StdpC 7.1

(Spike timing dependent plasticity Clamp),

in parts based on **DYNCLAMP2** [Pinto et al., 2001]

Manual

Thomas Nowotny

Centre for Computational Neuroscience and Robotics,
School of Engineering and Informatics,
University of Sussex,
Brighton BN1 9QJ, UK
E-mail: t.nowotny@sussex.ac.uk

Felix B. Kern

School of Life Sciences,
University of Sussex,
Brighton BN1 9QG, UK

Original DYNCLAMP2 developers:

Reynaldo Daniel Pinto,
Robert C. Elson,
Attila Szücs,
M. I. Rabinovich,
A. I. Selverston,
H. D. I. Abarbanel

Institute for Nonlinear Science,
University of California, San Diego,
9500 Gilman Dr.
Mail Code 0402
La Jolla, CA 92093-0402, USA

StdpC is software building on the ideas that led to the development of the earlier DYNCLAMP2 software. The new software is based on Qt and comes in two versions, one for the old Digi-Data 1200/A interface, the other supporting all National Instruments devices that support NI's NIDAQmx API.

Some functionality in this software has not been tested thoroughly. Any feedback on problems with the software or potential bugs is greatly appreciated. Older versions of the software (StdpC) are described in Journal of Neuroscience Methods [Nowotny et al., 2006] and, more recently, in Nature Protocols [Kemenes et al., 2011]. *If you use StdpC and publish results based on its use, please cite the website and these papers.* Please contact us for any further questions (t.nowotny@sussex.ac.uk).

1 Introduction

The Dynamic Clamp protocol, developed by [Sharp et al., 1993] and independently by [Robinson and Kawai, 1993] allows inserting simulated membrane conductances into cells, and/or adding synapses between cells as well as a variety of other manipulations. Here, we describe StdpC 2019, a software for controlling biological neurons and an internal spike generator in (soft) real time.

The software allows its user to define artificial synapses, Hodgkin-Huxley conductances [Hodgkin et al., 1949], simulated neurons built from such conductances, and presynaptic spike generators. The parameters of these simulated entities can be controlled on the graphical user interface (GUI) or through script files.

StdpC has a built in electrode compensation method (Active Electrode Compensation [Brette et al., 2008]), which is optionally usable after going through a short and automatic calibration procedure. This compensation technique allows for application of a single electrode both for stimulation and recording with higher signal fidelity than conventional compensation techniques, like bridge balancing or discontinuous injection/recording.

During the experiment, the acquired data can be saved continuously into a data file either in ASCII or binary format for off-line post-processing and results analysis.

The software supports the DIGIDATA 1200A data acquisition card of Axon Instruments, Inc (now part of MDS Analytical Technologies), all NIDAQmx capable devices of National Instruments, and a pure simulation mode for testing. The program was written in C++ (using QtCreator). The graphical user interface is based on Qt libraries. The program has been developed and tested on Windows 7 but should also be compatible with newer versions up to Windows 10. StdpC may or may not work on older versions of Windows, subject particularly to the DIGIDATA and NIDAQmx drivers. Newer DIGIDATA versions are currently not supported; please contact us if you wish to change this.

2 Hardware and software requirements

A standard PC, Windows 7-10 (older versions of Windows may work, but are untested). A DIGIDATA 1200A ADC/DAC board from Axon Instruments, or a National Instruments DAQ and installed NIDAQmx driver.

The program was tested on a PC with Windows 7 Professional and a National Instruments PCIe-6251 DAQ using NIDAQmx for Windows, version 18.5.

3 Asynchronous Dynamic Clamp Cycle

The dynamic clamp protocol consists of a cycle of reading the membrane potential of the cells, calculating the current to be injected into the cells according to the synapses and conductances that are to be simulated, and generating the voltage commands that will generate the currents. StdpC aims at repeating this cycle at the maximal rate supported by the hardware in order to simulate continuous biological processes. Since Windows is not a real time operating system, the actual time from one cycle to the next can not be controlled in a strict way. Instead of aiming at such a control StdpC is based on an asynchronous dynamic clamp cycle. Every time the program updates the membrane potential of the cells it also reads the system clock, which allows it to calculate the duration of the previous Dynamic Clamp cycle. The measured time intervals between voltage updates are used in the computation of the currents. This limits the impact of unequal delays during repeated cycles unless these delays become large compared to the biological time scales in the system studied. As a side effect of the asynchronous Dynamic Clamp cycle, StdpC update cycles will always run at the maximum rate supported by the hardware. With the typical

successive upgrade of computer hardware the quality of the Dynamic Clamp interaction will improve over time.

4 How to install the software

4.1 Precompiled executables

Download the exe file to a convenient location, e.g., C:\Program Files\StdpC.exe and execute.

4.2 Source package

Download the source package and unzip it to a convenient location, or check it out via git. To build StdpC from source, you will need Qt 5 and either the msvc toolchain (recommended) or mingw (if you require Digidata 1200/A support). We recommend compiling from within QtCreator, but you may find it more convenient to invoke qmake.exe from a shell instead. You can control driver support by adding "nidaqmx", "nonidaqmx" or "digidata" to the CONFIG environment variable (e.g. "qmake.exe [...] CONFIG+=nidaqmx").

5 Troubleshooting

5.1 I/O conflicts

In the past, there have been problems with I/O conflicts between the DigiData board and other hardware devices. If you are experiencing problems, you may need to change the base address of the DigiData board. Hardware conflicts can occur even if the DIGIDATA board works fine with Axon programs like Axoscope because, unlike these, StdpC 2017 does read the Real Time Counter (RTC) in the Digidata board. Usually, the DigiData board is configured to use the I/O base address 0x320, and in this case the RTC reading ports are in the range of 0x330 - 0x332. Many computers use the address 0x330 for some devices like sound cards, etc.

To check for and resolve a hardware address conflict, *Windows NT, 2000, XP:*
Open: Start → Programs → Administrative Tools → Windows NT Diagnostics. Choose the folder ⟨Resources⟩, click in ⟨I/O Port⟩ and use the cursor to browse the list of used ports. You should look for 330. Note that 320 will not appear because the Digidata Board is not detected by Windows. If you find any reference in the range 330 - 33F and your Digidata board is configured to use the base address 320 you need to change I/O address of your Digidata board to run StdpC and avoid other problems due to the I/O conflict.
Windows 95, 98:
Open: Start → Settings → Control Panel → System. Choose the folder ⟨Device Manager⟩, click ⟨I/O Port⟩ and use the cursor to browse the list of used ports. You should look for 330. Note that 320 will not appear because the Digidata Board is not detected by Windows. If you find any reference in the range 330 - 33F and your Digidata board is configured to use the base address 320 you need to change the I/O address of your Digidata board to run StdpC and avoid other problems due to the I/O conflict.
Changing the base address of the Digidata Board and in the StdpC program:
Identify a free range of port addresses. If 0x330 is in use, addresses around 340-370 are usually available which is sufficient for the Digidata ports. Reconfigure the Digidata board for the base address 340 or 350 manually (the RTC will be 350 or 360 and no conflicts!) and run the StdpC program. The Digidata base address can be changed within StdpC on the Digidata settings dialog.

6 Changes from previous versions/programs

6.1 Changes DYNCLAMP2 to StdpC version 3

The following is a list of features which were new in StdpC v3:

- *Spike generator*
  A spike generator can replace a biological presynaptic neuron. Spikes are generated either periodically in a fixed pattern or from a file containing predefined spike times. This feature is actually an original feature of R. Pinto but was not published yet.
- *Hidden parameter panels*
  To de-clutter the screen of the dynamic clamp computer, in StdpC the parameter dialogs for chemical synapses and Hodgkin-Huxley conductances have been moved to separate pop-up windows.
- *Data displays for debugging*
  To allow for easy debugging of wiring and gain problems the software now has two data displays that can show input and/ or output channels or functions thereof (averages, spikes detected).
- *Save and load parameter settings*
  You can now save and load the current parameter settings of the clamp. This way one does not have to redo parameter settings every time the StdpC program is restarted. The settings are stored in a simple ASCII format that allows editing by hand or snooping around for curiosity.
- *Spike timing dependent plasticity*
  The two chemical synapses can now be made plastic obeying a Spike Timing Dependent Plasticity protocol. There are two different protocols to choose from and a variety of parameters determining the details of the learning mechanism
- *Experimental automatization and scripting*
  The StdpC supports a simple form of experimental protocol automation (scripting). The user can specify a script file that contains events at given times. These events include switching on and off of synaptic connections or Hodgkin-Huxley type conductances as well as arbitrary parameter changes. The script is loaded on starting the clamping process and executes the commands at the given times after clamping started.

6.2 Changes from StdpC version 3 to version 4

- Since this version of StdpC, all synapses are freely reconfigurable as chemical or electrical synapses.
- the presynaptic and postsynaptic neuron of each synapse is freely reconfigurable. In particular it is now possible to combine synapses between two biological neurons (on channels IN0/OUT0 and IN1/OUT1) with synapses from the spike generator (channel SPG) to either of the neurons. This gives more flexibility in the experimental design
- Each chemical synapse can now been chosen individually to be plastic or not.
- The initial value for a plastic synapse is given by its individual G_s setting, not by a global initialisation value in the plasticity block. The displays for the synaptic strength have been removed because they turned out to be disruptive for the clamping performance.
- Some smaller adjustments aiming at better user interaction and performance of the data displays. We, however, still recommend not to use the data displays during real experiments. They are for debugging purposes only and degrade clamping performance considerably when used.

6.3 Changes from StdpC v4 to StdpC 2007 (v5)

- The user interface is now based on QT
- The driver for the DigiData 1200 board was re-created using (an extended version of) the free PortTalk package.
- StdpC 2007 supports all NIDAQmx capable National Instruments boards
- The source code has been re-organised to allow for rapid inclusion of additional hardware support.
- StdpC 2007 allows to use all input and output channels available on the acquisition hardware.
- All synaptic properties including the details of the synaptic plasticity are now individually adjustable for every synapse
- There are now two alternative descriptions for Hodgkin-Huxley type conductances, including several functional forms for the activation and inactivation curves.
- With the additional support for other hardware, there is now more fine-grained control over the channels that are used for voltage and current conversion. All channels are available in "input channels" and "output channels" dialogs. Only channels activated in these dialogs will appear in the drop-down menus within synapses and HH conductances.

6.4 Changes from StdpC 2007 to StdpC 2010 (v6.0)

- Active Electrode Compensation feature included as an optional, automatic, digital electrode compensation technique calculated within the software.
- A new feature allows for the measurement of the resistance and capacitance properties of both the electrode and cell membrane, conveniently controlled from the software.
- Full experimental data saving support has been included. Any combination of the active input and output channels, along with the spike generator, can be saved, with a frequency (quasi) independent of the main dynamic clamp loop frequency. Data can be saved either in ASCII, or in binary format.
- A constant bias current can now be defined for any active output channel for special experiment configuration needs.

6.5 Changes from StdpC 2010 to StdpC 2011 (v6.1)

- Active electrode compensation is now stable.
- Through scripting, a sample-and-hold mechanism can be envoked if measurement artefacts due to the experimental protocol need to be removed from voltage signals. For example, the monitoring of membrane potentials can be interrupted during stimulation from sources external to the dynamic clamp.
- The synapse models have been updated with the addition of the "Destexhe synapse" and a modification of the alph-beta synapse.
- A variety of minor bugs and issues have been fixed.
- Data saving can now be enabled and disabled globally in teh data saving dialog. This overrides the choices for individual channels, and if disabled, the overheads of this mechanism can be avoided.

6.6 Changes from StdpC 2012 to StdpC 2017 (v7.0)

- m/h/tau HH currents received two new tau formulations.
- AEC computed currents can now be directed to a separate channel for data output purposes in addition to its usual function. In addition, there is no longer any restriction on the number of AEC channels.
- It is now possible to use several data sources (DAQs, files, spike generators, models) simultaneously.
- There is now an option to simulate entire (single-compartment) Hodgkin-Huxley neurons, see section 7.4.6 for details.
- It is now possible to define an arbitrary number of HH conductances and synapses.
- Currents and synapses can now be assigned to more than one set of channels. Each assignment draws on the common parameter set, but is functionally independent from other assignments. As a consequence of this update, the channel assignments in protocols and scripts written for/by older StdpC versions may be wrongly converted. Please check for correctness and/or update your scripts and protocols, see section 8 for details.
- Synapses (but not gap junctions) now support conduction delays.
- Currents, synapses, channel assignments, neuron models and model instances can now all be turned on or off by script while the clamp cycle is running, provided that they were active when the clamp cycle was started.
- Graph performance and handling has been improved.
- A performance indicator has been added to the main window showing the actual clamp cycle frequency.
- Large parts of the code have been cleaned up and refactored/rewritten into a more modular design.

6.7 Changes from StdpC 2017 to StdpC 7.1

- The clamp thread's priority can now be set by the user to trade off between maximum performance and the threat of potentially freezing one's system.
- The clamp cycle can now be started after a settling period, or by way of a digital trigger input.
- A new data source was added, Voltage Stepper, which produces square steps.
- Several tools were added, including a generator pseudo-random synaptic background noise, a "wire" facility to pass data from input to output channels, and an experimental software-defined voltage clamp using PID control.
- The chemical synapse model can now produce stochastically sized post-synaptic potentials.
- Synaptic gSyn can now be multiplied with samples from an input channel, e.g. from SimulDAQ input.
- Data saving has become much more flexible, both in terms of output format and in terms of the naming of the output file(s).
- The GUI has received numerous upgrades and improvements to usability.
- We have added MSVC build capability. The MSVC build runs faster, but does not support the Digidata driver. For Digidata 1200/A users, a mingw build is still provided.
- A number of bugs have been fixed.

