B(a,b) + C$$

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1} dx$$

$$a = \alpha\tau, b = \beta\tau$$

*Requires a normalized X axis.*

*All parameters except the offset (C) are forced positive.*

The beta function is used to describe the steady state filtered amplitude distribution of a two - state process, where  $\alpha$  and  $\beta$  are rate constants and  $\tau$  is the time constant of a first order filter. This method has been used to measure block and unblock rates in the microsecond time range even under conditions when individual blocking events were not time resolved by the recording system.

This function describes a probability distribution that requires the X axis data range to be 0 to 1. If the data do not meet this criterion the X axis values are automatically normalized prior to fitting. These values are rescaled after fitting so that the fitted curve will conform to the original data.

The beta function is intended to be used primarily with data that have been imported into a Results or Graph window, and therefore has limited utility with respect to time based Analysis window data.

The fit solves for parameters  $a$ ,  $b$ ,  $B(a,b)$  and the constant  $y$  offset  $C$ . The rate constants  $\alpha$  and  $\beta$  can be obtained by dividing  $a$  and  $b$ , respectively, by the filter time constant,  $\tau$ . Clampfit Batch Analysis does not perform this calculation.

The recommended fitting method is Levenberg Marquardt.

## Binomial

$$f(x) = \frac{n!}{(n-x)!x!} P^x (1 - P)^{n-x}$$

Requires an integral X axis.

Requires normalized data.

Maximum number of points = 170.

The parameter P is forced positive.

The binomial distribution describes the probability,  $P$ , of an event occurring in  $n$  independent trials. For example, this function can be used to determine the probability for a particular number of  $n$  independent ion channels being open simultaneously in a patch (Colquhoun, et al. 1995). This function has also been applied to quantal analysis of transmitter release.

This function requires integer values of  $x$ . If the X axis values are not integers then they are converted to such prior to, and rescaled following, fitting. The ordinate data are also normalized so that they range from 0 to 1, and are rescaled after fitting so that the fitted curve conforms to the original data. Rescaling is such that the area under the fitted curve is equal to the area under the original data curve.

The number of sample points for this function is limited to 170 to conform to the computational limit for a factorial.

The binomial function has limited utility with respect to time based Analysis window data. It is intended to be used primarily with data that have been imported into a Results or Graph window.

The fit solves for the probability variable,  $P$ . Since the X axis scale is integral the fitted curve will appear as a histogram.

The recommended fitting method is Levenberg Marquardt.

## Boltzmann, Charge Voltage

$$f(x) = \frac{I_{max}}{1 + e^{(V_{mid} - V)/V_c}} + C$$

This function can be used to examine activation and inactivation profiles of voltage gated currents.

The charge voltage Boltzmann distribution is given by

$$Q_{on}(E) = Q_{on} - max / [1 + exp((E_{mid} - E)/K)] + C$$

where  $Q_{on} - max$  is the maximum charge displaced,  $E_{mid}$  is the potential at which  $Q_{on} = 0.5 \cdot Q_{on} - max$ ,  $K$  is the number of millivolts required to change  $Q_{on}$   $e$  fold, and  $E$  is the measured potential.  $C$  is a constant offset term. This function can be used to fit current voltage curves of the form:

$$I = I_{max} / [1 + exp((V_{mid} - V)/V_c)]$$

or

$$g = g_{max} / [1 + exp((V_{mid} - V)/V_c)]$$

where  $V$  is the membrane potential,  $V_{mid}$  is the membrane potential at which the current is half - maximal, and  $V_c$  is the voltage required to change  $I$  or  $g$   $e$  fold. If  $I$  or  $g$  are normalized then the data points should be input as  $I/I_{max}$  or  $g/g_{max}$  and the dependent variable  $I_{max}$  or  $g_{max}$  in the function above should be fixed to 1.0.



The fit solves for  $I_{max}$  (or  $G_{max}$ ),  $V_{mid}$ ,  $V_c$  and the constant  $y$  offset  $C$ .

The recommended fitting method is Levenberg Marquardt. The variable metric method also works well with this function but is slower to converge.

