

RTHybrid

v1.1.2

USER MANUAL



Grupo de Neurocomputación Biológica
(GNB)

Universidad Autónoma de Madrid

23.06.2020

Summary

1	Introduction	1
I	How to install it	2
2	Environment configuration	2
2.1	Real-time operating systems (RTOS)	2
2.1.1	Preempt-RT	2
2.1.2	Xenomai 3	2
2.2	Comedi DAQ drivers	3
2.3	Other libraries required by RTHybrid	3
2.3.1	Qt5	3
2.3.2	libxml2	4
3	RTHybrid installation	4
II	How to use it	5
4	Launching the program	5
5	Graphical user interface	5
6	Command-line interface	8
7	Data storage	11
III	Model library	12
8	Integration methods	12
9	Neuron models	12
9.1	None	12
9.2	Izhikevich, 2003	12
9.3	Hindmarsh and Rose, 1984	12
9.4	Rulkov, 2002	13
9.5	Ghiglazza and Holmes, 2004	13
9.6	Wang, 1993	14
9.7	Komendantov and Kononenko, 1996	15
9.8	Nowotny et al, 2008	15

10 Synapse models	17
10.1 None	17
10.2 Electrical	17
10.3 Golowasch et al, 1999	18
10.4 Destexhe et al, 1994	18
10.5 Greenberg and Manor, 2005	19

1 Introduction

Complete characterization of the complex non-linear and partially observable neural dynamics, influenced by the previous situation and feedback, is a difficult task, specially when addressed using the traditional stimulus-response paradigm. Closed-loop technology deals with this issue by providing novel ways of online observation, control and bidirectional interaction.

However, utilizing these techniques in neuroscience experiments is frequently a challenging task due to the strict temporal precision required when acquiring data and stimulating the biological components. In order to comply with such restrictions, often under the scale of the milliseconds or lower, real-time technology is needed. There exists a wide range of real-time tools, but they usually present numerous difficulties in their installation and utilization, and many of them have a high economic cost as well. A lot of researchers and laboratories overlook implementing closed-loop solutions due to these complications. Therefore, experimental neuroscience can greatly benefit from the existence of standardized and accessible real-time tools.

RTHybrid is a free and open-source software that includes a neuron and synapse model library to build hybrid circuits with living neurons through dynamic clamp. It has been developed to run over Linux, an open-source operating system, and both of its two most important real-time implementations, Xenomai and Preempt-RT. RTHybrid is the first of a collection of standardized, multiplatform, user-friendly and open-source software tools designed for open- and closed-loop experimental neuroscience.

If you use RTHybrid, please cite the following papers:

RTHybrid: a standardized, multiplatform, real-time software model library for experimental neuroscience

Rodrigo Amaducci, Manuel Reyes-Sanchez, Irene Elices, Francisco B. Rodriguez, Pablo Varona. Frontiers in Neuroinformatics, 13, 11. (2019); doi: <https://doi.org/10.1101/426643>.

Automatic adaptation of model neurons and connections to build hybrid circuits with living networks

Manuel Reyes-Sanchez, Rodrigo Amaducci, Irene Elices, Francisco B. Rodriguez Pablo Varona. Neuroinformatics (2020); doi: <https://doi.org/10.1007/s12021-019-09440-z>.

Further description of the algorithms is provided in these papers.

Contact:

Rodrigo Amaducci: rodrigo.amaducci@uam.es

Manuel Reyes: manuel.reyes@uam.es

Pablo Varona: pablo.varona@uam.es

Part I

How to install it

2 Environment configuration

RTHybrid is an open-source project and all the external libraries that it uses, as well the operating systems supported, are also open-source and free. The following elements need to be installed in order to use RTHybrid.

2.1 Real-time operating systems (RTOS)

The program can be run over any GNU/Linux distribution. Since they are GPOS (General Purpose Operating Systems), if hard real-time is required, special real-time patches need to be applied. RTHybrid supports both Preempt-RT¹ and Xenomai 3², and any of them can be chosen.

Once the RTOS is installed you can use by selecting it in the Boot Menu, usually inside of Advance Options. The name of the OS will be the same as the normal one, just including the tag **-rt** or **-xenomai** in the name.

2.1.1 Preempt-RT

Long way

The Linux Foundation provides an installation guide in their website (https://wiki.linuxfoundation.org realtime/documentation/howto/applications/preempt_rt_setup). It may not be a problem for users with experience in kernel configuration, but most people might find it a bit complicated.

Short way

If you are using a Debian distribution (preferably the latest version, 9) you can simply download and install the patch from the repositories by simply typing in a terminal:

```
sudo apt install linux-image-rt-amd64
```

for 64 bits OS or:

```
sudo apt install linux-image-rt-686-pae
```

if you use a 32 bits one.

Latest Debian version can be downloaded from www.debian.org and some useful resources to configure it can be found at www.github.com/manurs/xfce_config/tree/master/config_files/debian_basic and www.github.com/manurs/guides.md/blob/master/sudo.md.