7 How to use StdpC

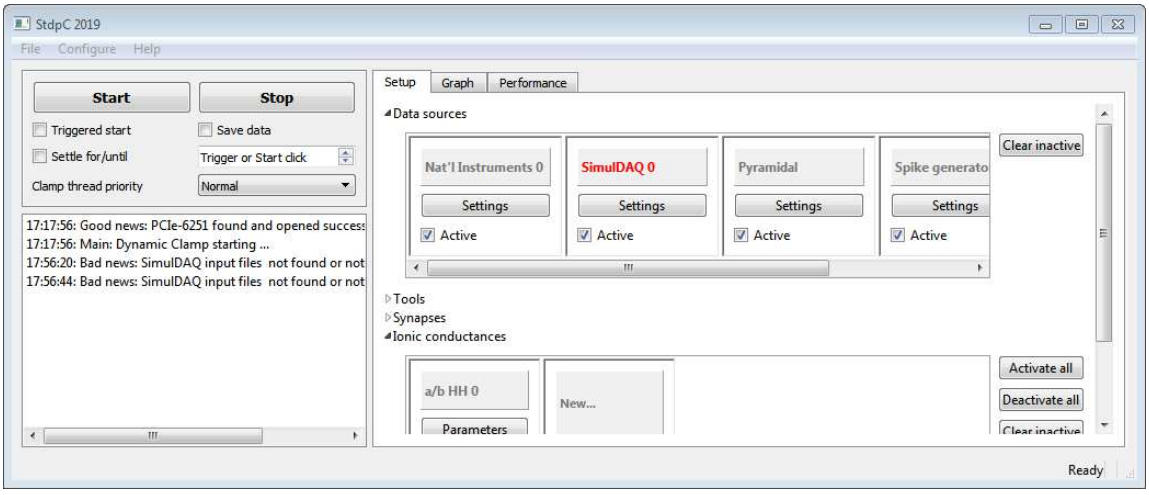
7.1 "Connecting" the program to the cells

To avoid the generation of wrong current commands due to artifacts introduced into the membrane measurement by the current injection, one must either (1) insert of two electrodes into each cell (one for injection, the other one for recording), or (2) use a careful compensation of a single electrode in order to recover the actual membrane potential from the raw artefactual voltage. For the latter solution StdpC has a built-in artifact compensation method, AEC (Active Electrode Compensation [Brette et al., 2008]), that allows for a convenient, software-based electrode compensation. One can think of special experiment configurations, like simultaneously recording in multiple machines, in which AEC might not be found appropriate for compensation. In these cases any external analogue technique like microelectrode amplifier features could be applied instead of (and NOT in conjunction with) AEC. However, we generally encourage the application of AEC, as it is an automatic method that provides high quality signals at arbitrarily high time resolution. See section 9 for more detail about electrode compensation.

The membrane potential output of the microelectrode amplifiers should be connected to the analog input channels chosen to be scanned in the "input channels" dialog. The command voltage for the injection of current into the cells will be applied to the analog output channels chosen in the corresponding drop-down boxes. For specific channels to appear in these boxes, they need to be activated in the "output channels" dialog, and the data source they belong to must be activated.

The currents are calculated individually for each active conductance and synapse and then are summed for each cell to generate the corresponding voltage command for the microelectrode amplifiers. It is recommended to turn on the dynamic clamp cycle and to monitor the current command voltage in an oscilloscope before turning on the injection of the current in the microelectrode amplifier, to see if it looks like what is expected. If it is orders of magnitude wrong, check the conversion factors in the "input channels" and "output channels" dialogs. Alternatively, the channel setup settings can be checked and corrected safely on a model cell, which are usually shipped with the microelectrode amplifiers.

7.2 Control Panel of the Program



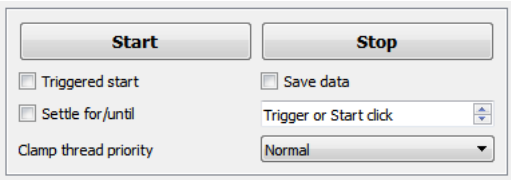
The main control panel contains different blocks of control for the different functions of the software – setting up and configuring acquisition devices and neuron models (data source con-

trol block), adding and configuring miscellaneous tools (tools block), simulating synapses (synapse control block), and simulating ionic conductances (Hodgkin-Huxley (HH) type conductances; conductance control block). The control blocks are front-ends to more detailed control dialogs. The "Graph" and "Performance" tabs serve to display channel and conductance data while the clamp cycle is running, or data about the clamp cycle performance, respectively. There is a message window in the left part of the panel that shows system messages. These messages can be saved for future reference. They are also automatically written to a local file "StdpC_last.log".

7.3 Configuration of the Hardware and Control of the Program

Once StdpC is started, the last known hardware configuration is loaded (from a local file named "StdpC.conf"). Other parameters are initialized with default values that are pre-compiled into the software. The message window will show a message whether the chosen hardware was successfully initialized. Failing hardware initialization can have several causes:

1. If using the DigiData 1200(A):
   StdpC's default address for the board is 0x320 (hexadecimal), which is the default address of the board, but the specific board can have been configured to use another range of addresses. Possible values are 0x340, 0x360, 0x380, 0x3A0, 0x3C0, etc. Make sure that the correct address is entered in the "Digidata" dialog.
2. If using a National Instruments board:
   Make sure that NIDAQmx is installed and the board is working properly (using the NI Automation explorer). Make sure your device is in the list of devices that support NIDAQmx (as opposed to the older devices supporting "old style" NIDAQ / NIDAQ legacy only).
3. I you are running with simulated data acquisition (SimulDAQ):
   Make sure that you have a correctly formatted input file for the membrane potential of cells and that the file name in the "SimulDAQ" dialog points to the right location of this file. Also make sure that the location the output file name points to is writeable to you.



The remaining controls are the "start" and the "stop" button. The "start" button initiates the start of dynamic clamp cycles. In particular, pressing this button will stop a running dynamic clamp thread, load the settings from all dialogs into memory, and start the dynamic clamp thread again with the new settings. The dynamic clamp thread will then run until stopped by pressing the "stop" button.

The following additional controls are found in this area:

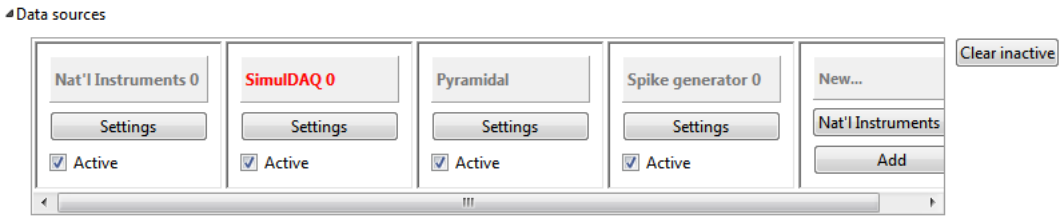
- "Triggered start": When checked, pressing "Start" will prime the clamp cycle to wait until a signal on a digital input channel is received before engaging the cycle. The trigger channel is set in the *Start trigger* dialog in the *Configure* menu.
- "Settle for/until:" When checked, pressing "Start" will start the clamp cycle in a reduced mode, in which only tools and conductances selected to be active "when settling" are enabled. The settling period can be specified in seconds or set to 0, meaning indefinite settling. When the settling period is over, "Start" is pressed again, or a trigger input is applied, the full clamp cycle is immediately engaged.
- "Save data": Enable or disable data saving, see section 10.
- "Clamp thread priority": Set the priority of the thread running the clamp cycle. We recommend using the default "Highest" priority, as "Time-critical" may be unstable and cause freezing. Depending on your system, other settings may be preferable, however.

The data acquisition hardware can be chosen in the data source control block in the setup tab. The specific hardware settings can be adjusted in the "Settings" dialog of each added device.

Other general control elements can be found in the menus of the main menu bar.

- "File" menu:
  - "Load Protocol": Load parameter settings from a previous session. The standard file name extension for these is "cpr", albeit the settings are saved in plain ASCII format.
  - "Save Protocol": Save the current parameter settings into a file for later use or documentation.
  - "Load Script": Load a script for experiment automation. Scripting is described in section 8.
  - "Unload Script": Remove a script from memory and use StdpC interactively.
  - "Export Log": Export the contents of the message box to a file.
  - "Clear Log": Remove the message box contents.
  - "Exit": Quit StdpC.
- "Config" menu:
  - ' "Start trigger" lets you select a digital input channel from your National Instruments board for triggered clamp starting.
  - "Data saving" Shows a dialog to configure data saving settings, see section 10.
- "Help": Offers a brief description about StdpC.

7.4 Data sources



The data sources control block is used to configure all acquisition devices, recorded inputs, spike generators and Hodgkin-Huxley models. Sources are added by selecting the appropriate type in the rightmost box in the list, labelled "New...", and clicking "Add". An initially inactive data source is then added to the list, which can be configured by clicking the "Settings" button and activated by checking the "Active" checkbox. If the configuration fails, the failed data source's title will be highlighted in red. When a data source is no longer required, it can be removed by deactivating it and clicking the "Clear inactive" button to the right of the list.

7.4.1 Digidata 1200(A)

The Digidata 1200/1200A settings are limited to device address, which is given in hexadecimal. Channels are configured in subordinate dialogs (see below).

Scripting: Script access is given through the `DigiDatap[#]` variable. Only input/output channels are available for scripting, see below.

7.4.2 National Instruments

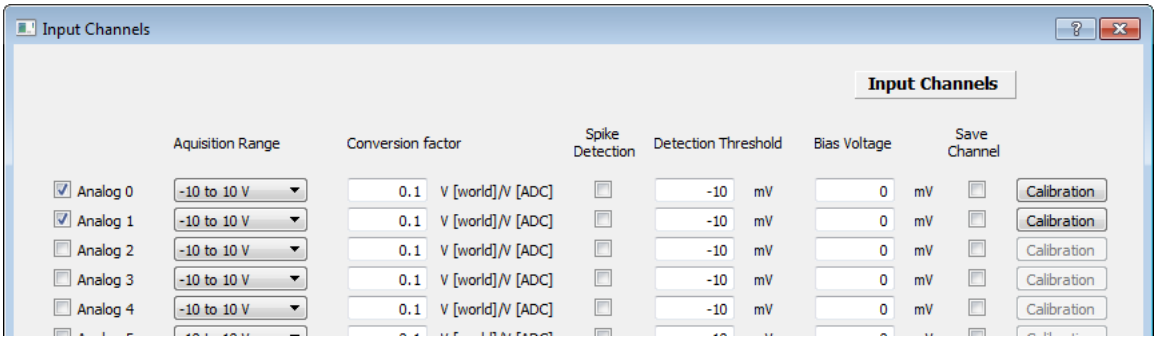
National Instruments devices are configured by giving the exact name of the device, usually "Dev1" or similar. This can be found e.g. by running the NI MAX utility that comes with the device drivers. Channels are configured in subordinate dialogs (see below).
Scripting: Script access is given through the `NIDAQp[#]` variable. Only input/output channels are available for scripting, see below.

7.4.3 SimulDAQ

For testing and debugging, you can use a text file with recorded voltages in place of an acquisition device. Each line corresponds to a set of data points: The first column gives the time, while subsequent columns (separated by any non-numeric character(s)) give voltage values. In the SimulDAQ settings dialog, configure the number of input channels to reflect the data - note that all data are read, and a mismatch between the settings and the data will most likely result in garbage. SimulDAQ output provides an alternative to normal data saving. Both input and output channels are configured in subordinate dialogs (see below).
Scripting: Script access is given through the `SDAQp[#]` variable. Only input/output channels are available for scripting, see below.

7.4.4 Input channel configuration

The *Input Channels* dialog contains the basic configurable properties for the input channels, like whether they are going to be active or not, their acquisition range and conversion factor, along with some more StdpC specific properties, like whether spike detection is going to be on or off on them (required for spike-time dependent plasticity rules in synapses), and if it is on, what spike detection threshold will be applied, and finally whether the channel is going to be saved or not during dynamic clamping (Section 10). Always make sure that the right conversion factor is applied for each active channel, and a sufficiently wide acquisition range is set in order to prevent out of range data being cut off of the recorded data.
Input channels of an active, fully configured acquisition device can be AEC-calibrated from here by clicking "Calibrate". For more details about AEC and the calibration process, see section 9.

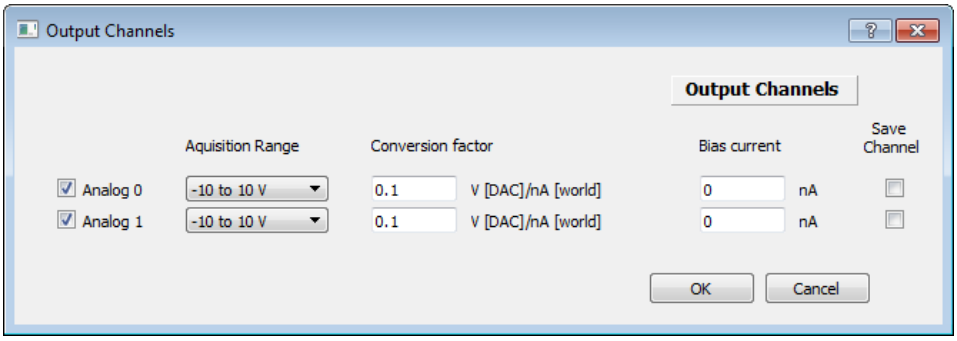


Scripting: Script access to input channels is given through the `<device>.inChn[#]` variable (e.g. `NIDAQp[0].inChn[0].active`). The following members are available for scripting:

<device>.inChn[#].␣	Value range	Notes
`active`	0,1	Channels must be initially active. Deactivating a channel suspends all synapses and conductances that rely on it.
`spkDetect`	0,1	Spike detection on/off
`spkDetectThreshold`	double	
`bias`	double	A constant bias added to every sample

7.4.5 Output channel configuration

The *Output Channels* dialog contains the basic configurable properties for the output channels, like whether they are going to be active or not, their acquisition range and conversion factor, what is the bias current level applied on that channel if any, and finally whether the channel is going to be saved or not during dynamic clamping (Section 10). Always make sure that the right conversion factor is applied for each active channel, and a sufficiently wide acquisition range is set in order to prevent out of range values being cut off of the injected current.



Scripting: Script access to output channels is given through the `<device>.outChn[#]` variable (e.g. `NIDAQp[0].outChn[0].active`). The following members are available for scripting:

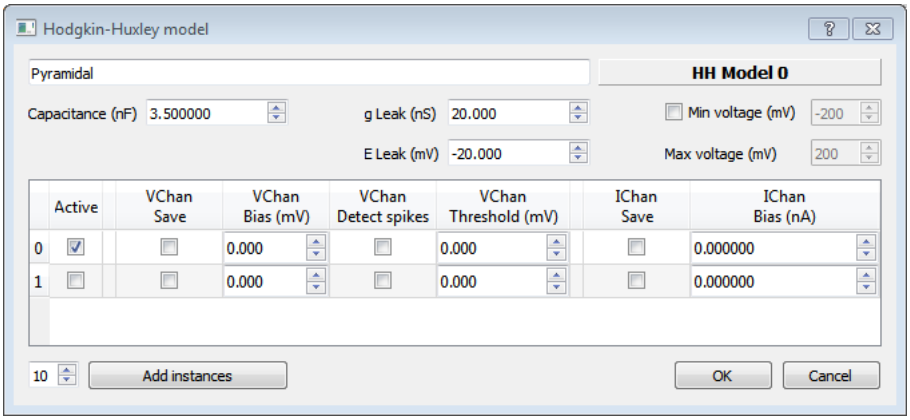
<device>.outChn[#].␣	Value range	Notes
`active`	0,1	Channels must be initially active. Deactivating a channel suspends all synapses and conductances that rely on it.
`bias`	double	Bias current

7.4.6 Hodgkin-Huxley model neurons

This data source type creates a single membrane compartment model. The base parameters for a model are its capacitance and a passive leak conductance, defined with its conductivity "g Leak" and its reversal potential "E Leak". Below these parameters, there is a list of instances, which can be thought of as individual neurons. Each instance can be active or inactive, and it creates a voltage channel ("VChan", with options for data saving, voltage offset, and spike detection), and a current channel ("IChan", with options for data saving and current offset). These channels appear alongside analog input and output channels, respectively, in channel selection dialogs elsewhere. Instance-specific channels are labelled "HH X:Y", where X is the model number, and Y is the instance number.