### Boltzmann, Shifted

$$f(x) = \sum_{i=1}^n \frac{A_i}{1 + B e^{(-x/\tau_i)}} + C$$

Like the standard Boltzmann, this function also defines a sigmoidal curve.  $A$  is the amplitude,  $\tau$  is the 'slope' and  $C$  is a constant offset in the  $y$  direction (see Boltzmann, standard). However, unlike the standard Boltzmann function, the shifted Boltzmann includes an offset parameter  $B$  that shifts the curve along the  $X$  axis such that the half maximal amplitude is at  $x = -\ln(1/B) \tau$ . Thus, this function is to be used when fitting a sigmoidal curve to data where the half amplitude point is expected to be displaced from zero along the  $X$  axis.

The fit solves for  $A$ ,  $B$ , and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt or Chebyshev if fitting only a single term.

### Boltzmann, Standard

$$f(x) = \sum_{i=1}^n \frac{A_i}{1 + e^{(-x/\tau_i)}} + C$$

This function defines a sigmoidal curve. Unlike the shifted Boltzmann function, this function does not include an offset parameter along the  $X$ -axis.

The physical correlates for  $A$  and  $x$  are not specified in this equation. It is up to the user to define these quantities. For example,  $A$  might be conductance and  $x$  might be cell membrane voltage. The parameter  $\tau$  is the slope of the function that specifies the change in  $x$  required to produce an  $e$  fold change in  $A$ .  $A$  is half maximal at  $x = 0$  (where  $f(x) = A/(1 + e^{-0/\tau}) = A/2$ ).

Consequently, the fitted curve is sigmoidal only if there are both positive and negative  $x$  data with the half amplitude at or very close to zero. If the half amplitude is offset significantly from zero the shifted Boltzmann function should be used.

The fit solves for the amplitude  $A$ , the width  $\tau$  and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

## Boltzmann, Z Delta

$$f(V) = V_{min} + \frac{V_{max} - V_{min}}{1 + e^{\frac{Z_d F}{RT}(V - V_{mid})}}$$

This function can be used to analyze the voltage dependence of gating charges in ion channels (Hille, 1992).

$V_{min}$  and  $V_{max}$  are the minimum and maximum voltages,  $Z_d$  is the magnitude of the charge valence associated with the electric field  $V$ ,  $V_{mid}$  is the voltage at which  $f(V)$  is half maximal,  $F$  is the Faraday constant,  $R$  is the Gas constant,  $T$  is absolute temperature. The temperature is optionally specified (in °C).

The fit solves for  $V_{max}$ ,  $V_{min}$ ,  $V_{mid}$  and the constant  $y$  offset  $C$ .

The recommended fitting method is Levenberg Marquardt.

## Current Time Course (Hodgkin Huxley)

$$f(t) = I'(1 - e^{-t/\tau_j})^a (k_{\infty} - (k_{\infty} - 1)e^{-t/\tau_k})^b + C$$

This is the Hodgkin Huxley model for describing the time course of voltage dependent ionic membrane currents. This equation was originally used to describe voltage activated sodium and potassium currents (with  $a = 3$  and  $b = 1$ ). The term  $k_{\infty}$  is the steady-state inactivation,  $I'$  is the maximum current that is achieved in the absence of inactivation,  $\tau_j$  is the activation time constant,  $\tau_k$  is the inactivation time constant, and the power terms  $a$  and  $b$  are empirically determined.

The fit solves for  $I'$ ,  $\tau_j$ ,  $\tau_k$ ,  $k_{\infty}$  and the constant  $y$  offset  $C$ . The power terms  $a$  and  $b$  must be optionally specified.

The recommended fitting method is Levenberg Marquardt.

## Exponential, Alpha

$$f(t) = \sum_{i=1}^n A_i t e^{-t/\tau_i} + C$$

The alpha exponential function has been used to describe temporal responses at the neuronal soma to synaptic input.

The fit solves for the amplitude  $A$ , the time constant  $\tau$  and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

## Exponential, Cumulative Probability

$$f(t) = \sum_{i=1}^n P_i (1 - e^{-t/\tau_i}) + C$$

This function fits data that have been binned cumulatively. That is, each successive bin contains its own data plus the data in all of the preceding bins.

This function should not be used for binned data because cumulative binning creates artificial correlations between successive bins. The correlation occurs because each successive bin contains all of the data in the preceding bins. The cumulative exponential function provides meaningful results only if the data values are not correlated.