2.1.2 Xenomai 3

Long way

Xenomai's developers provide an installation guide for their platform that can be found in their website (https://gitlab.deny.de/Xenomai/xenomai/wikis/Installing_Xenomai_3), but even so it can be a challenging task to do it from scratch.

If you decide to patch and install Xenomai by yourself remember to enable *Analogy* and *RTIPC* drivers.

¹ More information on Preempt-RT: <https://wiki.linuxfoundation.org realtime/start>

² More information on Xenomai: <http://xenomai.org/>

Short way

In order to avoid the troublesome installation of Xenomai we provide and already patched Ubuntu 16.04 distribution. It can be downloaded from <http://arantxa.ii.uam.es/~gnb/rtubuntu-16.04.4-1.0-desktop-amd64.iso>.

If you install Xenomai 3 this way you will have to activate *Analogy* and *RTIPC* drivers **every time you turn on the computer**. For *RTIPC* type in a terminal:

```
sudo insmod /lib/modules/4.9.90/kernel/drivers/xenomai/ipc/xeno_rtipc.ko
```

And for *Analogy*:

```
sudo analogy_config analogy0 analogy_ni_pcimio
```

Note that *analog0* refers to the DAQ board you want to configure, in this case the first one, but in case that you have more than one you can use *analog1*, *analog2*, etc. *analog_ni_pcimio* is the name of the driver, in this case National Instruments PCIMIO, but you can use any driver supported by Comedi (<http://www.comedi.org/hardware.html>).

2.2 Comedi DAQ drivers

Note: Xenomai already includes their own DAQ drivers, called *Analogy*, so these steps are not necessary.

The Comedi³ project develops open-source drivers, tools, and libraries for data acquisition. Installation instructions can be found in the official Github page (<https://github.com/Linux-Comedi/comedi/blob/master/INSTALL>), or the following instructions can be entered in a terminal:

For 64 bits OS:

```
sudo apt install git build-essential dkms automake linux-headers-rt-amd64
```

For 32 bits OS:

```
sudo apt install git build-essential dkms automake linux-headers-rt-686-pae
```

For both:

```
git clone https://github.com/Linux-Comedi/comedi.git
cd comedi/
./autogen.sh
cd ..
sudo dkms add ./comedi
sudo dkms install comedi/0.7.76.1+20170502git-1.nodist
sudo depmod -a
sudo apt install libcomedi0 libcomedi-dev
```

The list of Comedi supported hardware can be consulted at <http://www.comedi.org/hardware.html>.

2.3 Other libraries required by RTHybrid

2.3.1 Qt5

RTHybrid graphical user interface has been designed using the open-source framework Qt5. To download and install it from command-line just enter in a terminal:

```
sudo apt install qt5-default
sudo apt install qtmultimedia5-dev
```

³ More information on Comedi: <http://www.comedi.org/>

2.3.2 libxml2

Libxml2 is a free XML C parser and toolkit used by RTHybrid to read and write data from this kind of files. It can be easily installed from command-line by typing:

```
sudo apt install libxml2
```

3 RTHybrid installation

Once all the above components have been configured, installing RTHybrid is an easy task. Choose the directory where you want to download and install the program, open a terminal there and type:

```
git clone https://github.com/GNB-UAM/RTHybrid.git  
cd RTHybrid  
sudo sh compile.sh
```

And that should be it!

Part II

How to use it

4 Launching the program

From the RTHybrid directory, enter in a terminal the command

```
sudo ./RTHybrid
```

to launch the program with graphical user interface. On the other hand, if you prefer to launch the program without GUI, loading the experiment configuration from an XML file, enter

```
sudo ./RTHybrid --xml <xml configuration file path>
```

Several experiments can be sequentially executed automatically with the no-GUI mode using scripts such as the following:

```
#!/bin/bash
sudo ./RTHybrid --xml experiment1_conf.xml
sudo ./RTHybrid --xml experiment2_conf.xml
sudo ./RTHybrid --xml experiment3_conf.xml
```

Details on how to use both versions is provided in the following sections.

5 Graphical user interface

Figure 1 represents RTHybrid graphical user interface. It contains the following elements:

1. **Neuron model selector:** dropdown menu to select the neuron model. The *Configure* button opens a pop-up window to select the parameters of the model. More information on available models can be found in Annex A.
2. **Living neuron to neuron model synapse model selector:** dropdown menu to select the synapse model. The *Configure* button opens a pop-up window to select the parameters of the model. More information on available models can be found in Annex A.
3. **Neuron model to living neuron synapse model selector:** dropdown menu to select the synapse model. The *Configure* button opens a pop-up window to select the parameters of the model. More information on available models can be found in Annex A.
4. **Dynamic hybrid circuit representation:** picture of the circuit selected in the previous options.
5. **Exploration algorithms selector:** this feature is currently not available, but will be soon!
6. **Experiment temporal settings:**
 - (a) **Frequency:** program working frequency (read/write from DAQ, generate model points, etc). *In Hertz*.
 - (b) **Observation:** duration of the initial observation time. During this time the program only records the living neuron signal and calculates its amplitude and characteristic temporal scale. Data is not stored. *In seconds*.
 - (c) **Duration:** duration of the experiments itself. During this time the hybrid interaction takes place. *In seconds*.