In order to complete a neuron model, you will need to define a number of Hodgkin-Huxley conductances (see section 7.6) and assign these to the model. This is done by selecting the model's "All" channel ("HH X:all") for both the voltage and the current channel assignment of the conductance. Using the "All" channel guarantees that the conductance is applied to all active instances of the

model.
In a similar manner, synapses can be assigned to all active instances of a model using the "All" channel, or to a specific instance using that instance's numbered channel.

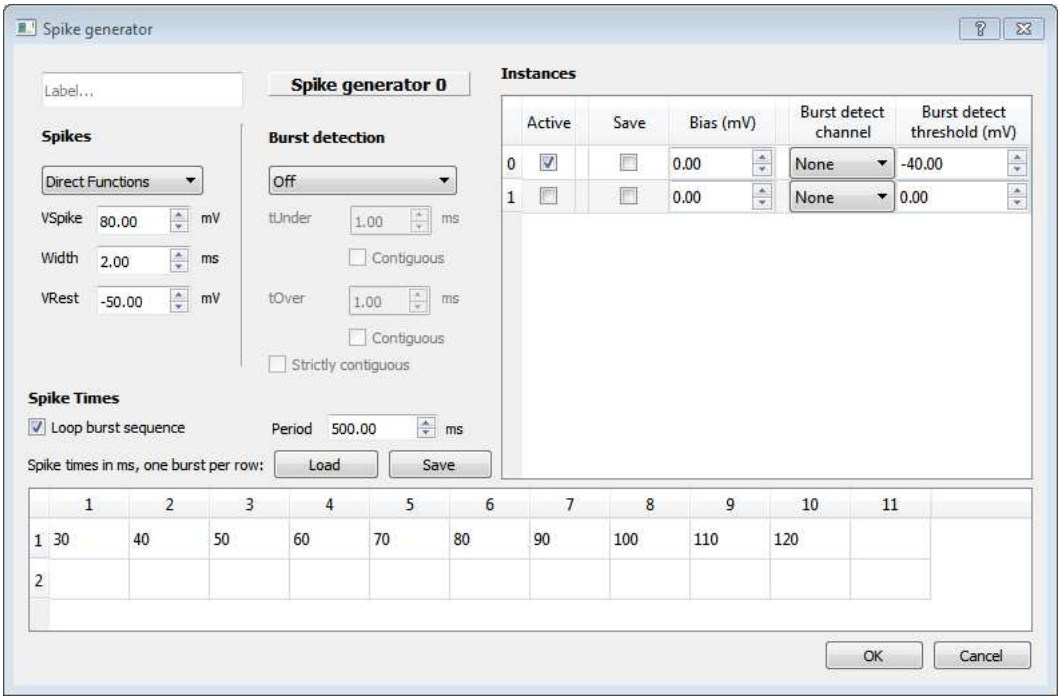


Scripting: Script access to models is given through the `HHNeuronp[#]` variable. The following members are available for scripting:

HHNeuronp[#].␣	**Value range**	**Notes**
`active`	0,1	Models must be initially active. Deactivating a model suspends all of its instances.
`C`	double	
`gLeak`	double	
`ELeak`	double	
`inst[#].active`	0,1	Model instances must be initially active. Deactivating a model instance suspends its input and output channels and all synapses and conductances that rely on them.
`inst[#].inChn.spkDetect`	0,1	Spike detection on/off
`inst[#].inChn.spkDetectThreshold`	double	
`inst[#].inChn.bias`	double	Bias voltage
`inst[#].outChn.bias`	double	Bias current

7.4.7 Spike generator

The spike generator unit can replace a presynaptic cell. It generates spikes of a predefined shape, in a temporal pattern defined by the user. It can operate in a triggered mode, producing spike patterns (bursts) in response to a threshold crossing on a different (measured) channel, or in stand-alone mode, producing bursts in periodic fashion.
Note: In earlier versions of StdpC, the spike generator could also be used to replay recorded voltages from files. This function is no longer available here, but can easily be replaced by using a SimulDAQ data source instead.



The control elements for the spike generator are:

"Spikes" : Each spike rises exponentially from the resting potential, reaches its peak at the designated spike time, then drops off symmetrically to its rise.

- VSpike: The amplitude of the spikes generated
- Width: The width of spikes in ms
- VRest: The resting potential from which the spikes depart

"Burst detection"

- The transition combo allows to choose whether to trigger bursts independently of external events (Off), or whether to trigger an event for low to high or high to low transitions through a threshold value. Threshold detection is performed on a separate channel for each instance of the spike generator, see below.
- Threshold: For low→high detection, an event is triggered whenever tUnder milliseconds were below threshold and afterwards tOver milliseconds were above. Note that for noisy signals it may be necessary to increase both numbers for reliable detection.
- When the respective "Contiguous" box is checked, the events below and above are required to be contiguous. Otherwise, time spent below or above threshold is added up cumulatively.
- Strictly contiguous: Checking this box means that an event is triggered if and only if the burst detection voltage makes a single, smooth threshold crossing.

"Instances"

- The spike generator can be instantiated multiple times. Each instance has options to save the output, to offset it by a fixed value, and to detect threshold crossings on a specific channel, at a given threshold voltage.

- Each spike generator instance creates a voltage channel named "SG X:Y", where X is the spike generator number, and Y is the instance number.

  "Spike times"

- Loop burst sequence: If this box is checked, the burst sequence is repeated. Otherwise, the spike generator will turn itself off once all bursts are generated.
- Period: If burst detection is turned off, each burst is generated in order, triggered at the start of the clamp cycle, and once for each period. Bursts that last longer than the period are truncated.
- Spike times are given in the table at the bottom of the dialog. Each line corresponds to one burst, which has an unrestricted number of spikes. Spike times are given in milliseconds relative to the start of the burst. Note: Spike times are sorted in ascending order when the dialog is closed and before saving to file. Spike times at t=0.0 are ignored.
- Load, Save buttons: Burst patterns can be read from and written to files. StdpC expects descriptions of spike patterns containing

  - An integer denoting the number of spikes in the pattern
  - A matching number of spike times in seconds, measured from the detection event.

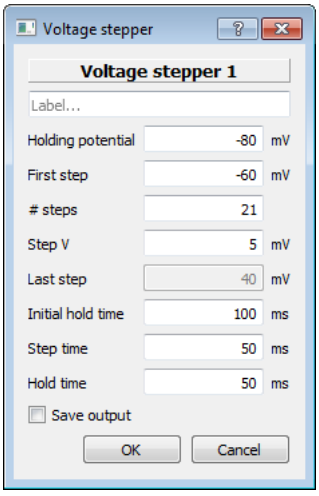
  These pattern descriptions are best separated by newlines. Note: Previous versions of StdpC allowed input files with a single spike sequence in untriggered mode. Since all modes now allow several bursts, such files must be converted; simply add the number of spikes before the first spike time value.

Scripting: Script access to spike generators is given through the `SGp[#]` variable. The following members are available for scripting:

SGp[#].␣	Value range	Notes
`active`	0,1	Spike generators must be initially active. Deactivating a spike generator suspends all of its instances.
`VSpike`	double	
`spkTimeScaling`	double	= 5000 / Spike width (ms)
`VRest`	double	
`bdType`	0,1,2	Burst detection: 0-off, 1-low to high, 2-high to low
`bdtUnder`	double	
`bdtUnderCont`	0,1	Contiguous on/off
`bdtOver`	double	
`bdtOverCont`	double	
`bdStrictlyCont`	double	
`period`	double	
`loopBursts`	double	
`SpikeT[#][#]`	double	Spike times in s relative to burst initiation. The first index is for the burst, the second for the spike. Indexing is zero-based. Note that if bursts or spikes are added by a script, they will exist from the moment the clamp cycle is started. E.g., if the protocol specifies one burst, and the script defines SpikeT[2][5] at any time, there will be one empty burst (index 1) and one burst with 6 spikes at time 0 (index 2). Empty bursts are not skipped, they just happen to produce no spikes. Spike times at or below 0 are ignored.
`inst[#].active`	0,1	Spike generator instances must be initially active. Deactivating an instance suspends its voltage channel and all synapses and conductances that rely on it.
`inst[#].inChn.bias`	double	Bias voltage
`inst[#].bdThresh`	double	Burst detection threshold

7.4.8 Voltage stepper

The voltage stepper generates a series of square voltage pulses that can be used e.g. for software-defined voltage clamp protocols.



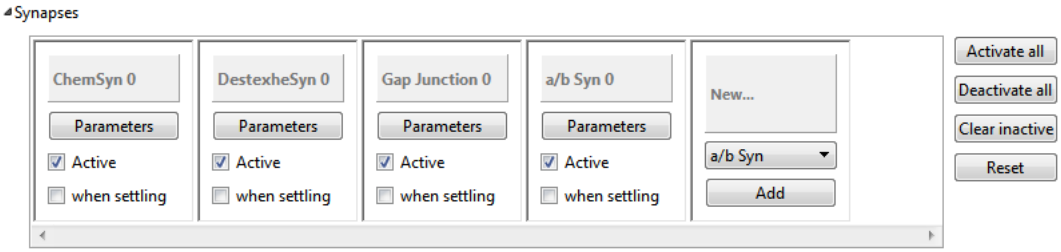
The following parameters control the step generation process:

- Holding potential: The base voltage before, between, and after any steps.
- First step: The voltage of the first step.
- # steps: The number of steps to be generated.
- Step V: The voltage offset added to each step after the first. E.g., with a first step at -60 mV and a step V of 5 mV, the second step will be at -55 mV, the third at -50 mV, etc.
- Last step: Displays the voltage of the final step in the series, given the three inputs above.
- Initial hold time: Duration of the period before the first step, during with the holding potential is applied.

- Step time: Duration of each step.
- Hold time: Duration of the holding period between steps.
- Save output: Saves the channel data to file, if data saving is activated.

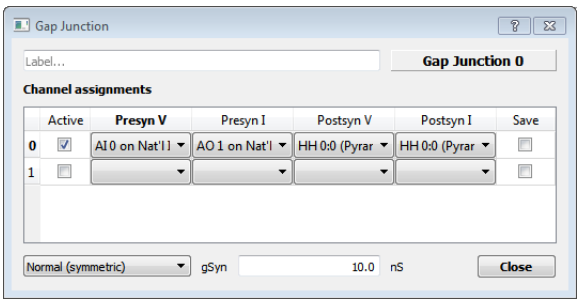
The voltage stepper module does not currently have any scriptable variables.

7.5 Synapses



The synapse control block contains an arbitrary number of synapse configurations. Synapses are added by selecting the appropriate type in the rightmost box in the list, labelled "New...", and clicking "Add". An initially active synapse is then added to the list, which can be configured by clicking the "Parameters" button and activated or deactivated by checking or unchecking the "Active" checkbox. When a synapse is no longer required, it can be removed by deactivating it and clicking the "Clear inactive" button to the right of the list. The "Reset" button just below that resets the entire control block to its most recent saved state, i.e. the state it was in upon program start, loading or saving a protocol, or clicking the "Start" button.

7.5.1 Electrical synapses (Gap Junctions)



The currents I_{pre} and I_{post} to be injected into the pre- and post-synaptic cells, respectively, are calculated according to:

$$I_{\text{post}}(t) = g_{\text{Syn}}[V_{\text{pre}}(t) - V_{\text{post}}(t)], \quad (1)$$

$$\text{and} \quad I_{\text{pre}}(t) = -I_{\text{post}}(t), \quad (2)$$

where $V_{\text{pre}}(t)$ and $V_{\text{post}}(t)$ are the membrane potential of the two cells. If the gap junction is chosen as rectifying, $I_x(t) = 0$ if $V_{\text{pre}}(t) < V_{\text{post}}(t)$.

The controls in the gap junction dialog are

- Channel assignments: Each individual gap junction is assigned to presynaptic and postsynaptic voltage (input) and current (output) channels and can be individually turned on and off.
  Note that assignments to neuron model "All" channels are resolved intelligently (e.g. assigning presynaptic V and I to HH 0:All and postsynaptic V and I to analog channels gives rise to one synapse per active model instance) and on an all-to-all basis (e.g. assigning presynaptic V and I to HH 0:All and postsynaptic V and I to HH 1:All causes every instance of HH 0 to form synapses with each instance of HH 1).
- Rectification: You can choose whether the gap junction is ordinary ("Normal (symmetric)", the current can flow from the pre-synaptic cell to the postsynaptic cell and vice versa; both cells

have perfectly symmetrical roles in this case) or rectifying (positive current can only flow from the presynaptic cell to the postsynaptic cell but not in the other direction).

- gSyn: The synaptic conductance in nS.