The fit solves for the proportion (amplitude)  $P$ , the time constant  $\tau$  and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

### Exponential, Log Probability

$$f(t) = \sum_{i=1}^n P_i e^{[\ln(t) - \ln(\tau_i)] e^{\ln(t) - \ln(\tau_i)}}$$

*Can only be used with Results or Graph window data.*

*The dwell time data (t) must be input as  $\log_{10}(t)$ .*

This function describes dwell time data, usually from single channel experiments, that have been binned on a logarithmic time scale. Logarithmic binning is often preferable to conventional linear binning because of its superior resolution of widely spaced time constants.

The fit solves for the proportion (amplitude)  $P$ , the time constant  $\tau$  and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is variable metric with maximum likelihood estimation.

### Exponential, Power

$$f(t) = \sum_{i=1}^n A_i (1 - e^{-t/\tau_i})^a + C$$

The fit solves for the amplitude  $A$ , the time constant  $\tau$  and the constant  $y$  offset  $C$  for each component  $i$ . The power term  $a$  must be optionally specified.

The recommended fitting method is Levenberg Marquardt or Chebyshev if fitting only a single term (Chebyshev can solve for a single term only).

### Exponential, Product

$$f(t) = \sum_{i=1}^n A_i (1 - e^{-t/\tau_i})(e^{-t/\tau_{d_i}}) + C$$

This function can be used to fit postsynaptic events (excitatory or inhibitory postsynaptic potentials). The fit solves for the amplitude  $A$ , the rise time constant  $\tau_r$  and the decay time constant  $\tau_d$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

### Exponential, Sloping Baseline

$$f(t) = \sum_{i=1}^n A_i e^{-t/\tau_i} + mt + C$$

This function is used to fit an exponential of the standard form to data that are superimposed on a sloping baseline, for example resulting from a constant baseline drift.

The fit solves for the amplitude  $A$ , the time constant  $\tau$  for each component  $i$ , and the common parameters, the slope  $m$  and constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Chebyshev.

### Exponential, Standard

$$f(t) = \sum_{i=1}^n A_i e^{-t/\tau_i} + C$$

This is the most basic function used to fit changes in current or voltage that are controlled by one or more first order processes. The fit solves for the amplitude  $A$ , the time constant  $\tau$ , and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Chebyshev.

## Exponential, Weighted

$$f(t) = K_0 \left( \sum_{i=1}^n f_i e^{-K_i t} \right) + C$$

This function is identical to the constrained exponential function except that the sum of the  $f_i$  components is not constrained to 1.

The fit solves for the proportion (amplitude)  $f$ , the rate constant  $K$ , the 'weight'  $K_0$  and the constant y offset  $C$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

## Exponential, Weighted/Constrained

$$f(t) = K_0 \left( \sum_{i=1}^n f_i e^{-K_i t} \right) + C \text{ where } \sum_{i=1}^n f_i = 1.0$$

Requires the variable metric fitting method.

This function has been used to describe the recovery rate of ground state absorption following photo excitation of intercalated metal complexes bound to DNA.

The fit solves for the proportion (amplitude)  $f$ , the rate constant  $K$ , the weight  $K_0$  and the constant y offset  $C$  for each component  $i$ . The  $f_i$  terms sum to 1.0.

The fitting method must be variable metric.

## Gaussian

$$f(t) = \sum_{i=1}^n A_i \frac{e^{-(t-\mu_i)^2/2\sigma_i^2}}{\sigma_i \sqrt{2\pi}} + C$$

This is for data that can be described by one or more normal distributions. For  $n$  components, the fit solves for the amplitude  $A$ , the Gaussian mean  $\mu$ , the Gaussian standard deviation  $\sigma$  and the constant y-offset  $C$  for each component  $i$ .

This function is generally used for describing amplitude distributions of single channel events (Heinemann, 1995).

The recommended fitting method is Levenberg Marquardt.

## Goldman Hodgkin Katz

$$f(x,y,z) = \frac{RT}{F} \ln \frac{[X]_1 + \alpha[Y]_1 + \beta[Z]_1}{[X]_2 + \alpha[Y]_2 + \beta[Z]_2}$$

$$\alpha = pY/pX$$

$$\beta = pZ/pX$$

This function can only be used with Results window data.

This function is used to describe the steady state dependence of membrane voltage on ion concentrations and the relative permeability of those ions through the membrane.

The equation assumes that all the ions are monovalent. For positive ions  $[X]_1$  refers to the concentration outside the membrane and  $[X]_2$  refers to the intracellular concentration. For negative ions  $[X]_2$  refers to the concentration outside the membrane and  $[X]_1$  refers to the intracellular concentration.