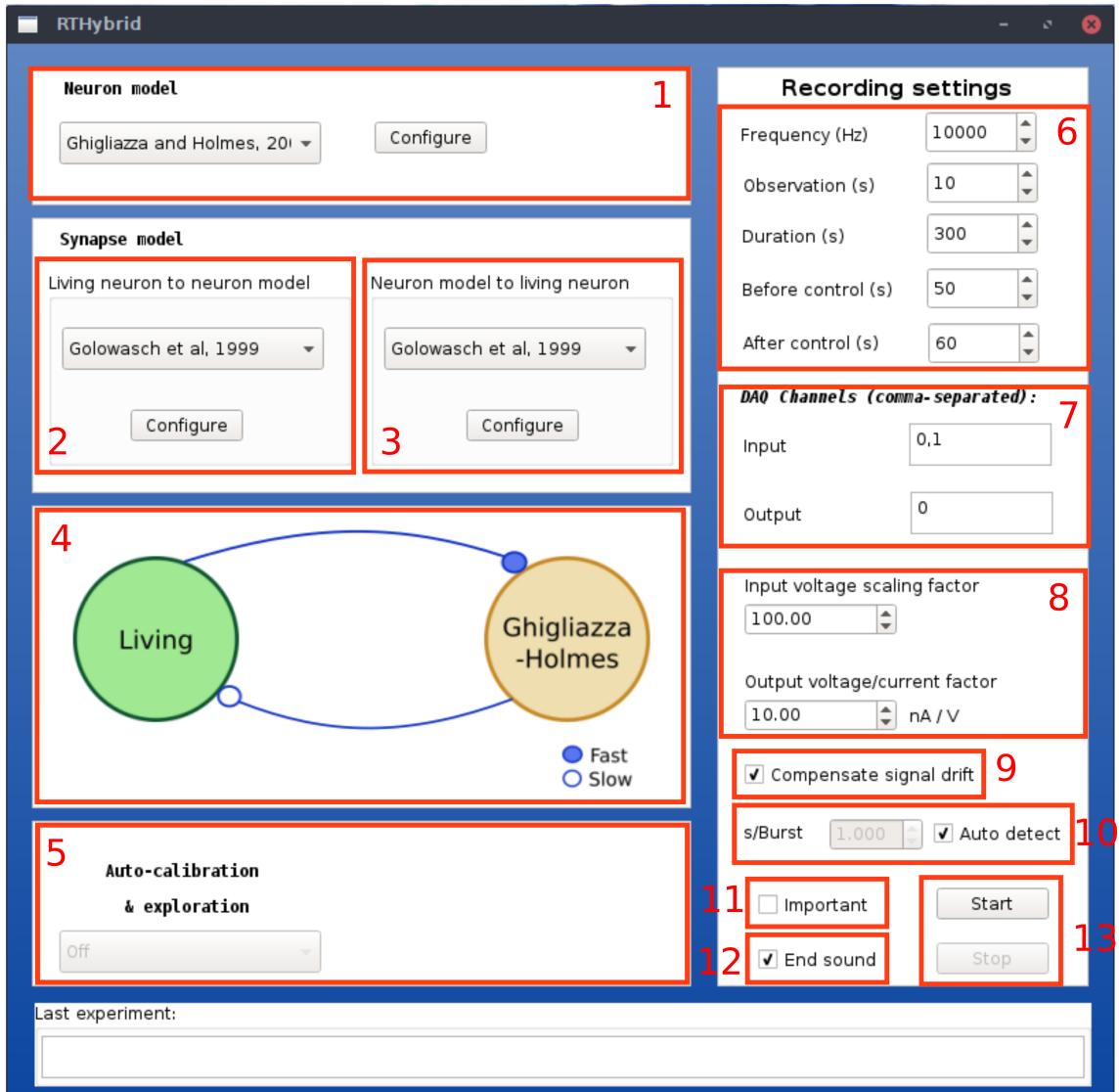


Figure 1: Illustration of RTHybrid graphical user interface.

- (d) **Control recording duration (before the hybrid experiment):** duration of the control time before the experiment. During this time both neurons are running but there is no interaction. *In seconds*.
- (e) **Control recording duration (after the hybrid experiment):** duration of the control time after the experiment. During this time both neurons are running but there is no interaction. *In seconds*.
- 7. **DAQ channels:** list of the input and output DAQ channels, separated by commas. RTHybrid always uses the **first channels from Input and Output lists** to read the living neuron membrane potential and write the calculated current, respectively. For example, if input channels list "4,0,1" is provided, the program will read the potential of the living neuron from channel 4, but also record from channels 0 and 1 and store the data. The second Output channel will be used to send the model membrane potential.
- 8. **Input/output factors:** experimental setups (amplifiers, etc) often scale the signals. These values can be used to compensate these effects, so the program can work with the real values from the living neuron.