Scripting: Script access to gap junctions is given through the `ESynp[#]` variable. The following members are available for scripting:

ESynp[#].␣	**Value range**	**Notes**
`active`	0,1	Synapses must be initially active. Deactivating a synapse suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`type`	0,1	0-normal, 1-rectifying
`gSyn`	double	

7.5.2 Chemical Synapses

The current to be injected into the postsynaptic cell, I_{post}, is calculated in each dynamic clamp cycle using a first order kinetics model of the release of neurotransmitter, an additional inactivation term, $h(t)$, to simulate short term depression, and an optional magnesium block term, $g_{\text{Mg}}(t)$:

$$I_{\text{post}} = g_{\text{Syn}}\, g_{\text{Mg}}(t)\, S(t)\, h(t)[V_{\text{Syn}} - V_{\text{post}}(t)], \tag{3}$$

where the instantaneous activation, S(t), and inactivation, h(t), terms are given by the differential equations

$$(1 - S_\infty(V_{\text{pre}}))\tau_{\text{Syn}} \frac{dS(t)}{dt} = (S_\infty(V_{\text{pre}}) - S(t)) \tag{4}$$

$$\tau_h \frac{dh(t)}{dt} = h_\infty(V_{\text{pre}}) - h(t), \tag{5}$$

where

$$S_\infty(V_{\text{pre}}) = \begin{cases} \tanh\left[\frac{V_x(t) - V_{\text{Thresh}}}{V_{\text{Slope}}}\right] & \text{if } V_{\text{pre}} > V_{\text{Thresh}} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$h_\infty(V_{\text{pre}}) = \frac{A}{1 + \exp\left(\frac{V_{\text{pre}} - V_{\text{Thresh}}}{V_{\text{Slope}}}\right)}, \tag{7}$$

$$\tau_h(V_{\text{pre}}) = \tau_0 - \frac{A_\tau}{1 + \exp\left(\frac{V_{\text{pre}} - V_{\text{Thresh},\tau}}{V_{\text{Slope},\tau}}\right)} \tag{8}$$

and the optional magnesium block term is given by

$$g_{\text{Mg}}(t) = \frac{1}{1 + F_{\text{Mg}}\, \exp(X_{\text{Mg}}\, V_{\text{post}}(t))} \tag{9}$$

The controls in the parameter dialog for the chemical synapse are

- Channel assignments: Each individual synapse is assigned to a presynaptic input channel (Presyn V) and postsynaptic input and output channels (Postsyn V, I).
- Delay: Each assignment comes with a fixed delay (in ms), which can be used to mimic conduction latencies.

"General":

- gSyn: The maximal conductance g_{syn} of the synapse in nS.
- VSyn: The reversal potential V_{Syn} in mV.
- tauSyn: The characteristic time constant τ_{Syn} of the synapse in ms.
- VThresh: The threshold potential V_{Thresh} for the release of neurotransmitter in mV.
- VSlope: The "slope" parameter V_{Slope} of the activation curve in mV.

"Stochastic":

- Check to enable stochastic postsynaptic potentials, defined after [Redman, 1990]. Each upward threshold crossing triggers one independently calculated synaptic event, as follows:
  The number of active release sites n during a given synaptic event is drawn from a binomial distribution, $n \sim B(n_{\mathrm{rel}}, p_{\mathrm{rel}})$. Then, a quantal amplitude factor q is drawn from a normal distribution, $q \sim N(n, \frac{n\sigma_q}{n_{\mathrm{rel}}p_{\mathrm{rel}}})$, and multiplied with $S(t)$ to yield a stochastically scaled postsynaptic conductance.
  Note, this formulation implies that g_{syn} is the mean conductance per release site.
- # release sites: Number of independent transmitter release sites n_{rel}.
- p(release): Transmitter release probability per site p_{rel}.
- PSP variance: The variance σ_q^2 of the conductance evoked by a single release event.

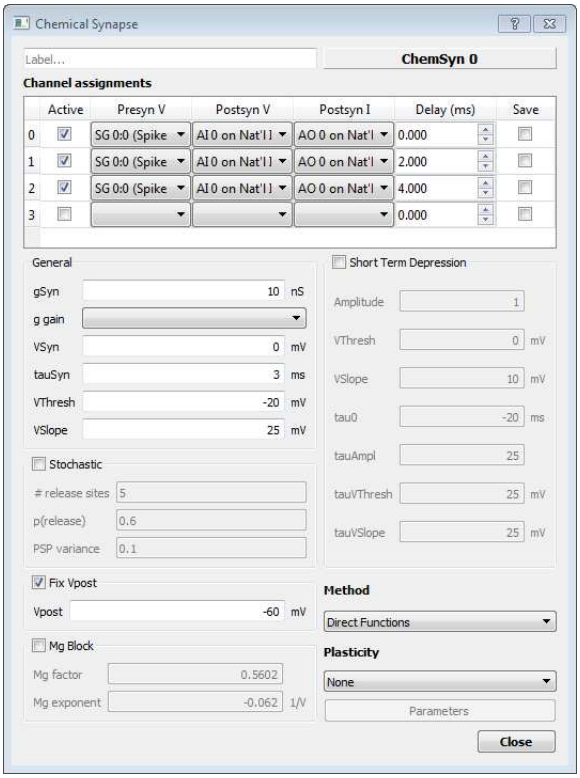
"Fix Vpost":

- Check to draw V_{post} from a set value, rather than from the postsynaptic voltage channel.
- Vpost: The fixed value to use for V_{post}.

"Mg Block":

- Check to enable the magnesium block term g_{Mg}, defined after [Fellous and Sejnowski, 2003].
- Mg factor: The scaled magnesium concentration F_{Mg}, a unitless value.
- Mg exponent: The exponential factor X_{Mg} in 1/V.

"Method": Any sigmoid, tanh and exponential functions can optionally be computed using lookup tables. This is less precise than direct methods, but may increase speed on old computers.



"Short Term Depression":

- Check to enable.
- Amplitude: The amplitude of the $h(t)$ depression variable. Typically set to 1 (so that h varies between 0 and 1). This was previously used to switch short term depression on or off and remains for historical reasons.
- VThresh: The threshold potential $V_{\mathrm{Thresh},\tau}$ for the activation of h in mV.
- VSlope: The "slope" parameter of the depression activation curve in mV.
- tau0, tauAmpl, tauVThresh, and tauVSlope: Parameters τ_0, A_τ, $V_{\mathrm{Thresh},\tau}$ and $V_{\mathrm{Slope},\tau}$ for the voltage-dependent characteristic time τ_h of short term depression.
- The plasticity combo box allows you to choose whether the synapse is subject to long term plasticity and according to which model the plasticity is determined.

For "Spike STDP" a spike-timing based rule is applied according to the parameters defined in the corresponding control panel that appears upon clicking the "Parameters" button. If "ODE STDP" is chosen, the synaptic plasticity is implemented according to the ordinary differential equation (ODE) description in [Abarbanel et al., 2002]. Parameters again are adjusted in the separate panel that appears after pressing "Parameters". For details on both methods, see section 7.5.5.

Scripting: Script access to chemical synapses is given through the `CSynp[#]` variable. The following members are available for scripting:

CSynp[#].␣	Value range	Notes
active	0,1	Synapses must be initially active. Deactivating a synapse suspends all its assignments.
assign[#].active	0,1	Assignments must be initially active.
gSyn	double	
VSyn	double	
tauSyn	double	
VThresh	double	
VSlope	double	
stochastic	0,1	Stochasticity on/off
stoch_nRel	int	
stoch_pRel	double	
stoch_variance	double	
fixVpost	0,1	
Vpost	double	
STD	0,1	Short-term depression on/off
STDAmpl	double	
STDVThresh	double	
STDVSlope	double	
STDtau0	double	
STDtauAmpl	double	
STDtauVTresh	double	
STDtauVSlope	double	
MgBlock	0,1	
MgFac	double	
MgExpo	double	
Plasticity	0,1,2	0-off, 1-Spike 2-ODE. See below for STDP scripting.

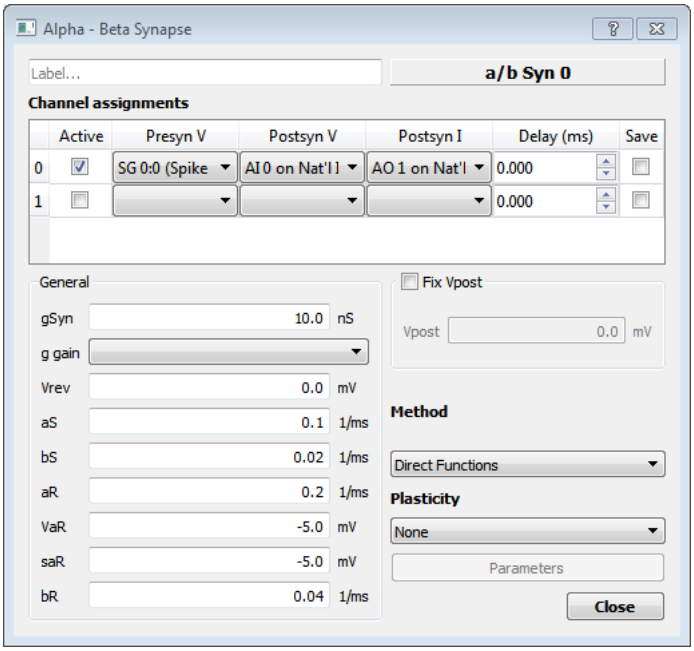
7.5.3 Alpha beta synapse

The $\alpha\beta$ synapse implemented in StdpC is a variation on the classic Rall synapse [Rall, 1967] with a pre-synaptic release variable R and a post-synaptic binding variable S which then gates the synaptic current. The model is described by:

$$I_{\text{syn}} = g_{\text{syn}} S (V_{\text{rev}} - V_{\text{post}}) \tag{10}$$

$$\frac{dS}{dt} = \alpha_S (1 - S) R - \beta_S S \tag{11}$$

$$\frac{dR}{dt} = \alpha_R (1 - R) \frac{1}{1 + \exp(\frac{V_{\text{pre}} - V_{\alpha R}}{s_{\alpha R}})} - \beta_R R \tag{12}$$



- Channel assignments, method, fixed Vpost and plasticity are described above.
- gSyn: Maximal synaptic conductance
- Vrev: Reversal potential of the synapse
- aS: Rate of transmitter binding to post-synaptic targets
- bS: Rate of transmitter removal (channel closing) postsynaptically
- aR: Rate of transmitter release when pre-synaptic potential is elevated
- VaR: Midpoint of the sigmoid function expressing the level of pre-synaptic release as a function of pre-synaptic membrane potential
- saR: Width of the release sigmoid function
- bR: Fall rate of the presynaptic release

Scripting: Script access to alpha-beta synapses is given through the `abSynp[#]` variable. The following members are available for scripting:

abSynp[#].␣	**Value range**	**Notes**
`active`	0,1	Synapses must be initially active. Deactivating a synapse suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`gSyn`	double	
`Vrev`	double	
`aS`	double	
`bS`	double	
`aR`	double	
`VaR`	double	
`saR`	double	
`bR`	double	
`fixVpost`	0,1	
`Vpost`	double	
`Plasticity`	0,1,2	0-off, 1-Spike, 2-ODE. See below for STDP scripting.

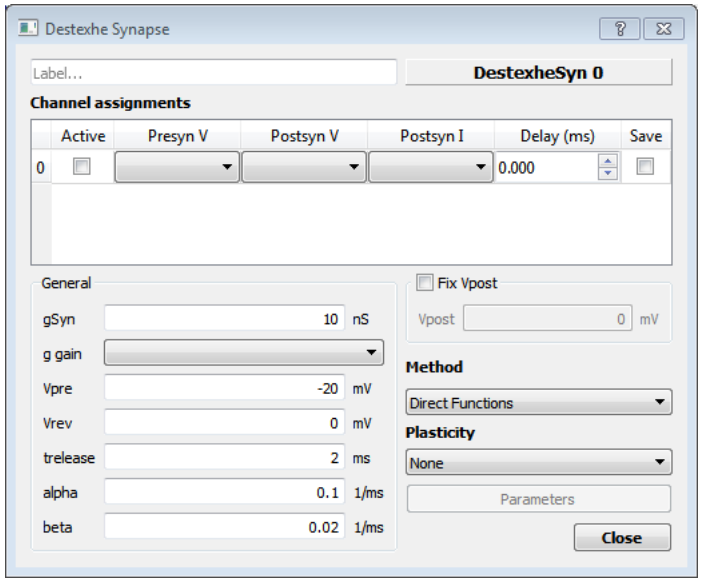
7.5.4 Destexhe synapse

This synapse model is the simple first order synapse description introduced in [Destexhe et al., 1994]. In brief,

$$I_{\text{syn}} = g_{\text{syn}} S (V_{\text{rev}} - V_{\text{post}}) \tag{13}$$

$$\frac{dS}{dt} = \begin{cases} \alpha(1-S) - \beta S & \text{if } t - t_{\text{spike}} < t_{\text{release}} \\ -\beta S & \text{otherwise} \end{cases} \tag{14}$$

where t_{spike} is the time of the occurrence of the last spike in the pre-synaptic neuron.



- Channel assignments, method, fixed Vpost and plasticity are described above.
- gSyn: Maximal synaptic conductance
- Vpre: pre-synaptic threshold for triggering synaptic transmitter release
- Vrev: Reversal potential of the synapse
- trelease: The time (duration) of transmitter release after a presynaptic spike
- alpha: The rise rate for the EPSCs
- beta: the fall (decay) rate for the EPSCs

Scripting: Script access to Destexhe synapses is given through the `DxheSynp[#]` variable. The following members are available for scripting:

abSynp[#].␣	**Value range**	**Notes**
`active`	0,1	Synapses must be initially active. Deactivating a synapse suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`gSyn`	double	
`Vpre`	double	
`Vrev`	double	
`trelease`	double	
`alpha`	double	
`beta`	double	
`fixVpost`	0,1	
`Vpost`	double	
`Plasticity`	0,1,2	0-off, 1-Spike, 2-ODE. See below for STDP scripting.

7.5.5 Spike Timing Dependent Plasticity

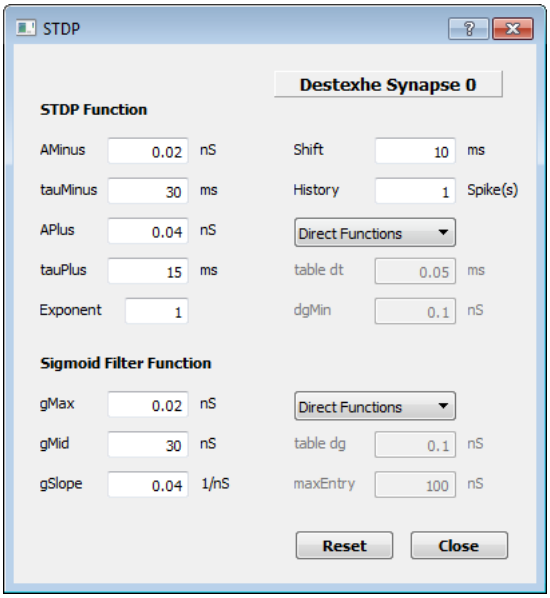
As described above, synapses can be equipped with a form of Spike Timing Dependent Plasticity (STDP). The typical way of implementation, denoted as "Spike STDP", is to detect spikes in the pre- and postsynaptic cells and define

$$\Delta g = \pm A_{\pm} \left(\frac{|\Delta t - \tau_{\text{Shift}}|}{\tau_{\pm}} \right)^{q} \exp\left(- \frac{|\Delta t - \tau_{\text{Shift}}|}{\tau_{\pm}} \right). \tag{15}$$

Δg is then added to the "raw" synaptic conductance g_{raw} whenever a pre- or postsynaptic spike occurs. Note that for correct function, spike detection needs to be switched on for the pre- and postsynaptic input channels (see section 7.4.4). The synaptic conductance g_{Syn} is then determined from g_{raw} through a sigmoid filter

$$g = g_{\max} \tanh\left(\frac{g_{raw} - g_{\text{Mid}}}{g_{\text{Slope}}} \right) \tag{16}$$

to avoid problems of "run-away" potentiation.