The fit solves for the permeability ratios  $\alpha$  and  $\beta$ .

The recommended fitting method is Levenberg Marquardt.

### Hill (4 Parameter Logistic)

$$f(x) = I_{min} + \frac{I_{max} - I_{min}}{1 + (C_{50}/[x])^h}$$

This is a modified form of the Hill equation that is useful for fitting doseresponse curves. The half maximal concentration is determined directly.  $I_{min}$  refers to the baseline response,  $I_{max}$  refers to the maximum response obtainable with drug  $x$ ,  $C_{50}$  is the concentration at half - maximal response (inhibitory or excitatory) and  $h$  is the Hill slope.

The fit solves for  $I_{min}$ ,  $I_{max}$ ,  $C_{50}$  and  $h$ .

The recommended fitting method is Simplex.

### Hill, Langmuir

$$f(x) = \sum_{i=1}^n \frac{I_{max_i} [x]^{h_i}}{C_{50_i}^{h_i} + [x]^{h_i}} + C$$

The Langmuir Hill equation allows for fitting a sum of Langmuir components. It is useful for fitting data where non specific binding of the agonist is linear.  $I_{max}$  refers to the maximum response obtainable with drug  $x$ ,  $C_{50}$  is the concentration at half maximal response (inhibitory or excitatory) and  $h$  is the Hill slope.  $C$  is a constant  $y$  offset.

The fit solves for  $I_{max}$ ,  $C_{50}$ ,  $h$  and  $C$ .

The recommended fitting method is Simplex.

### Hill, Steady State

$$f(S) = \frac{V_{max} [S]^n}{K^n + [S]^n} + C$$

This is a general equation that can be applied to many kinds of concentration dependent pharmacological or ion channel responses.  $V_{max}$  refers to the maximum response obtainable with drug  $S$ . By definition a partial agonist will have a  $V_{max}$  value that is less than the  $V_{max}$  of a full agonist.  $K$  is indicative of potency but it is not equal to the concentration at half maximal velocity except when  $n = 1$ . The value of  $n$  places some limitations on the degree of cooperativity of the liganddependent processes. In order to define a concentration-dependent inhibitory process,  $n$  can be seeded with a negative value.

The fit solves for  $V_{max}$ ,  $K$  and  $n$ .

The recommended fitting method is Simplex.

### Lineweaver Burk

$$f(S) = \frac{K_m}{V_{max} [S]} + \frac{1}{V_{max}} \text{ where } S = 1/x$$

This equation, derived by taking the reciprocal of both sides of the Michaelis Menten equation, describes a straight line with a slope of  $K_m/V_{max}$  and a  $y$  intercept of  $1/V_{max}$ . It is useful for gaining information on enzyme inhibition.

The fit solves for  $K_m$  and  $V_{max}$ .

The recommended fitting method is Levenberg Marquardt.

## Logistic Growth

$$f(x) = \frac{R_{max}}{1 + Ae^{-Bx}} + C$$

This function describes an exponential growth that is subject to saturation that occurs at the limiting value  $R_{max}$ . The parameter  $A$  is the number of times the initial value must grow to reach  $R_{max}$ , and  $B$  determines the rate and direction of growth. The function will increase when  $B$  is positive and decrease when  $B$  is negative.

The fit solves for  $R_{max}$  and  $A$ ,  $B$  and the constant  $y$  offset  $C$ .

The recommended fitting method is Levenberg Marquardt.

## Lorentzian Distribution

$$f(x) = \sum_{i=1}^n \frac{2A\omega}{4\pi(x-\mu)^2 + \omega^2} + C$$

The Lorentzian distribution function is generally used to characterize energy transition spectra that exhibit homogenous broadening of the peak. For example, the natural line shape in spectroscopy can be characterized by this function. The peak of a spectral line is narrow but broadens as a result of uncertainties in the energy level of the excited state. It, nevertheless, retains the Lorentzian line shape.

The fit solves for the area  $A$  under the curve, the half width  $\omega$  of the peak, the X axis center  $\mu$  of the peak (generally the center frequency) and the constant  $y$  offset  $C$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

## Lorentzian Power 1

$$S(f) = \sum_{i=1}^n \frac{S(0)_i}{1 + (f/f_c)^2}$$

The power spectra of current fluctuations can be described by the sum of one or more Lorentzian functions of this form.