- (a) **Input voltage scaling:** scaling factor applied by your amplifier to the real membrane potential recorded.
 - (b) **Output voltage/current:** the program sends the calculated synaptic current to the living neuron through a voltage DAQ channel and, therefore, it is in volts. In this field you should write the conversion factor from volts to nA applied by your equipment (10 nA/V for example, if per each volt received it produces 10 nA).
9. **Compensate signal drift:** RTHybrid adjusts the amplitude of the neuron model to the living neuron membrane potential value. If during the experiment this signal drifts and its minimum and maximum values change significantly the interaction will be affected. Check this box if you want the program to apply a compensation algorithm if there is signal drift, so both neurons keep working in the same voltage interval.
 10. **Seconds per burst:** if the box is checked, the neuron model will produce bursts of the same duration as the living neuron. If not, each burst will be as long as the *seconds* specified in the spin box.
 11. **Important:** check this box to mark the experiment as important of highly relevant in the log file. This is useful to tag specific experiments in a row.
 12. **End sound:** check this box if you want an alarm to sound when the experiment is finished.
 13. **Start/Stop:** the Start button launches an experiment with the parameters specified in all the previous fields. The Stop button allows to safely interrupt an on-going experiment (in Xenomai this button does not work and the only option to end the experiment manually is to close RTHybrid).

6 Command-line interface

Note: Some example configuration files are provided along with RTHybrid. They can be found at the xml folder.

The command line interface allows to run a set of experiments in an predefined automatic manner. The XML files required for command-line launch need to follow an specific structure, defined by RTHybrid. The following Listings provide an example of such files.

<pre> <clamp> <neuron type="4"> <vars> <v val="30.24"/> </vars> <params> <gca val="4.4"/> <eca val="120"/> <vthca val="-1.2"/> <kca val="0.11"/> <gk val="9"/> <ek val="-80"/> <vthk val="2.0"/> <kk val="0.2"/> <gks val="0.25"/> <vthks val="-27"/> <kke val="0.8"/> <gl val="2.0"/> <el val="-60"/> <c val="1.2"/> <epsilon val="4.9"/> <delta val="0.052"/> <i val="35.6"/> <method val="3"/> </params> </neuron> <freq val="10000"/> <times> <before val="5"/> <duration val="10"/> <after val="5"/> <observation val="10"/> </times> </pre>	<pre> <synapse_ltom type="2"> <g_fast val="0.8"/> <vth_fast val="25.0"/> <s_fast val="5.0"/> <g_slow val="0.0"/> <vth_slow val="0.0"/> <s_slow val="0.0"/> <k1_slow val="14.0"/> <k2_slow val="4.0"/> </synapse_ltom> <synapse_mtol type="2"> <g_fast val="0.0"/> <vth_fast val="0.0"/> <s_fast val="0.0"/> <g_slow val="0.08"/> <vth_slow val="15.0"/> <s_slow val="1.0"/> <k1_slow val="14.0"/> <k2_slow val="8.0"/> </synapse_mtol> <calib type="0"></calib> <output_channels val="0,1"/> <input_channels val="0"/> <input_factor val="1000"/> <output_factor val="1"/> <imp val="0"/> <drift val="1"/> <sec_per_burst val="0.33"/> </pre>
--	---

The tags to be used are the following:

- **<clamp> </clamp>** indicate that this file is an RTHybrid configuration one. All the other tags MUST be placed inside them.
- **<neuron type="NUMBER"> </neuron>** indicate that the neuron model with index NUMBER will be used. Additional characteristics of the models are added using tags inside of these. More information about the included neuron models and how to configure them is available in Annex A.
 - **<vars> </vars>** these tags are used inside the model ones to provide the initial values of the variables of the model.
 - **<params> </params>** these tags are used inside the model ones to provide the values of the model parameters.