The control parameters are:
"STDP Function"

- AMinus: The amplitude A_{-} of the negative (left) part of the STDP curve.
- tauMinus: The time scale τ_{-} (locus of the extremum) of the negative (left) part of the STDP curve.
- APlus: The amplitude A_{+} of the positive (right) part of the STDP curve.
- tauPlus: The time scale τ_{+} (locus of the extremum) of the positive (right) part of the STDP curve.
- Exponent: The exponent q of the polynomial factor in the STDP curve.
- Shift: The offset τ_{Shift} of the STDP curve on the Δt axis.
- History: The number of spikes to be considered in the other neuron if a spike occurs in a given neuron. For example, if set to 1 and a spike occurs in the postsynaptic neuron, the change Δg will only be calculated and applied for the last spike that occurred in the presynaptic neuron. If it was 2 it would be calculated for the last and the next to last spike in the presynaptic neuron. Note, history is capped at 20 spikes.
- The method combo box allows to choose whether the STDP function is calculated directly with the appropriate C functions when needed or whether it is tabled up front and this lookup-table is being used.
- table dt: The time step in the lookup table.
- dgMin: The minimum value for Δg that should be in the table. This determines how far to the left and right the table covers the STDP curve.

"Sigmoid Filter Function"

- gMax: The maximal g_{Syn} allowed.
- gMid: The midpoint g_{Mid} of the filter.
- gSlope: The "slope" parameter g_{slope} of the sigmoid filter.
- The method combo allowing the choice between direct calculation or lookup tables for the filter function.
- table dg: The stepping in terms of g_{raw} of the lookup table.
- maxEntry: The maximal g_{raw} entry in the table.

Scripting: Script access to Spike STDP is given through the `<synapse>.ST` variable. The following members are available for scripting:

<synapse>.ST.␣	**Value range**	**<synapse>.ST.␣**	**Value range**
`AMinus`	double	`Shift`	double
`tauMinus`	double	`History`	integer [0..20]
`APlus`	double	`gMax`	double
`tauPlus`	double	`gMid`	double
`Exponent`	integer	`gSlope`	double

Alternatively, if the "ODE STDP" option is chosen, the synaptic strength is governed by a set of differential equations according to the model of synaptic plasticity in [Abarbanel et al., 2002]. In this case, the maximal synaptic conductance g_{Syn} is subject to a differential equation system

$$\frac{dP}{dt} = v_{\mathrm{pre}} - \beta_P P \tag{17}$$

$$\frac{dD}{dt} = v_{\mathrm{post}} - \beta_D D \tag{18}$$

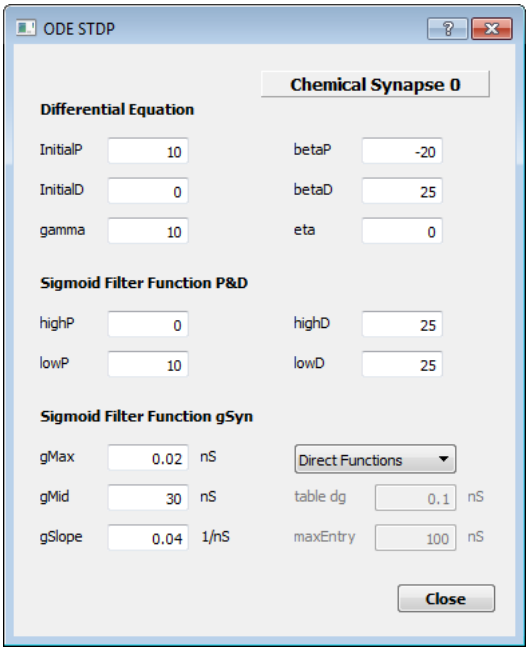
$$\frac{dg_{\mathrm{raw}}}{dt} = \gamma(PD^{\eta} - DP^{\eta}). \tag{19}$$

The normalized voltages v_x are derived from the measured potentials V_x through capped linear filters

$$v_{\mathrm{pre}} = \begin{cases} 0 & V_{\mathrm{pre}} < V_{P,\min} \\ \frac{V_{\mathrm{pre}} - V_{P,\min}}{V_{P,\max} - V_{P,\min}} & V_{P,\min} \le V_{\mathrm{pre}} \le V_{P,\max} \\ 1 & V_{P,\max} < V_{\mathrm{pre}} \end{cases} \tag{20}$$

and accordingly for v_{post},

$$v_{\mathrm{post}} = \begin{cases} 0 & V_{\mathrm{post}} < V_{D,\min} \\ \frac{V_{\mathrm{post}} - V_{D,\min}}{V_{D,\max} - V_{D,\min}} & V_{D,\min} \le V_{\mathrm{post}} \le V_{D,\max} \\ 1 & V_{D,\max} < V_{\mathrm{post}} \end{cases} \tag{21}$$



The control parameters for ODE based STDP are:
"Differential Equation"

- InitialP: Initial value for the "potentiation variable" P
- InitialD: Initial value for the "depression variable" D
- gamma: Exponent γ
- betaP: Rate of decay β_P of P in 1/ms= kHz.
- betaD: Rate of decay β_D of D in 1/ms= kHz.
- eta: Rate of change of g_{raw} in nS/ms.

"Sigmoid Filter Function P & D"

- highP: The upper limit $V_{P,\max}$ of the P filter
- lowP: The lower limit $V_{P,\min}$ of the P filter
- highD: The upper limit $V_{D,\max}$ of the D filter
- lowD: The lower limit $V_{D,\min}$ of the D filter

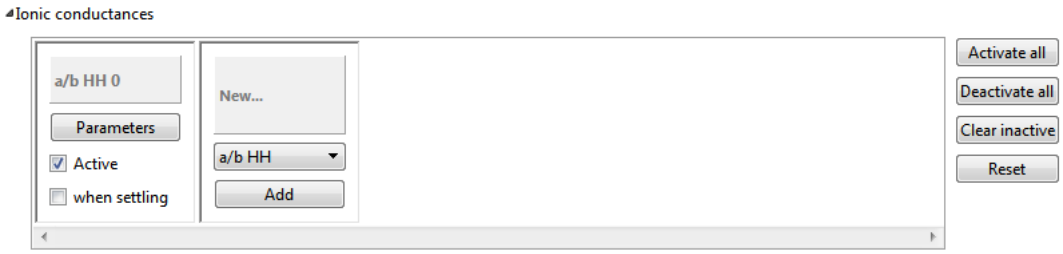
"Sigmoid Filter Function gSyn"

- gMax: The maximal allowed value for g_{Syn}.
- gMid: The mid point of the sigmoid filter for g_{Syn}
- gSlope: The "slope" parameter of the sigmoid filter function for g_{Syn}.
- The method combo allows to switch between direct evaluation of the filter or the use of a lookup table
- table dg: The step size in the lookup table
- maxEntry: The maximum of g_{raw} for which table entries are generated.

Scripting: Script access to ODE STDP is given through the `<synapse>.ODE` variable. The following members are available for scripting:

<synapse>.ODE.	Value range	<synapse>.ODE.	Value range
`InitialP`	double	`highP`	double
`InitialD`	double	`lowP`	integer [0..20]
`betaP`	double	`highD`	double
`betaD`	double	`lowD`	double
`gamma`	double	`gMax`	double
`eta`	integer	`gMid`	double
		`gSlope`	double

7.6 Ionic Conductances



The conductance control block contains an arbitrary number of Hodgkin-Huxley type conductance configurations. Conductances are added by selecting the appropriate type in the rightmost box in the list, labelled "New...", and clicking "Add". An initially active conductance is then added to the list, which can be configured by clicking the "Parameters" button and activated or deactivated by checking or unchecking the "Active" checkbox. When a conductance is no longer required, it can be removed by deactivating it and clicking the "Clear inactive" button to the right of the list. The "Reset" button just below that resets the entire control block to its last saved state, i.e. the state it was in upon program start, loading or saving a protocol, or clicking the "Start" button.

7.6.1 m/h/tau Hodgkin-Huxley conductances

For the "m/h/tau" conductance description, the current I_{HH} to be injected into a cell with membrane potential $V(t)$ is calculated according to

$$I_{HH}(t) = g_{\mathrm{Max}} m(t)^p h(t)^q (V_{\mathrm{rev}} - V(t)), \tag{22}$$

where m and h are modelled as

$$\tau_m \frac{dm}{dt} = m_\infty(V) - m, \tag{23}$$

$$\tau_h \frac{dh}{dt} = h_\infty(V) - h, \tag{24}$$

with steady state values

$$m_\infty = \frac{1 - C_m}{1 + \exp\left(\frac{V - V_m}{s_m}\right)} + C_m, \tag{25}$$

$$h_\infty = \frac{1 - C_h}{1 + \exp\left(\frac{V - V_h}{s_h}\right)} + C_h, \tag{26}$$

and time scales

$$\tau_m = \tau_{0,m} + A_{\tau,m} F_{\tau,m}(V), \tag{27}$$

$$\tau_h = \tau_{0,h} + A_{\tau,h} F_{\tau,h}(V). \tag{28}$$

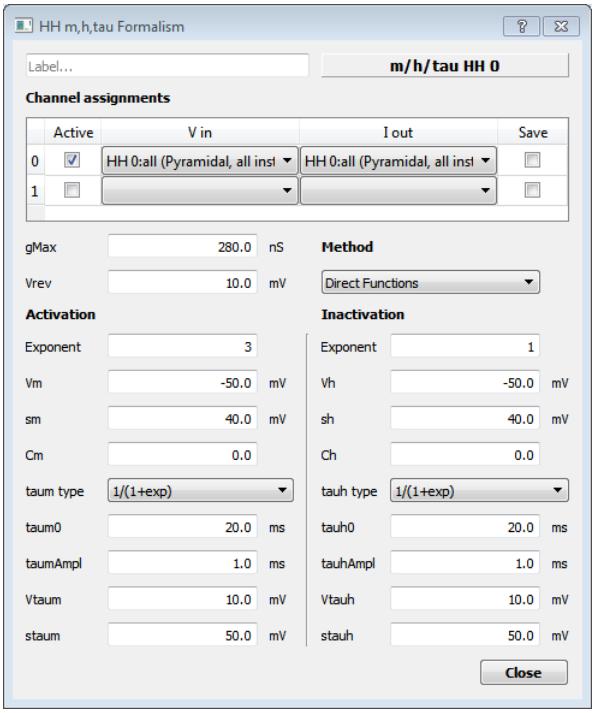
The time scale function $F_{\tau,\bullet}$ can be one of the following four choices:

$$\text{“1/(1+exp)”} : F_{\tau,\bullet} = \frac{-1}{1+\exp\left(\frac{V-V_{\tau,\bullet}}{s_{\tau,\bullet}}\right)}, \tag{29}$$

$$\text{“tanh\^2”} : F_{\tau,\bullet} = 1-\tanh^2\left(\frac{V-V_{\tau,\bullet}}{s_{\tau,\bullet}}\right), \tag{30}$$

$$\text{“exp2/(1+exp1)”} : F_{\tau,\bullet} = \exp\left(\frac{V-V_{\tau,\bullet}}{s_{\tau,\bullet}}\right) \bullet_\infty (V), \tag{31}$$

$$\text{“Linear”} : F_{\tau,\bullet} = V. \tag{32}$$



Control parameters of the ionic conductances in the m/h/tau formalism are:

- Channel assignments: Conductances can be assigned to multiple pairs of voltage input (“V in”) and current output (“I out”) channels. Each such assignment can be turned on or off individually. Assignments to model “All” channels are resolved intelligently, such that each active instance of the model receives one conductance.
- gMax: The maximal conductance g_{Max} of the inserted channel
- Vrev: The reversal potential V_{rev} of the inserted channel
- Method: The combo box allows to choose between direct evaluation of C functions or a pre-calculated lookup table.