The time constant is related to the cutoff frequency,  $f_c$ , at which the power has declined to  $S(0) / 2$  by

$$\tau = \frac{1}{2\pi f_c}$$

The fit solves for  $S(0)$  and  $f_c$  for each component  $i$ .

The recommended fitting method is Levenberg Marquardt.

## Lorentzian Power 2

$$S(f) = \sum_{i=1}^n \frac{S(0)_i}{1 + (2\pi f\tau_i)^2}$$

The power spectra of current fluctuations produced by ion channels can be described by the sum of one or more Lorentzian functions of this form where:

If the probability of the channel being open is low relative to the probability of the channel being closed then the channel closed time constant  $\tau_c$  can be ignored and  $\tau$  can be equated to the channel open time constant  $\tau_o$ . At low frequencies the function tends toward  $S(0)$ . At high frequencies the spectrum decays in proportion to  $f^{-2}$ .

The fit solves for  $S(0)$ ,  $\tau$  and the constant  $y$  offset  $C$  for each component  $i$ .

The parameter  $\tau$  has units of s if the frequency,  $f$ , is in Hz.

The recommended fitting method is Levenberg Marquardt.

## Michaelis Menten

$$f(S) = \frac{V_{max}[S]}{[S]+K_m} + C$$

This is a general equation to describe enzyme kinetics where  $[S]$  refers to the concentration of substrate. In this equation,  $V_{max}$  refers to rate of catalysis when the concentration of substrate is saturating.  $K_m$  refers to the Michaelis constant  $((k_{-1} + k_2)/k_1)$  where  $k_1$  and  $k_{-1}$  are forward and backward binding rates of the enzyme substrate complex, respectively, and  $k_2$  is the first order rate constant for the formation of product from the bound enzyme substrate complex.

The fit solves for  $V_{max}$ ,  $K_m$  the constant  $y$  offset  $C$ .

## Nernst

$$V(x) = \frac{RT}{zF} \ln \frac{[x]_1}{[x]_2}$$

This function describes the condition where an equilibrium exists between the energy associated with membrane voltage ( $V$ ) and the energy associated with a concentration gradient of an ion species  $x$ . Hence, the Nernst potential for a given ion is often also referred to as the equilibrium potential for that ion.

The fit solves for the concentration  $[x]_2$  given a series of concentrations  $[x]_1$ .

## Parabola, Standard

$$f(x) = Ax^2 + Bx + C$$

This is the parabolic function.

The fit solves for the parameters  $A$ ,  $B$  and the constant  $y$  offset  $C$ .

## Parabola, Variance Mean

$$f(x) = ix - x^2/N$$

This is a form of the parabolic function used to describe synaptic event data where  $i$  is the unitary synaptic current amplitude and  $N$  is the number of release sites.

The fit solves for the parameters  $i$ , and  $N$ .

## Poisson

$$f(x) = \frac{e^{-\lambda} \lambda^x}{x!}$$

*Requires an integral X axis.*

*Requires normalized data.*

*Maximum number of points = 170.*

The data will be automatically zero shifted.

The Poisson distribution describes the probability of getting  $x$  successes with an expected, or average number of successes denoted by  $\lambda$ . This is similar to the binomial function but is limited to cases where the number of observations is relatively small.

This function requires integer values of  $x$ . If the X axis values are not integers then they are converted to such prior to, and rescaled following, fitting. The ordinate data are also normalized so that they range from 0 to 1, and are rescaled after fitting so that the fitted curve conforms to the original data. Rescaling is such that the area under the fitted curve is equal to the area under the original data curve.

The number of sample points for this function is limited to 170 to conform to the computational limit for a factorial.

The fit solves for the  $\lambda$  given a series of observed probability values  $x$ . Since the X axis scale is integral the fitted curve will appear as a histogram.

The recommended fitting method is Levenberg Marquardt.

## Polynomial

$$f(x) = \sum_{i=0}^n a_i x^i$$

The fit solves for the polynomial coefficients  $a_i$ . The term  $a_0$  always exists in the solution. A first - order (or 'one term') polynomial is, therefore, given by  $f(x) = a_0 x^0 + a_1 x^1 = a_0 + a_1 x$ , which is a straight line fit.

The maximum order is 6.

## Straight Line, Origin at Zero

$$f(x) = ix$$

This function is used to fit variance - mean data (V - M analysis) to estimate the unitary current,  $i$ . The fit solves for the slope  $i$ , forcing the origin to zero.