- **<synapse_ltom type="NUMBER"> </synapse_ltom>** indicate that the synapse model with index NUMBER will be used in the connection from the living neuron to the neuron model. Additional characteristics of the models are added using tags inside of these. More information about the included synapse models and how to configure them is available in Annex A.
- **<synapse_mtol type="NUMBER"> </synapse_mtol>** indicate that the synapse model with index NUMBER will be used in the connection from the living neuron to the neuron model. Additional characteristics of the models are added using tags inside of these. More information about the included synapse models and how to configure them is available in Annex A.
- **<calib type="NUMBER"> </calib>** this feature is currently not available and the only admitted value is 0.
- **<freq val="NUMBER"/>** program working frequency (read/write from DAQ, generate model points, etc). NUMBER must be *in Hertz*.
- **<imp val="NUMBER"/>** set NUMBER to 1 to mark the experiment as important in the log file, else as 0.
- **<times> </times>** inside these tags the temporal properties are defined by using the tags:
 - **<observation val="NUMBER"/>** duration of the initial observation time. During this time, the program only records the living neuron signal and calculates its amplitude and period. Data is not stored. NUMBER must be *in seconds*.
 - **<duration val="NUMBER"/>** duration of the experiments itself. During this time the hybrid interaction takes place. NUMBER must be *in seconds*.
 - **<before val="NUMBER"/>** duration of the control time before the experiment. During this time both neurons are running but there is no interaction. NUMBER must be *in seconds*.
 - **<after val="NUMBER"/>** duration of the control time after the experiment. During this time both neurons are running but there is no interaction. NUMBER must be *in seconds*.
- **<input_channels val="LIST"/>** LIST of the input DAQ channels, separated by commas. RTHybrid always uses the **first channel from Input list** to read the living neuron membrane potential. For example, if input channels list "4,0,1" is provided, the program will read the potential of the living neuron from channel 4.
- **<output_channels val="LIST"/>** LIST of the output DAQ channels, separated by commas. RTHybrid always uses the **first channel from Output list** to write the calculated current. The second Output channel will be used to send the model membrane potential. For example, if output channels list "4,0,1" is provided, the program will write the synaptic current to channel 4, but also send the membrane potential of the model through channel 0.
- **<input_factor val="NUMBER"/>** experimental setups (amplifiers, etc) often scale or converts the signals. This value can be used to compensate these effects, so the program can work with the real values from the living neuron. NUMBER is the scaling factor applied by your amplifier to the real membrane potential recorded.
- **<output_factor val="NUMBER"/>** experimental setups (amplifiers, etc) often scale or converts the signals. This value can be used to compensate these effects, so the program can work with the real values from the living neuron. The program sends the calculated synaptic current to the living neuron through a voltage DAQ channel and, therefore, it is in volts. NUMBER is the conversion factor from volts to nA applied by your equipment (10 nA/V for example, if per each volt received it produces 10 nA).

- <**drift val="NUMBER"**> RTHybrid adjusts the amplitude of the neuron model to the living neuron membrane potential value. If during the experiment this signal drifts and its minimum and maximum values change significantly the interaction will be affected. Set NUMBER to 1 if you want the program to apply a compensation algorithm if there is signal drift, so both neurons keep working in the same voltage interval. Else NUMBER must be set to 0.
- <**sec_per_burst val="NUMBER"**> if NUMBER is -1, the neuron model will produce bursts of the same duration as the living neuron. If not, each burst will be as long as the *seconds* specified by NUMBER.

7 Data storage

The data for each experiment will be stored in a separate file inside the *data* folder. Currently, files are stored in ASCII. A new folder will be created for each day, named as the year, month and day of the experiment (i.e. 2018y_7m_2d is the folder for the 2nd of July, 2018). Inside those directories, the specific files for each experiment are named using the hour, minute and second when it was launched (i.e. 16h_26m_21s_data.txt refers to an experiment run at 16:26:21).

#Input channels	#Output channels	Living neuron voltage			Current from living to model	
1	1	Latency	Model voltage	Time	Current from model to living	
2	0.111	11015	-5.847	0.000	-0.000	-0.000
3	0.209	9359	-5.846	3.973	-0.000	-0.000
4	0.309	8776	-5.846	-4.012	-0.000	-0.000
5	0.409	8563	-5.845	-4.009	-0.000	-0.000
6	0.509	8512	-5.845	-3.967	-0.000	-0.000
7	0.608	8485	-5.844	-3.994	-0.000	-0.000

Figure 2: Example of a data file generated by RTHybrid.

The first line of each data file indicates the number of Input and Output channels. From the second line, each row is the data of an interval of the experiment. The columns of each row contain the following data:

1. Elapsed time since the beginning of the experiment. *In milliseconds*.
2. Real-time latency on that interval (difference between the true time when that interval started against the expected time). *In nanoseconds*.
3. Membrane potential of the neuron model, scaled to the units of the living one. *In volts*.
4. Membrane potential of the living neuron. *In volts*.
5. Synaptic current from the neuron model to the living one. *In nA*.
6. Synaptic current from the living neuron to the model one. *In nA*.

Part III

Model library

8 Integration methods

The model library implements different integration methods that can be used to solve differential equations from the models. In configuration XML files it can be selected using the tag *method* and the following IDs:

- **Euler:** 0
- **Heun:** 1
- **Runge-Kutta order 4:** 2
- **Runge-Kutta order 6(5):** 3

9 Neuron models

9.1 None

Its numerical ID is 0. Selecting this option is equivalent to not select any model.

9.2 Izhikevich, 2003

Model taken from the publication "Simple model of spiking neurons" by Izhikevich in 2003. It has two variables (x and y), four parameters (a , b , c and d) and the external current input (i). *method* refers to be integration method to be used, but only the Runge-Kutta 6(5) method is available for this model at the moment.

Its numerical ID is 1. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_iz1.xml*).

```
<neuron type="1">
    <vars>
        <x val="30.24"/>
        <y val="-5.5"/>
    </vars>
    <params>
        <a val="0.02"/>
        <b val="0.2"/>
        <c val="-50.0"/>
        <d val="2.0"/>
        <i val="10.0"/>
    </params>
    <method val="3"/>
</neuron>
```

9.3 Hindmarsh and Rose, 1984

Model taken from the publication "A model of neuronal bursting using three coupled first order differential equations" by Hindmarsh and Rose in 1984. It has three variables (x , y and z), two

parameters (r and s) and the external current input (i). *method* refers to be integration method to be used, but only the Runge-Kutta 6(5) method is available for this model at the moment.