“Activation” and “Inactivation” (substituting h for m as appropriate):

- Exponent: The exponent p (Activation) or q (Inactivation) in the current equation
- Vm: The activation potential V_m
- sm: The width s_m of the activation function
- Cm: The offset parameter C_m for persistent currents
- taum type: Selects the time scale function $F_{\tau,m}$.
- taum0: The minimal time scale $\tau_{0,m}$ for τ_m.
- taumAmpl: The range (Amplitude) $A_{\tau\, m}$ for the time scale τ_m in ms (or, when taum type is “Linear”, ms/mV).
- Vtaum: The mid point $V_{\tau,m}$ for the time scale sigmoid function.
- staum: The “slope” parameter $s_{\tau,m}$ for the time scale sigmoid function

Scripting: Script access to m/h/tau conductances is given through the `mhHHp[#]` variable. The following members are available for scripting:

mhHHp[#].␣	**Value range**	**Notes**
`active`	0,1	Conductances must be initially active. Deactivating a conductance suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`gMax`	double	
`Vrev`	double	
`mExpo`, `hExpo`	integer	Exponent
`Vm`, `Vh`	double	
`sm`, `sh`	double	
`Cm`, `Ch`	double	
`taumType`, `tauhType`	0,1,2,3	Ordered as above
`taum`, `tauh`	double	taum0, tauh0
`taumAmpl`, `tauhAmpl`	double	Units: For linear tau type, the value is interpreted as s/mV, otherwise as s.
`Vtaum`, `Vtauh`	double	
`staum`, `stauh`	double	

7.6.2 Alpha-Beta Hodgkin-Huxley conductances

Alternatively, the ionic conductances can be described in a α/β formalism, where

$$I_{\mathrm{HH}} = g_{\mathrm{Max}} m^p h^q (V_{\mathrm{rev}} - V) \tag{33}$$

$$\frac{dm}{dt} = \alpha_m (1 - m) - \beta_m m \tag{34}$$

$$\alpha_m = k_{\alpha,m} F_{\alpha,m} \left(\frac{V - V_{\alpha,m}}{s_{\alpha,m}} \right) \tag{35}$$

$$\beta_m = k_{\beta,m} F_{\beta,m} \left(\frac{V - V_{\beta,m}}{s_{\beta,m}} \right) \tag{36}$$

$$\frac{dh}{dt} = \alpha_h (1 - h) - \beta_h h \tag{37}$$

$$\alpha_h = k_{\alpha,h} F_{\alpha,h} \left(\frac{V - V_{\alpha,h}}{s_{\alpha,h}} \right) \tag{38}$$

$$\beta_h = k_{\beta,h} F_{\beta,h} \left(\frac{V - V_{\beta,h}}{s_{\beta,h}} \right). \tag{39}$$

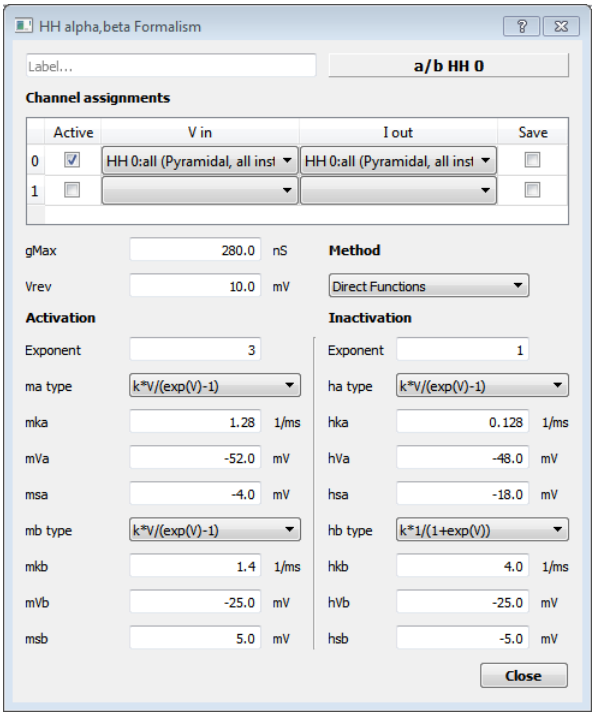
The "activation" and "inactivation" functions $F_{\bullet,\bullet}$ can be chosen from three options,

$$\text{"k*V/(exp(V)-1)"} : F_{\bullet,\bullet}(x) = \frac{x}{\exp(x) - 1} \tag{40}$$

$$\text{"k*exp(V)"} : F_{\bullet,\bullet}(x) = \exp(x) \tag{41}$$

$$\text{"k*1/(1+exp(V))"} : F_{\bullet,\bullet}(x) = \frac{1}{1 + \exp(x)}. \tag{42}$$

Note that this formalism allows you to implement, for example, the classic neuron model by Traub and Miles [Traub and Miles, 1991].



The control parameters are

- Channel assignments, gMax, Vrev, Method as above;

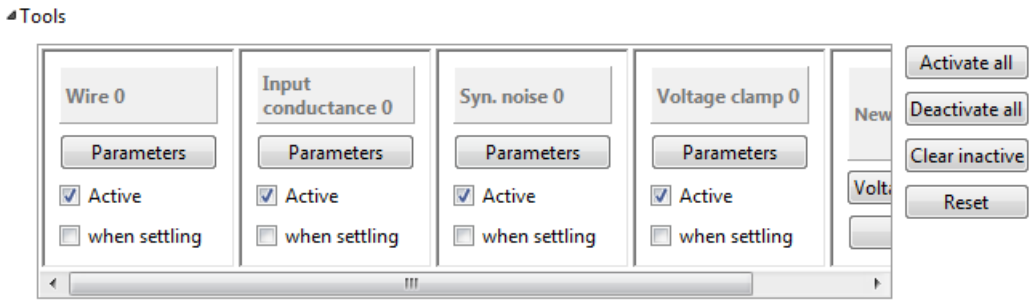
"Activation" and "Inactivation" (substituting h for m as appropriate):

- Exponent: The exponent p (Activation) or q (Inactivation) in the current equation
- ma type: The choice of the functional form for $F_{\alpha,m}$.
- mka: Rate parameter $k_{\alpha,m}$ for the rise of m
- mVa: The midpoint $V_{\alpha,m}$ for the sigmoid function for α_m
- msa: The "slope" parameter $s_{\alpha,m}$ for the sigmoid function for α_m
- mb type: The choice of the functional form for $F_{\beta,m}$.
- mkb: Rate parameter $k_{\beta,m}$ for the decay of m
- mVb: The midpoint $V_{\beta,m}$ for the sigmoid function for β_m
- msb: The "slope" parameter $s_{\beta,m}$ for the sigmoid function for β_m

Scripting: Script access to alpha-beta conductances is given through the `abHHp[#]` variable. The following members are available for scripting:

abHHp[#].␣	**Value range**	**Notes**
`active`	0,1	Conductances must be initially active. Deactivating a conductance suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`gMax`	double	
`Vrev`	double	
`mExpo`, `hExpo`	integer	Exponent
`maFunc`, `mbFunc`, `haFunc`, `hbFunc`	0,1,2	Activation/inactivation functions, ordered as above.
`mka`, `mkb`, `hka`, `hkb`	double	
`mVa`, `mVb`, `hVa`, `hVb`	double	
`msa`, `msb`, `hsa`, `hsb`	double	

7.7 Tools



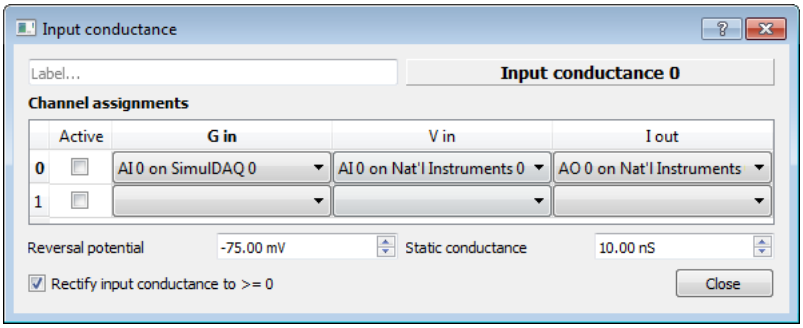
7.7.1 Wire

The wire tool is a very simple module that writes data from an input channel (e.g. from an acquisition device, from the spike generator, or similar) to an existing output channel. This circumvents StdpC's separation of input and output channels and may be useful in various non-standard use cases. Other than channel assignments, this module has only one functional parameter, namely a conversion factor.

Scripting: Script access to the Wire module is given through the `Wire[#]` variable. The following members are available for scripting:

Wire[#].␣	Value range	Notes
`active`	0,1	Modules must be initially active. Deactivating a module suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`factor`	double	The factor k used to calculate the output as $I = kV$.

7.7.2 Input conductance



The *Input conductance* tool can be used to inject an externally defined conductance waveform into a membrane. That is, rather than calculating the conductance according to a model, as the synapse and ionic conductance modules do, the conductance value is taken from an input channel, labelled "G in" in the channel assignments. For example, one might use a SimulDAQ input file to describe a conductance time series. The following controls are available:

- Reversal potential: The reversal potential of the conductance.
- Static conductance: Conversion factor k applied to the samples read from "G in". That is, the resulting current is defined as $I(t) = kG(t)(V_{\text{rev}} - V_{\text{in}}(t))$.

- Rectify input conductance to ≥ 0: If checked, any negative *conductance* values read from the "G in" channel are interpreted as zero. Note, this is not rectification of the current, only of the conductance.

Scripting: Script access to input conductances is given through the `InputCond[#]` variable. The following members are available for scripting:

InputCond[#].␣	Value range	Notes
`active`	0,1	Modules must be initially active. Deactivating a module suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`Vrev`	double	
`gStatic`	double	
`rectify`	0,1	

7.7.3 Synaptic noise

The synaptic noise module injects a pseudo-random conductance designed to mimic synaptic background noise, following the model proposed by [Destexhe et al., 2001]. A single (inhibitory or excitatory) synaptic background noise current with a reversal potential V_{rev} is calculated as follows:

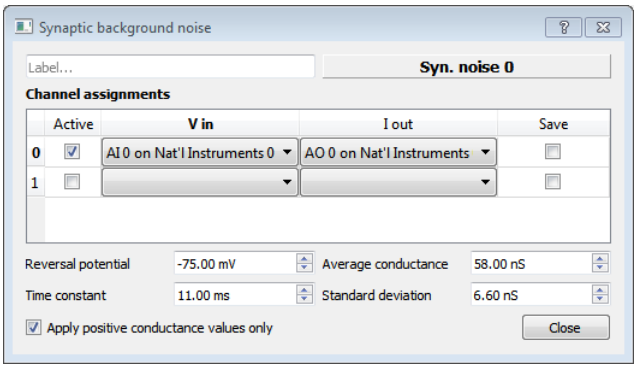
$$I(t) = g(t)(V_{\text{rev}} - V_{\text{in}}(t)) \tag{43}$$

$$g(0) = \bar{g} \tag{44}$$

$$g(t + \Delta t) = \bar{g} + (g(t) - \bar{g})e^{-\Delta t/\tau} + R\sqrt{\frac{D\tau}{2}(1 - e^{-2\Delta t/\tau})} \tag{45}$$

$$R \sim N(0, 1) \tag{46}$$

$$D = \frac{2\sigma^2}{\tau} \tag{47}$$



The following parameters are available:

- Reversal potential: V_{rev} in the above equations.
- Time constant: τ in the above equations. The time constant determines the noise quality, ranging from purely white noise ($\tau = 0$) to coloured noise that looks more like Brownian motion ($\tau > 0$).
- Average conductance: $\bar{g}$ in the above equations. The actual conductance will fluctuate around this value.
- Standard deviation: σ in the above equations. This determines how widely the conductance fluctuates.
- Apply positive conductance values only: Check to rectify the applied conductance and avoid physically impossible negative conductances. Note that $g(t)$ may fluctuate into negative values regardless, but with this setting turned on, $g(t) < 0$ implies $I(t) = 0$.

Scripting: Script access to synaptic noise conductances is given through the `SynapticNoisep[#]` variable. The following members are available for scripting:

SynapticNoisep[#].␣	Value range	Notes
`active`	0,1	Modules must be initially active. Deactivating a module suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`Vrev`	double	
`tau`	double	
`g0`	double	The average conductance $\bar{g}$.
`std`	double	The standard deviation σ.

7.7.4 Voltage clamp

This is an experimental software-defined voltage clamp implementation, using PID control to achieve voltage clamp e.g. in absence of a suitable amplifier, or – depending on the system used – possibly with better control characteristics. The module takes a command voltage input ("Command V") and calculates a control current ("Clamp I") based on the measured membrane potential ("Cell V"). Control current is the sum of proportional, integral and differential components I_p, I_i and I_d, defined as follows:

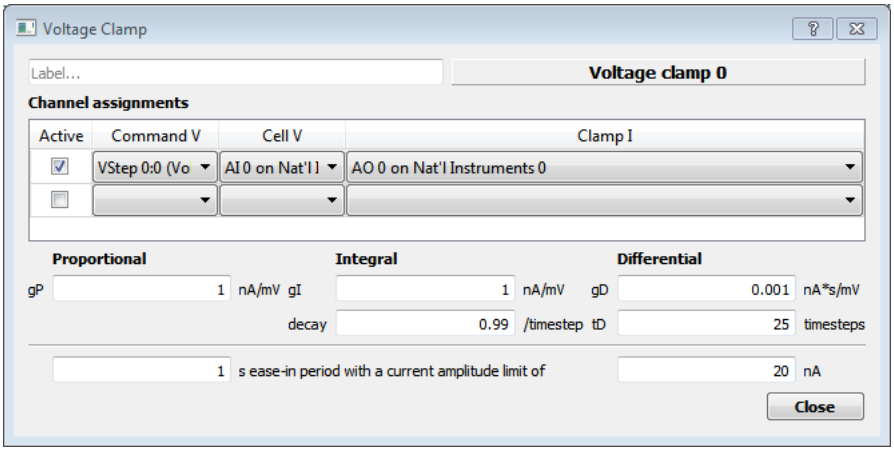
$$I_p = g_p(V_{\text{cmd}} - V_m) \tag{48}$$

$$I_i = g_i(\bar{V}_{\text{cmd}} - \bar{V}_m) \tag{49}$$

$$I_d = g_d(\frac{dV_{\text{cmd}}}{dt} - \frac{dV_m}{dt}) \tag{50}$$

The running averages $\bar{V}_{\text{cmd}}$ and $\bar{V}_m$ are calculated using a decay factor k, such that $\bar{V}(t + \Delta t) = k\bar{V}(t) + V(t)$. The differential voltages dV/dt are approximated at each time step using the difference between $V(t)$ and the voltage tD steps previous.

Besides these parameters, the module supports an ease-in period during which the total current output is multiplied with a linearly increasing factor, starting from 0 at the start of clamping to 1 at the end of ease-in. During this period, if the total current amplitude exceeds the given limit, the voltage clamp module is deactivated to protect the cell against damage from ringing.