## Straight Line, Standard

$$f(x) = mx + b$$

The straight line fit solves for the slope  $m$  and the  $y$  intercept  $b$ .

## Voltage Dependent Relaxation

$$f(V) = \frac{1}{a_0 e^{V/\alpha} + b_0 e^{-V/\beta}} + C$$

This function describes the relaxation kinetics for a two state voltage dependent process. The forward and reverse rate constants are  $\alpha$  and  $\beta$ , respectively. The term  $a_0$  is value of  $\alpha$  at  $V=0$  and  $b_0$  is value of  $\beta$  at  $V=0$ .

The fit solves for  $\alpha$ ,  $\beta$ ,  $a_0$ ,  $b_0$  and the constant  $y$  offset  $C$ .

The recommended fitting method is Levenberg Marquardt.

## Constants

$F$  = Faraday constant =  $9.648456 \times 10^4$  C/mol.

$R$  = gas (Rydberg) constant = 8.31441 J/mol deg K.



# Glossary

## A

### ABF

Axon Binary File, the standard file format for Axon data files. The trials from Clampex software are in ABF format.

### Acquisition Mode

The type of acquisition defined in the protocol. Options are: gap free, variable length events, fixed length events, high speed oscilloscope, and episodic stimulation. Clampex software has only episodic stimulation and high speed oscilloscope, which it terms 'internally triggered' and 'compound triggered' respectively.

### Amplifier Mode

Voltage clamp or current clamp.

## D

### Data Point

Measure of analysis throughput. One data point is the single result value for a test compound at a given concentration. Averaged result values from replicates give one data point. Values for the same compound at different concentrations are different data points. Thus, a 5 point EC50 analysis is five data points, regardless of the number of replicates run to create it.

## E

### Episodic Stimulation Mode

In Clampex software, called 'internally triggered' mode. Noncontinuous acquisition where each sweep occurs at a predefined interval (set in Edit Protocol dialog). A stimulus can be delivered during the sweep (see high speed oscilloscope mode).

### Epoch

A part of a sweep, used for configuration of the stimulus waveform. Clampex software has 50 epochs per sweep.

## G

### GUID

Global Unique Identifier—a long random sequence of characters and numerals used to name files imported into the Clampfit Batch Analysis database.

## I

### Internally Triggered Acquisition Mode

Term used in Clampex software for episodic stimulation mode (see glossary entry).

## P

### P/N Leak Subtraction

Software subtraction of the transient current that charges  $C_m$  and the steady state current through  $R_m$ . Both of these are passive currents. By subtracting these passive currents the researcher can clearly observe ionic currents.

### Pre sweep Train

A set of regular, rectangular pulses that can be delivered prior to each sweep in a trial, to stimulate the cell prior to further stimulation and recording in the main sweep.

### Protocol

Set of definition parameters for data acquisition in trials. Protocols determine sampling rate and filtering, trial timing, command waveform, P/N leak subtraction and pre sweep trains, and on line measurements.

## R

### Run

Subcomponent of a trial, between sweep and trial. The corresponding sweeps of each run are averaged, so the ABF file of the trial has only as many sweeps as were configured for one run, even though actual acquisition might have consisted of several runs.

## S

### Sample

The datum produced by one A/D (Analog to Digital) or D/A (Digital to Analog) conversion. Prior to acquisition in trial definition, you must set the rate at which samples are output and acquired.

### Signal

Data streams from different sources, sharing the same time base, within a trial. These are displayed in horizontal subpanes within ABF trial files. In Clampex software there is only ever one signal recorded per channel, unless P/N leak subtraction is enabled, in which case the raw and the corrected data are both recorded, one in each signal.

### Sweep

A continuous sequence of data acquisition within a trial. Sweeps can be accompanied with delivery of a command waveform. In trials with multiple runs, corresponding sweeps in the different runs are averaged.

**T****Trace**

A sequence of data recorded in a trial, from one sweep in one signal.

**Trial**

A data acquisition, often consisting of multiple sweeps, and possibly accompanied with delivery of a stimulus waveform. Trials are defined in a protocol, and recorded in ABF files.

**W****Waveform**

The voltage command stimulus delivered during the acquisition sweep, configured in the protocol.

## Contact Us

Phone: [+1-800-635-5577](tel:+1-800-635-5577)  
Web: [moleculardevices.com](http://moleculardevices.com)  
Email: [info@moldev.com](mailto:info@moldev.com)

Visit our website for a current listing of worldwide distributors.