Its numerical ID is 2. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_hr1.xml*).

```
<neuron type="2">
    <vars>
        <x val="-0.712841"/>
        <y val="-1.936880"/>
        <z val="3.165680"/>
    </vars>
    <params>
        <r val="0.0021"/>
        <s val="4.0"/>
        <i val="3.0"/>
        <method val="3"/>
    </params>
</neuron>
```

9.4 Rulkov, 2002

Model taken from the publication "Modeling of spiking-bursting neural behavior using two-dimensional map" by Rulkov in 2002. It has two variables (x and y), three parameters (*alpha*, *sigma* and *mu*) and the external current input (i).

Its numerical ID is 3. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_rlk1.xml*).

```
<neuron type="3">
    <vars>
        <x val="-1.701750"/>
        <y val="-4.919230"/>
    </vars>
    <params>
        <alpha val="6.0"/>
        <sigma val="0.1"/>
        <mu val="0.001"/>
        <i val="1.0"/>
    </params>
</neuron>
```

9.5 Ghigliazza and Holmes, 2004

Model taken from the publication "Minimal Models of Bursting Neurons: How Multiple Currents, Conductances, and Timescales Affect Bifurcation Diagrams" by Ghigliazza and Holmes in 2004. It has one variable (x) and sixteen parameters, related to its four different currents: I_{Ca} (*gca*, *eca*, *vthca*, *kca*), I_K (*gk*, *ek*, *vthk*, *kk*), I_{KS} (*gks*, *vthks*, *kke*) and I_L (*gl*, *el*), plus the external current input (i), the capacitance (c), and parameters δ (*delta*) and ϵ (*epsilon*). *method* refers to be integration method to be used, but only the Runge-Kutta 6(5) method is available for this model at the moment.

Its numerical ID is 4. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_gh1.xml*).

```
<neuron type="4">
    <vars>
        <x val="30.24"/>
    </vars>
    <params>
        <gca val="4.4"/>
        <eca val="120"/>
    </params>
</neuron>
```

```

<vthca val="-1.2"/>
<kca val="0.11"/>

<gk val="9"/>
<ek val="-80"/>
<vthk val="2.0"/>
<kk val="0.2"/>

<gks val="0.25"/>
<vthks val="-27"/>
<kks val="0.8"/>

<gl val="2.0"/>
<el val="-60"/>

<c val="1.2"/>
<epsilon val="4.9"/>
<delta val="0.052"/>
<i val="35.6"/>

<method val="3"/>
</params>
</neuron>

```

9.6 Wang, 1993

Model taken from the publication "Ionic basis for intrinsic 40 Hz neuronal oscillations" by Wang in 1993. It has one variable (v) and fourteen parameters, related to its five different ionic currents: I_{Na} and I_{Nap} (g_{na} , g_{nap} , v_{na}), I_K and I_{Ks} (g_k , g_{ks} , v_k) and I_L (g_l , v_l), plus the external current input (i), the capacitance (cm), and parameters σ ($sigma$), ρ (rho), ϕ (phi) and τ_m ($taum$). $method$ refers to be integration method to be used, but only the Runge-Kutta 6(5) method is available for this model at the moment.

Its numerical ID is 5. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_wang1.xml*).

```

<neuron type="5">
  <vars>
    <v val="-66.5"/>
  </vars>
  <params>
    <g_na val="52.0"/>
    <v_na val="55.0"/>
    <g_nap val="0.1"/>

    <g_k val="20.0"/>
    <v_k val="-90.0"/>
    <g_ks val="14.0"/>

    <g_l val="0.1"/>
    <v_l val="-60.0"/>

    <cm val="1.0"/>
    <sigma val="0.0"/>
    <rho val="0.6"/>
    <phi val="28.57"/>
    <taum val="6.0"/>
    <i val="2.5"/>

    <method val="3"/>
  </params>
</neuron>

```

9.7 Komendantov and Kononenko, 1996

Model taken from the publication "Deterministic Chaos in Mathematical Model of Pacemaker Activity in Bursting Neurons of Snail, Helix Pomatia" by Komendantov and Kononenko in 1996. It has one variable (V) and sixteen parameters, related to its eight different ionic currents: g_{na} , g_k , g_b , g_{na_v} , V_k , V_b , V_{na} , g_{na_ttx} , g_{ca} , k_{beta} , g_k_{tea} , V_{ca} , β_{eta} , g_{ca_ca} , k_s and ρho , plus the external current input (i) and the capacitance (Cm). *method* refers to be integration method to be used.