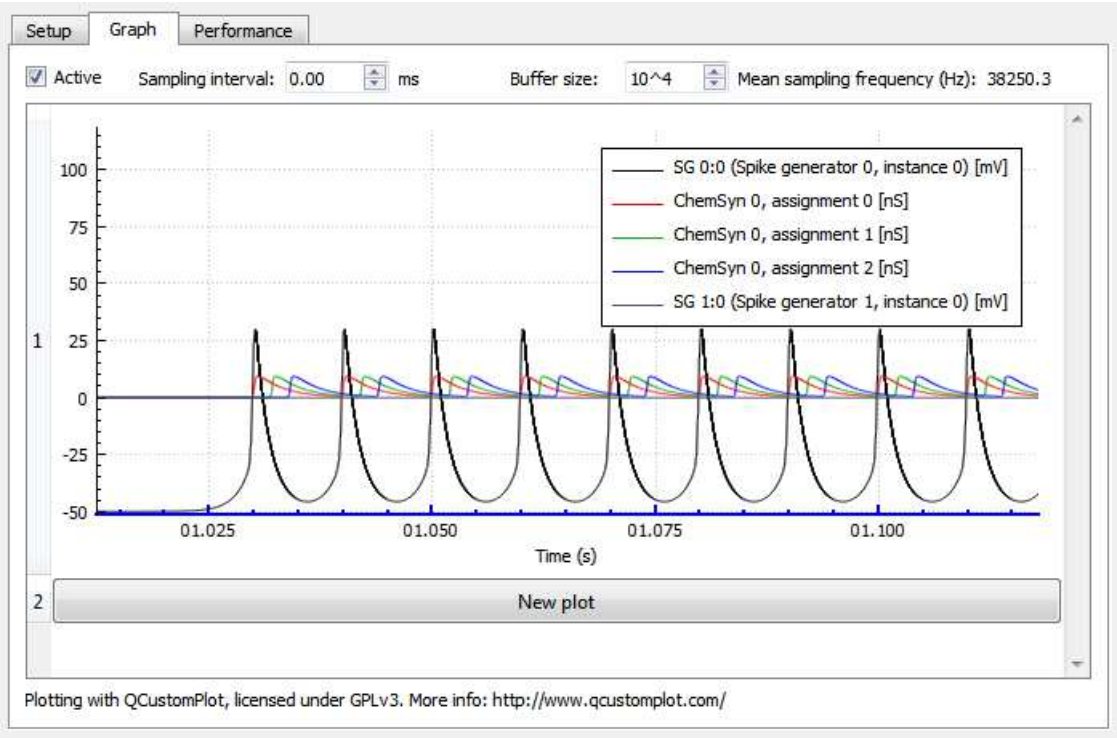


Scripting: Script access to the voltage clamp module is given through the `VoltageClamp[#]` variable. The following members are available for scripting:

VoltageClamp[#].␣	Value range	Notes
`active`	0,1	Modules must be initially active. Deactivating a module suspends all its assignments.
`assign[#].active`	0,1	Assignments must be initially active.
`gP`, `gI`, `gD`	double	
`decayI`	double	Decay factor per time step.
`easeIn`	double	Ease-in time; note that linear interpolation always starts from t=0.
`easeInAmpLimit`	double	

7.8 Data display

The data display in StdpC is very useful to check the parameter settings chosen. Typical errors in dynamic clamp setups are wrong conversion factors on input and output channels. By displaying the data acquired on input channels in the graph displays, some of these errors can easily be detected and corrected. On reasonably modern hardware (i.e., any computer with multiple processor cores), graphing should have very little impact on clamp cycle performance.



The graph panel control elements are:

- Sampling interval: Sets the graph sampling frequency that StdpC aims for. The achieved sampling frequency is displayed to the right and is usually a little smaller than the desired rate.
- Buffer size: Sets the number of samples that can be held in each trace's buffer. It may be necessary to increase this for high sampling rates. Tell-tale signs of a buffer overflow are gaps in the data, which appear as straight line segments, interspersed with smooth curve sections.

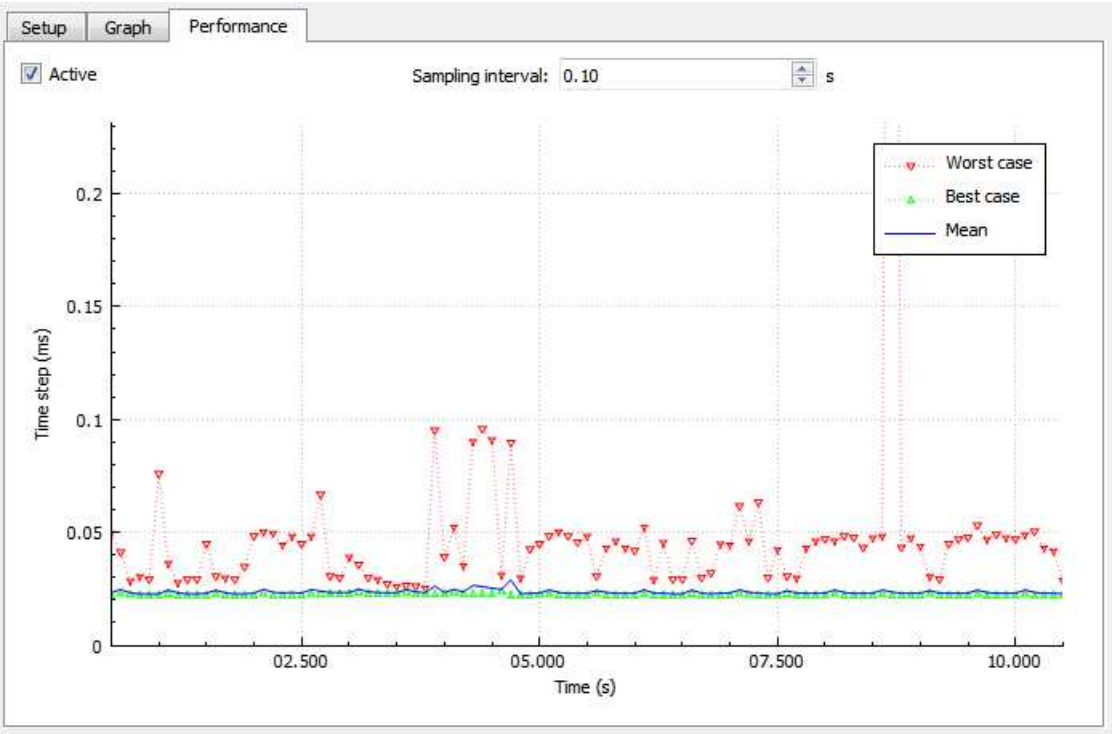
To select channels for plotting, click "New plot" or double-click an existing plot to bring up the plot configuration dialog. All input and output channels as well as raw conductance values are

available for plotting. All graphs are plotted on the same axis, but can be scaled by selecting an appropriate unit modifier.

The data display is controlled with the mouse. To zoom in or out, use the scroll wheel while mousing over the display. To move the plot, click and drag it. You can also select each of the axes by clicking them (the axis will turn blue) in order to zoom/drag along this axis only. Individual traces can be hidden by clicking their legend entry. The height of a plot can be adjusted along the left edge. These data display adjustments can be made at any time and do not interfere with the data displayed.

7.9 Performance monitor

The performance monitor is a valuable diagnostic tool which displays the average clamp cycle duration, as well as best and worst case markers. The performance monitor has two control elements; a checkbox to enable the monitor, and an input to set the sampling interval. Worst case, best case and mean cycle duration are all computed with regards to the sampling interval.



8 Experiment automation

So far we have described how to control StdpC using the graphical user interface. Instead, or in addition, most parameters can be controlled through a scripting mechanism. Through the File → Load Script one can specify a script file. The script file should contain three columns:

1. A time in seconds when a certain event shall occur
2. A parameter name in clear text which shall be changed
3. A new value for the parameter in question

The value column can contain double precision numbers or integers depending on the parameter being changed.

The script file will be read into a list of events in memory such that changes to the file have no effect after the script was loaded.

The parameter names are the same as used when the state of the GUI is saved ("save protocol"), and are listed in the relevant sections above. Note that protocols contain further parameters that cannot be changed while the clamp is running, and which are therefore not listed above. Their inclusion in a script file is discouraged and may lead to unexpected behaviour.

Please note that in script and protocol files, all parameters and variables are in SI units V, A, s, S, etc. unless noted otherwise.

9 Electrode compensation

For the compensation of electrode artifacts, StdpC uses the so-called Active Electrode Compensation (AEC) algorithm, developed by Romain Brette and his colleagues. Here we only discuss those parts of this cutting-edge compensation technique, that are necessary for the StdpC end-user to be aware of, however we also encourage the user to consult [Brette et al., 2008] and its supplementary material for the full description of the algorithm.

The AEC method consist of two parts: an initial calibration phase and the subsequent compensation phase happening during the actual experiment. The first part is the calibration phase, during which StdpC first injects a probing current into the cell (which is an uncorrelated white noise signal), and records the voltage response of the system (comprising of the cell and the electrode) to that current. Based on the acquired signal-response relationship, the software then calculates the actual parameter values of the used electrode for the general AEC model. Having had the electrode calibrated, in the second phase, StdpC automatically uses this digital model of the electrode on the corresponding input channel, and calculates the artefactual voltage drop across the electrode occurring on the recording of that channel due to current injection through the electrode.

9.1 Basic considerations and troubleshooting for electrode compensation

High fidelity artifact compensation requires well-calibrated electrodes, thus the calibration phase is a crucial part of the electrode compensation process. Even though it is implemented in the most automated way possible, there are several points the user must make sure before calibration and take care after calibration (just like in the case of any other calibration technique). Briefly these are the following (see [Brette et al., 2008] Supp. Mat. for details):

- *Do the calibration only when the electrode is already in the cell.* Electrode properties change after cell membrane impalement, and the electrode has to be calibrated in its stable state.
- *Make sure that the electrode is (sufficiently) linear in terms of its steady state resistance.* One of the basic assumptions of AEC is that the steady state electrode voltage response changes linear with the amplitude of the current level injected into it. The Electrode Compensation dialog has a built-in feature to assess the linearity of the electrode (see below). If you experience too high deviation in the steady-state voltage levels in response to different voltage levels (e.g. above 10% of the electrode resistance), we recommend to change the electrode, and rerun the linearity test for the new one.
- *Make sure that the electrode has reached an equilibrium state in terms of its steady state resistance before calibration.* During experiment, any change in the electrode properties after calibration would result in an error in the artifact compensation. The most varying property is the resistance of the electrode, which can change remarkably especially in the first few minutes

after cell impalement. Thus one must always make sure that the steady-state electrode resistance is not changing any more before calibration. The electrode measurement feature on the Electrode Compensation dialog provides a convenient way for this assessment (see below).

- *Make sure that the electrode is at least ten times faster than the cell membrane.* Another basic assumption of the AEC calibration phase is that the electrode and the passive cell membrane (ie. not spiking, but subthreshold) responses are separable in the time domain. This usually satisfied in general electrode/cell configurations, but does not stand in case of a very slow electrode and a very fast cell membrane. The Electrode Calibration dialog provides means for the measurement of both electrode and cell membrane response properties (resistance levels, time constants and their standard deviations, see below). As in the case of large cells it can be difficult to effectively drive and measure the cell's subthreshold properties, generally one can satisfy this criterion by making sure that the electrode time constant (which can be measure more reliably) is around half millisecond or less. If the electrode has been found too slow, one can use a certain level of capacitance neutralization (CN) during the calibration and the compensation. This CN level must be set before calibration, and left unchanged during the usage of the same calibrated model of the electrode. See also next point for applying CN.

- *Turn other compensation techniques off before calibration and during compensation as well.* AEC is a complete electrode compensation method. The simultaneous operation of any other active electrode compensation technique, like bridge balancing or discontinuous injection/recoding, would lead to disastrous compensation results. The only exception from this rule is the capacitance neutralization feature of many microelectrode amplifier, which is able to increase the artifact compensation fidelity of AEC, if it is carefully set to the maximum possible level before calibration and left unchanged during the experiment (see previous point).

- *Assess actual data acquisition frequency after calibration.* Right after calibration the user should make sure that StdpC had been able to run the data acquisition part of the compensation in a relatively constant frequency by observing the *Std in sampling rate* text field in the *Data acquisition results* subpanel (in the bottom right corner). The software automatically warns the user if this deviation is higher than 0.01, what results in a degradation in the quality of the electrode calibration and thus in the subsequent artifact compensation as well. Most of the time the popping up of this warning means that a significant operation system interruption has occurred during calibration (it is also possible that the hardware (PC and/or DAQ) is not able to reach the required sampling rate set in the top right corner, in which case lowering the rate should solve the issue). To avoid the operation system interruption problem, and also to reach the best clamping quality, we strongly recommend to stop all computational intensive processes the computer might run in the background during calibration and dynamic clamping.

- *Recalibrate if electrode properties has changed over time.* As for all electrode artifact compensation technique, the compensation quality degrades if changes in the electrode properties (mostly in the resistance) have occured as a long experiment goes on. To rule out this possibility, we recommend the check of the electrode resistance or even the recalibration of the electrode from time to time during the experiment, and also the assessment of the electrode after the experiment has finished to validate the quality of the compensation.

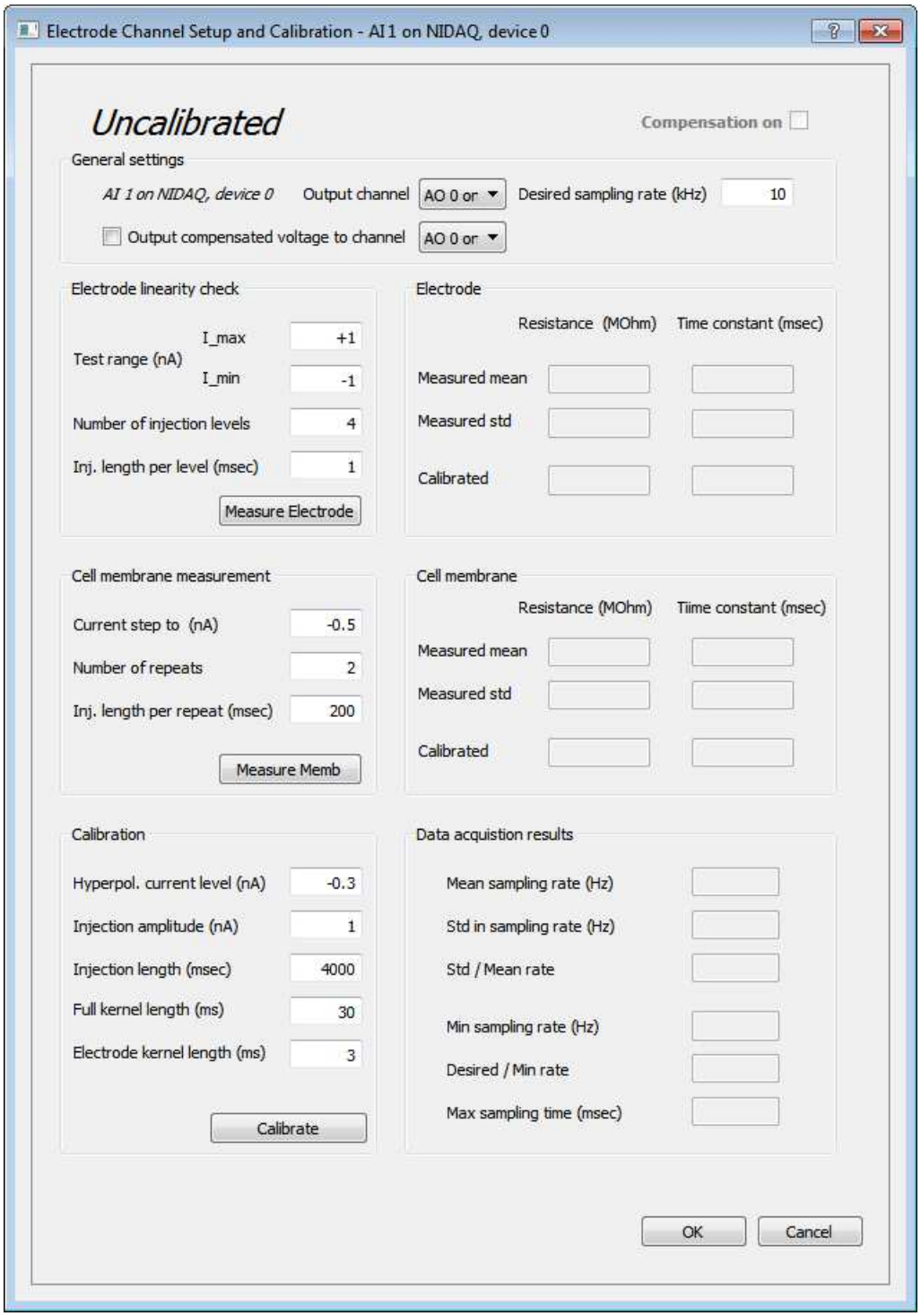
9.2 Supported features and the dialog panel

Along with the electrode calibration, the Electrode Compensation dialog has two auxiliary features to facilitate calibration and subsequent artifact compensation at high quality.

