RTHybrid

v1.0

USER MANUAL

Grupo de Neurocomputación Biológica
(GNB)

Universidad Autónoma de Madrid

04.07.2018

Summary

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. bioRxiv 426643; 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. bioRxiv 419622; doi: **https://doi.org/10.1101/419622**.

Further description of the algorithms is provided in these papers.

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-RT [1] and Xenomai 3 [2], 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/preemptrt_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.denx.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 https://mega.nz/#!CPwCRIJQ!jWVGb08wjSP-02-kiv7KK_bKzcaPERWBp7MjH6coXVs.

2.2 Comedi DAQ drivers

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

The Comedi [3] 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
```

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] More information on Comedi: http://www.comedi.org/

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>
```

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

5 Graphical user interface

Figure 1: Illustration of RTHybrid 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. More information on available models can be found in Annex A.
2. **Neuron model parameters:** menu with the parameters of the selected neuron model. More information on the parameters of each model can be found in Annex A.
3. **Synapse model selector:** dropdown menu to select the synapse model. More information on available models can be found in Annex A.

4. **Synapse model parameters:** menu with the parameters of the selected synapse model. More information on the parameters of each model can be found in Annex A.
5. **Exploration algorithms selector:** this feature is currently not available, but will be soon!
6. **Dynamic hybrid circuit representation:** picture of the circuit selected in the previous options.
7. **Experiment temporal settings:**
   (a) **Frequency:** program working frequency (read/write from DAQ, generate model points, etc). *In Hertzs.*
   (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.*
   (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.*
8. **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.
9. **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).
10. **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.
11. **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.
12. **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.
13. **End sound:** check this box if you want an alarm to sound when the experiment is finished.

14. **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 predesigned automatic manner. The XML files required for command-line launch need to follow an specific structure, defined by RTHybrid. Figure 6 shows an example configuration file. The tags that have 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.

```
<clamp>

    <model 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>
    </model>

    <synapse type="1">
        <g_real_to_virtual val="0.4"/>
        <g_virtual_to_real val="0.4"/>
        <antiphase val="1"/>
    </synapse>

    <calib type="0"></calib>

    <freq val="20000"/>

    <times>
        <observation val="10"/>
        <before val="5"/>
        <duration val="300"/>
        <after val="5"/>
    </times>

    <output_channels val="0,1"/>

    <input_channels val="0"/>

    <input_factor val="1"/>

    <output_factor val="1"/>

    <imp val="0"/>

    <drift val="0"/>

    <sec_per_burst val="0.25"/>

</clamp>
```

- <**model type="NUMBER"**> </**model**> 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 type="NUMBER"**> </**synapse**> indicate that the synapse model with index NUMBER will be used. 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 Hertzs*.
- <**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. These values 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. These values 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).

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.*

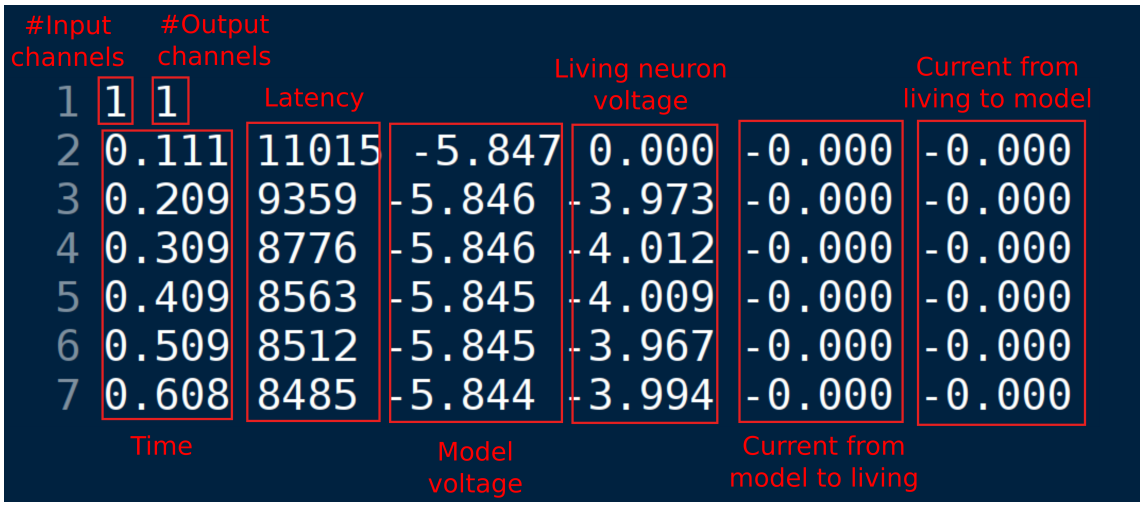

Figure 2: Example of a data file generated by RTHybrid.

Part III

Annex A: Model library

8 Neuron models

8.1 None

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

8.2 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).

Its numerical ID is 1. The following is an example of its declaration in a configuration XML file (complete example can be found at $xml/test\_iz1.xml$).

```
<model 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>
</model>
```

8.3 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).

Its numerical ID is 2. The following is an example of its declaration in a configuration XML file (complete example can be found at $xml/test\_hr1.xml$).

```
<model 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"/>
        </params>
</model>
```

8.4 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 $xml/test\_rlk1.xml$).

```
<model 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>
</model>
```

8.5 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, $vthks$, kks) and I_L (gl, el), plus the external current input (i), the capacitance (c), and parameters δ ($delta$) and ϵ ($epsilon$).

Its numerical ID is 4. The following is an example of its declaration in a configuration XML file (complete example can be found at $xml/test\_gh1.xml$).

```
<model type="4">
        <vars>
                <x 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"/>
                <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"/>
        </params>
</model>
```

9 Synapse models

9.1 None

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

9.2 Electrical

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

Conductance from the real to the model neuron ($g\_real\_to\_virtual$) and from model to real neuron ($g\_virtual\_to\_real$) 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, respetively.

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 type="1">
        <g_real_to_virtual val="0.2"/>
        <g_virtual_to_real val="0.2"/>
        <antiphase val="1"/>
</synapse>
```

9.3 Golowasch et al, 1999

Model taken from the publication "Network Stability from Activity-Dependent Regulation of Neuronal Conductances" by Golowash et al. in 1999. This graded chemical synapse model includes to 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 kinetic parameters (k_1, k_2).

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 type="2">
        <g_real_to_virtual_fast val="0.01"/>
        <vfast_real_to_virtual val="25.0"/>

        <g_real_to_virtual_slow val="0.0"/>
        <vslow_real_to_virtual val="0.0"/>
        <k1_real_to_virtual val="14.0"/>
        <k2_real_to_virtual val="4.0"/>

        <g_virtual_to_real_fast val="0.0"/>
        <vfast_virtual_to_real val="0.0"/>

        <g_virtual_to_real_slow val="0.002"/>
        <vslow_virtual_to_real val="15.0"/>
        <k1_virtual_to_real val="1.4"/>
        <k2_virtual_to_real val="0.4"/>
</synapse>
```