Its numerical ID is 6. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_komendantov_kononenko1.xml*).

```
<neuron type="6">
  <vars>
    <V val="-55.0"/>
  </vars>

  <params>
    <i val="0.0"/>
    <Cm val="0.02"/>

    <g_na val="0.0231"/>
    <g_k val="0.25"/>
    <g_b val="0.165"/>
    <g_na_v val="0.11"/>
    <V_k val="-70.0"/>
    <V_b val="-58.0"/>
    <V_na val="40.0"/>
    <g_na_ttx val="400.0"/>
    <g_ca val="1.5"/>
    <k_beta val="15000.0"/>
    <g_k_tea val="10.0"/>
    <V_ca val="150.0"/>
    <beta val="0.00004"/>
    <g_ca_ca val="0.02"/>
    <k_s val="50.0"/>
    <rho val="0.002"/>

    <method val="3"/>
  </params>
</neuron>
```

9.8 Nowotny et al, 2008

Model taken from the publication "Probing the Dynamics of Identified Neurons with a Data-Driven Modeling Approach" by Nowotny, Levi and Selverston in 2008. It is a multicompartmental model with several variables and parameters, plus the external current input (i). All names are taken from the original paper. *method* refers to be integration method to be used.

NOTE: since it is an implementation of a multicompartmental model with high computational cost it does not always perform correctly in hard real-time. Usually it seems to have some trouble during the beginning of the experiment and then it works just fine.

Its numerical ID is 7. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_nowotny_et_al_1.xml*).

```
<neuron type="7">
  <vars>
    <V val="-65.675307"/>
      <Vaxon val="-70.600978"/>
      <mna val="0.004480"/>
      <hna val="0.999018"/>
      <mcat val="0.000299"/>
      <hcat val="0.497247"/>
      <mcas val="0.0"/>
  </vars>
</neuron>
```

```

<mkca val="0.0"/>
<hkca val="0.891752"/>
<mkd val="0.014577"/>
<ma val="0.113476"/>
<ha1 val="0.614612"/>
<ha2 val="0.133255"/>
<mh val="0.156078"/>
<Ca val="0.04"/>
<mm val="0.017216"/>
</vars>

<params>
  <i val="-6"/>
    <ioffset val="-2"/>
    <iscale val="0.01222505828"/>

    <Cma val="0.1148312178"/>
    <Cms val="0.0543909161"/>
    <AreaAxon val="1"/>
    <AreaSoma val="0.9"/>
    <gvv val="0.004017089073"/>

    <gna val="7.15"/>
    <Vna val="50"/>

    <cica val="0.05049639866"/>
    <RTF val="11.4866"/>
    <Pca val="0.00155982"/>
    <Caout val="15123.5"/>
    <Ca0 val="0.02"/>
    <kca val="0.01729930764"/>

    <gcat val="0.1247796343"/>
    <kmcat val="0.045"/>
    <khcat val="0.02"/>
    <Vmcat val="15"/>
    <Vhcat val="-40"/>
    <Smcat val="-9.8"/>
    <Shcat val="3.2"/>
    <Vkhcat val="-15"/>
    <Skhcat val="-10"/>

    <gcas val="0.07142141077"/>
    <kmcas val="0.0608644"/>
    <Vmcas val="29.1336"/>
    <Smcas val="-4.43083"/>

    <gkca val="1.663612026"/>
    <Vkca val="-72"/>
    <kmkca val="0.6"/>
    <khkca val="0.035"/>
    <Vmkca1 val="0"/>
    <Vmkca2 val="-16"/>
    <Smkca1 val="-23"/>
    <Smkca2 val="-5"/>
    <f val="0.6"/>
    <cmkca val="2.5"/>
    <chkcal val="0.7"/>
    <chkca2 val="0.6"/>

    <Vkd val="-72"/>
    <gkd val="1.43"/>

    <Va val="-72"/>
    <ga val="0.8008791415"/>
    <kma val="0.14"/>
    <kha1 val="0.05"/>
    <kha2 val="0.0036"/>
    <Vma val="-12"/>
    <Vha val="-62"/>

```

```

<Vkha2 val="-40"/>
<Vaa val="7"/>
<Sma val="-26"/>
<Sha val="6"/>
<Skha2 val="-12"/>
<Saa val="-15"/>

<gh val="0.0126874385"/>
<Vh val="-20"/>
<kmh val="0.00033"/>
<Vmh val="-70"/>
<Vkmh val="-110"/>
<Smh val="8"/>
<Skmh val="-21.6"/>

<gls val="0.0006901584324"/>
<gla val="0.001004288227"/>
<Vl val="-50"/>

<Vm val="-80"/>
<gm val="0.2613041227"/>
<kmm val="0.0001386796618"/>
<Vmnm val=" -26.9916223"/>
<Smm val="-5.957441281"/>
<Vkmnm val="-60.57875745"/>
<Skmnm val="-13.25674673"/>

    <method      val="3"/>
  </params>
</neuron>
```

10 Synapse models

10.1 None

Its numerical ID is 0. Selecting this option is equivalent to not select any model.

10.2 Electrical

Simple electrical synapse model following the equation $I = g * (V_{post} - V_{pre})$, where g is the synapse conductance (in μS), V_{post} the post-synaptic neuron potential (in mV) and V_{pre} the pre-synaptic neuron potential (in mV). The resulting current is in nA .