The *electrode measurement* feature acquires the electrode resistance and time constants according to a simple parallel RC circuit model by injecting different constant current levels into the

electrode and simultaneously recording the onset-dynamic and equilibrium level of the voltage response. There are three important things to assess here: (1) the electrode is fast enough (ie. that its average time constant is less then about 1 ms), (2) the electrode is linear enough (ie. that the standard deviation in its resistance levels at different current levels is below 10% of its absolute average resistance), and (3) the electrode has reached an equilibrium level in its resistance (ie. the average electrode resistance does not change remarkably in between subsequent measurements). Also see the troubleshooting subsection above for a more detailed discussion of these issues. It is crucial to find a long enough, but not too long injection length while the electrode is in the cell, in order to let the electrode reach its steady state voltage response, but in the same also time prevent the cell membrane to markedly influence the measurement of the electrode characteristics. A rule of thumb is to set the injection length between 3 and 5 times of the expected electrode time constant, what usually leads to injection lengths around 1 and 2 ms. An incidental spike of the cell simultaneously with the electrode measurement always result in incorrectly measured properties, this case the one should remeasure the electrode.

The *cell membrane measurement* feature allows for the measure of the same (passive) properties of the cell membrane, based on the same parallel RC model of the membrane and the already acquired electrode characteristics. This measurement is done by injecting a particular current step into the cell, holding it for a certain duration, then stepping back to zero current level, waiting for the same amount of time to allow the membrane potential to recover, and then doing the same injection/waiting cycle for a user-settable number of times. This feature is included to allow the user to get a rough idea about the time constant of the cell membrane and this way the appropriateness of the electrode for AEC (the electrode must be at least 10 times faster than the passive cell membrane for good quality artifact compensation) Also see the troubleshooting subsection above for a more detailed discussion of this issues. Please be aware, that because of the nature of measurement protocol, it is likely to provide good cell properties, only if (1) the electrode has been measured previously and its resistance has not changed significantly meanwhile, (2) the passive membrane response is at least one order of magnitude slower than the electrode (only required for precise membrane time constant calculation), and most importantly (3) the injected current is able to effectively drive the membrane potential of the cell, that is, no indenpendent activity of the cell (let it be sub- or suprathreshold) intervene and contaminate the measurement remarkably. This last point implies that successful membrane property measurement of large cells probably requires high amplitude currents, if it is possible at all with this technique. As large cells usually have slower membrane, in these cases the user can satify the "time-separability" criterion (see Troubleshooting section) by making sure that they do not use a very slow electrode (ie. time constant is lower than 1ms).



The calibration dialog is reached through the input channel setup dialog. The control and result elements are the following:

- Information line: Each page has a text line on the top (saying *Uncalibrated* on the included picture) to provide some basic information about the calibration process for the user.

- *Compensation on*: after calibration, this checkbox allows the user to turn the compensation on and off on the corresponding electrode.

General settings: This panel contains the settings that apply to all the available features (ie. the electrode measurement, the membrane measurement and the calibration).

- *Output channel*: The output channel assigned to the electrode, must be one of the active output channels.
- *Output compensated voltage to channel* Check and select a different channel to copy the compensated voltage to. This can be used to record the voltage actually used by StdpC on a separate machine, for example.
- *Desired sampling rate*: The sampling frequency that StdpC will aim to keep during the calibration or the electrode/membrane measurement.

Electrode linearity check: This panel contains the settings specific for the electrode measurement feature.

- *I_max*: Maximal current level injected into the electrode.
- *I_min*: Minimal current level injected into the electrode.
- *Number of injection levels*: Number of current levels injected during electrode measurement. The current levels are equally spaced between the maximal and minimal levels, with the limits included. For example, if I_max = 1 nA, I_min = -1 nA and *Number of current levels* = 4, then the current levels to be injected are $I_1 = -1nA, I_2 = -1/3nA, I_3 = +1/3nA, and I_4 = +1nA$. Zero current level is always skipped.
- *Inj. length per level*: defines how long the injection of each current level lasts.
- *Cell membrane measurement*: This panel contains the settings specific for the cell membrane measurement feature. See also *General settings*.
- *Current step to*: Sets the current step at which the cell properties will be measured. Negative current steps are highly recommended, as depolarization increases the chance that active membrane phenomena, like spiking, will disrupt the measurement of passive membrane properties.
- *Number of repeats*: defines how many time the above current level will be injected into the cell. Higher repeat numbers give numerical stability to the measurement, but only if no spike has occured during any of the injections.
- *Inj. length per repeat*: defines how long each injection will lasts. Also defines the recovery time in between current level injections.

Calibration: This panel contains the settings specific for the electrode electrode calibration feature.

- *Hyperpol. current level*: allows for the injection of a certain constant level of hyperpolarization current into the cell during the whole duration of the calibration in order to minimize the number of spikes occurring during calibration. High hyperpolarization levels (ie. above -1nA) are not favourable, as they can remarkably change the electrode characteristics.
- *Injection amplitude*: the amplitude of the injected white noise current signal. Setting 1 nA here usually means a good tradeoff between reaching a high signal-to-noise ratio and avoiding driving the cell into an active state (like spiking) in the same time. Always make sure that the DAQ hardware allows for the issuing of this current level (both to positive and negative levels).

- *Injection length*: defines how long the current injection during the calibration phase lasts. In most cases, a 4-5 second long calibration is sufficient to acquire statistically significant amount of data for the electrode parameter calculation.
- *Full kernel length*: one of the crucial parameters of the AEC technique. A rule of thumb is to set it to one or two times of the time constant of the membrane. After cell membrane measurement StdpC automatically fills this field, however, as the membrane property measurement can be quite imprecise in some cases (huge cell, changing electrode properties or spiking activity), always check that it is somewhere in between 30 and 60 ms before issuing the calibration command.
- *Electrode kernel length*: the other very important parameter for the calibration. A rule of thumb here is to set it to 10 times of the time constant of the electrode, but to no more than 5ms. After electrode measurement StdpC automatically fills this field, however, always check that it is somewhere in between 1 and 5 ms before issuing the calibration command.

10 Saving of experiment data

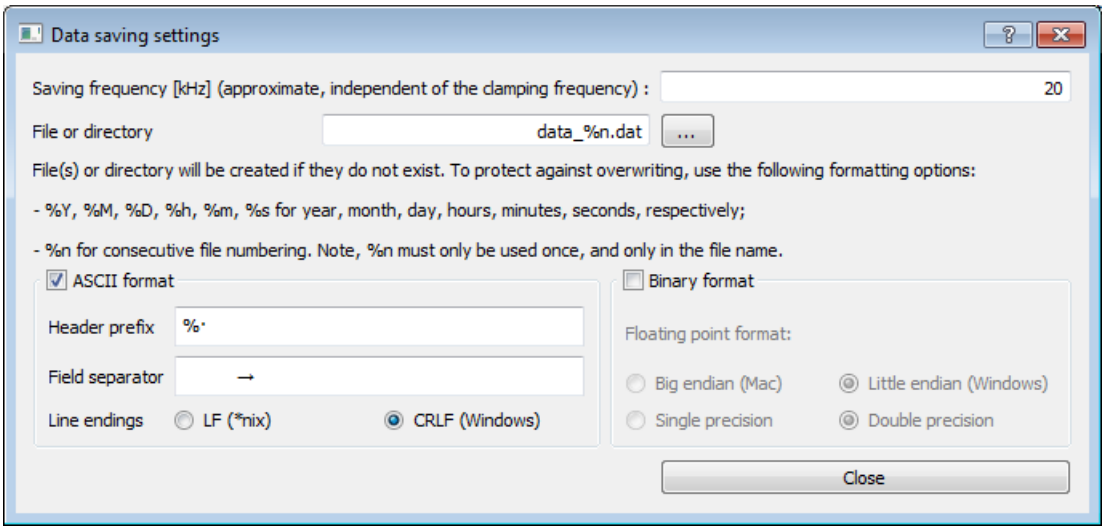
StdpC allows saving experiment data to disk. The user can choose between binary and ASCII data formats, pick any subset of the active channels (see Section 7.4.4 and 7.4.5), set the data saving frequency and select a file which the experiment data will be saved into. After starting the dynamic clamp, in each clamping cycle the software decides whether it is time to write samples to file by comparing the last saving time and the current time. This way the saving frequency is quasi independent of the dynamic clamping frequency.
In ASCII mode, when a data saving occurs, a new line is added to the data file obeying the following pattern:

$$Time\_stamp \quad Input\_chnl\_1 \quad ... \quad Input\_chnl\_n \quad Output\_chnl\_1 \quad ... \quad Output\_chnl\_n$$

Please be aware, that, as no offline compression is implemented in the current version, StdpC can generate huge data files during long experiments. The size of the saved data can be reduced by

- choosing binary format rather than ASCII
- decreasing the saving frequency
- only saving the channels which are going to be important for data post-processing and analysis

For a simple comparison, the simulation of a gap junction (electrical synapse) between two neurons produces about 30 MByte/minute data if the user wants to save all the four channels (2 input and 2 output channels + the extra time stamp) at 10kHz frequency in ASCII mode, while the same experiment produces about 3.6 MByte/minute data in case the user only saves the membrane voltages of the two cells (2 input channels + the extra time stamp) at 5kHz in binary mode. Having finished the experiment, the user can load the experiment data file into any data analysis platform they prefer (Matlab, R, Octave, Scilab, etc) for offline processing and analysis of the data.



The configuration settings on the *Data saving settings* dialog are the following:

- Saving frequency: sets the target frequency of data saving.
- Filename: Indicates the name of the file (ASCII mode) or the directory (binary mode) where data will be saved. Use placeholders as indicated. Year (%Y) and consecutive numbers (%n) are four digits, all other fields are two digits. Consecutive numbering respects any existing files; if e.g. there are already files named "data_0001.dat" and "data_0004.dat" in the same location, StdpC will write to a new file named "data_0005.dat" on the next clamping run.
- ASCII format: Writes the output as a text file. The first line of the file is a header with the given header prefix, labelling the columns of data. Each subsequent row is a snapshot of the data in the chosen channels, at the time (in seconds after clamp cycle initiation) indicated in the first column. Columns are separated with the character(s) specified in the "Field separator" input.
- Binary format: Instead of writing to a single file, the given filename is interpreted as a directory name. In this directory, each saved channel is saved to a separate file, along with a file containing the time stamps. In addition to the raw data, a file named "meta.json" is deposited in the same directory, describing the data set in JSON format.

11 User feedback

Please do give feedback if you found the software useful, buggy, or have general comments or questions. Please send correspondence through GitHub or directly to T.Nowotny@sussex.ac.uk. If you are interested in contributing to the future development please do not hesitate to contact me as well.

12 Acknowledgements

The StdpC software would not exist without the early dynamic clamp versions developed by Reynaldo D. Pinto and I (TN) am grateful to him for supplying the source code for further development back then. We are also grateful to both Attila Szücs and Naoki Kogo for their excellent suggestions for improving and extending the software and their tireless testing of new and buggy versions.

References

Abarbanel, H. D. I., Huerta, R., and Rabinovich, M. I. (2002). Dynamical model of longt-term synaptic plasticity. *P Natl Acad Sci USA*, 99:10132–10136.

Brette, R., Piwkowska, Z., Monier, C., Rudolph-Lilith, M., Fournier, J., Levy, M., Frégnac, Y., Bal, T., and Destexhe, A. (2008). High-resolution intracellular recordings using a real-time computational model of the electrode. *Neuron*, 59(3):379–391.

Destexhe, A., Mainen, Z. F., and Sejnowski, T. J. (1994). An efficient method for computing synaptic conductances based on a kinetic model of receptor binding. *Neural Comput*, 6:14–18.

Destexhe, A., Rudolph, M., Fellous, J.-M., and Sejnowski, T. J. (2001). Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. *Neuroscience*, 107(1):13–24.

Fellous, J.-M. and Sejnowski, T. J. (2003). Regulation of persistent activity by background inhibition in an in vitro model of a cortical microcircuit. *Cerebral Cortex*, 13(11):1232.

Hodgkin, A. L., Huxley, A. F., and Katz, B. (1949). Ionic current underlying activity in the giant axon of the squid. *Arch Sci Physiol*, 3:129–150.

Kemenes, I., Marra, V., Crossley, M., Samu, D., Staras, K., Kemenes, G., and Nowotny, T. (2011). Dynamic clamp with stdpc software. *Nature Protocols*, 6(3):405–417.

Nowotny, T., Szucs, A., Pinto, R. D., and Selverston, A. I. (2006). Stdpc: a modern dynamic clamp. *J Neurosci Methods*, 158(2):287–299.

Pinto, R. D., Elson, R. C., Szücs, A., Rabinovich, M. I., Selverston, A. I., and Abarbanel, H. D. I. (2001). Extended dynamic clamp: Controlling up to four neurons using a single desktop computer and interface. *J Neurosci Meth*, 108:39–48.

Rall, W. (1967). Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic inputs. *J Neurophysiol*, 30:1138–1168.

Redman, S. (1990). Quantal analysis of synaptic potentials in neurons of the central nervous system. *Physiological reviews*, 70(1):165–198.

Robinson, H. P. and Kawai, N. (1993). Injection of digitally synthesized synaptic conductance transients to measure the integrative properties of neurons. *J Neurosci Meth*, 49:157–165.

Sharp, A. A., O'Neil, M. B., Abbott, L. F., and Marder, E. (1993). Dynamic clamp: computer-generated conductances in real neurons. *J Neurophysiol*, 69:992–995.

Traub, R. D. and Miles, R. (1991). *Neural Networks of the Hippocampus*. Cambridge University Press, New York.