Conductance (g) can be defined by the user. Moreover, the neurons can be set to fire in phase or antiphase by setting that parameter (*antiphase*) to 0 or 1, respectively.

Its numerical ID is 1. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_rlk1.xml*).

```

<synapse_ltom type="1">
  <g val="0.2"/>
  <antiphase val="1"/>
</synapse_ltom>

<synapse_mtol type="1">
  <g val="0.2"/>
  <antiphase val="1"/>
</synapse_mtol>
```

10.3 Golowasch et al, 1999

Model taken from the publication "Network Stability from Activity-Dependent Regulation of Neuronal Conductances" by Golowash, Casey, Abbott and Marder in 1999. This graded chemical synapse model includes two different dynamics, one fast and one slow. All the parameters for each kind of dynamic, in both ways, can be adjusted and modified by the user, including the conductances, voltage thresholds (v_{th} and s) and kinetic parameters (k_1 , k_2). v_{th} and s are expressed as a percentage of the total amplitude of the membrane potentials. For example, if the membrane potential goes from -60 to 40 mV, then the total range is 100 mV and if the user sets v_{th} to 25% then the effective threshold will be -35 mV ($-60 + (100 * 0.25) = -35$). In the case of the s , if it is set to, for example, 5% then the effective value will be 5 mV (five percent of the total range). The resulting current is in nA .

Its numerical ID is 2. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_gh1.xml*).

```
<synapse_ltom type="2">
  <g_fast val="0.8"/>
  <vth_fast val="25.0"/>
  <s_fast val="5.0"/>

  <g_slow val="0.0"/>
  <vth_slow val="0.0"/>
  <s_slow val="0.0"/>
  <k1_slow val="0.0"/>
  <k2_slow val="0.0"/>
</synapse_ltom>

<synapse_mtol type="2">
  <g_fast val="0.0"/>
  <vth_fast val="0.0"/>
  <s_fast val="0.0"/>

  <g_slow val="0.08"/>
  <vth_slow val="15.0"/>
  <s_slow val="1.0"/>
  <k1_slow val="1.0"/>
  <k2_slow val="0.03"/>
</synapse_mtol>
```

10.4 Destexhe et al, 1994

Chemical synapse model from "An efficient method for computing synaptic conductances based on a kinetic model of receptor binding" by Destexhe, Mainen and Sejnowski in 1994. Conductance (g) is in μS and synaptic pulse duration (*pulse_duration*) in ms . Frequency (*freq*) is in Hz and MUST be the same as RTHybrid sampling frequency. Reversal potential e_{syn} is expressed as a percentage of the total amplitude of the membrane potentials. For example, if the membrane potential goes from -60 to 40 mV, then the total range is 100 mV and if the user sets e_{syn} to -15% then the effective threshold will be -85 mV ($-60 + (100 * -0.15) = -85$). The resulting current is in nA .

Its numerical ID is 3. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_destexhe.xml*).

```
<synapse_ltom type="3">
  <g val="1.5"/>
  <e_syn val="-15.0"/>

  <alpha val="2.0"/>
  <beta val="1.0"/>

  <tmax val="1.0"/>
  <pulse_duration val="0.001"/>
```

```

<freq val="10000.0"/>
</synapse_ltom>

<synapse_mtol type="3">
  <g val="1.5"/>
  <e_syn val="-15.0"/>

  <alpha val="2.0"/>
  <beta val="1.0"/>

  <tmax val="1.0"/>
  <pulse_duration val="0.001"/>
  <freq val="10000.0"/>
</synapse_mtol>
```

10.5 Greenberg and Manor, 2005

Graded chemical synapse model from "Synaptic Depression in Conjunction With A-Current Channels Promote Phase Constancy in a Rhythmic Network" by Greenberg and Manor in 2005. Reversal potential e_{syn} , midpoint voltages (v_{12}) and slopes (k) are expressed as a percentage of the total amplitude of the membrane potentials. For example, if the membrane potential goes from -60 to 40 mV, then the total range is 100 mV and if the user sets e_{syn} to -15% then the effective threshold will be -85 mV ($-60 + (100 * -0.15) = -85$), and the same applies for midpoint voltages. In the case of k , if it is set to, for example, 5% then the effective value will be 5 mV (five percent of the total range). The resulting current is in nA .

Its numerical ID is 4. The following is an example of its declaration in a configuration XML file (complete example can be found at *xml/test_greenberg_manor.xml*).

```

<synapse_ltom type="4">
  <g val="0.5"/>
  <e_syn val="-5.0"/>

  <p val="1.0"/>
  <v12_m val="25.0"/>
  <k_m val="10.0"/>
  <tau_hi_m val="0.01"/>
  <tau_lo_m val="0.01"/>

  <q val="1.0"/>
  <v12_h val="18.0"/>
  <k_h val="2.5"/>
  <tau_hi_h val="1900.0"/>
  <tau_lo_h val="300.0"/>
</synapse_ltom>
```