RELACS — a modular software platform for closed-loop experiments

Jan Benda
Biozentrum Martinsried
Ludwig-Maximilians Universität München
Content

Closed-loop experiments

Dynamic clamp

RELACS

Metadata
Closed-loop experiments
“Traditional” experiments

1. A set of stimuli and a more or less fixed experimental protocol are prepared
“Traditional” experiments

1. A set of stimuli and a more or less fixed experimental protocol are prepared

2. The recordings are done on a few cells (> one week of experimental work)
   - the raw voltage trace is the only feedback
   - changes during the running experiment often involve manual manipulations
     ⇒ precious recording time is wasted
“Traditional” experiments

1. A set of stimuli and a more or less fixed experimental protocol are prepared

2. The recordings are done on a few cells (> one week of experimental work)
   • the raw voltage trace is the only feedback
   • changes during the running experiment often involve manual manipulations
     ⇒ precious recording time is wasted

3. The data are analyzed offline
“Traditional” experiments

1. A set of stimuli and a more or less fixed experimental protocol are prepared

2. The recordings are done on a few cells (> one week of experimental work)
   - the raw voltage trace is the only feedback
   - changes during the running experiment often involve manual manipulations
     ⇒ precious recording time is wasted

3. The data are analyzed offline

4. The stimuli and the protocol are modified
“Traditional” experiments

1. A set of stimuli and a more or less fixed experimental protocol are prepared

2. The recordings are done on a few cells (> one week of experimental work)
   - the raw voltage trace is the only feedback
   - changes during the running experiment often involve manual manipulations
     ⇒ precious recording time is wasted

3. The data are analyzed offline

4. The stimuli and the protocol are modified

5. A new set of recordings is made
“Traditional” experiments

1. A set of stimuli and a more or less fixed experimental protocol are prepared

2. The recordings are done on a few cells (> one week of experimental work)
   - the raw voltage trace is the only feedback
   - changes during the running experiment often involve manual manipulations
     ⇒ precious recording time is wasted

3. The data are analyzed offline

4. The stimuli and the protocol are modified

5. A new set of recordings is made

6. After several iterations a paper is written
Closed-loop experiments

1. Present a stimulus
2. Record the response
3. Immediately analyze and visualize the data
4. Generate the next stimulus
Simple closed-loop experiments

- Online visualization of processed data:
  - General infos, e.g. quality of spike detection, sensitivity of the cell, temperature, condition of animal, ...
  - Specific results, e.g. spike raster, firing rates, spike-triggered averages, ...

⇒ Speeds up manual (“traditional”) closed-loop
Simple closed-loop experiments

- Online visualization of processed data:
  - General infos, e.g. quality of spike detection, sensitivity of the cell, temperature, condition of animal, ...
  - Specific results, e.g. spike raster, firing rates, spike-triggered averages, ...

⇒ Speeds up manual ("traditional") closed-loop

- Set stimuli relative to the neuron’s dynamic range
- Automatically control motorized electrodes (great for dual unit recordings!)
- Optimize tuning curve measurements
- ...
Example: tuning curve measurement

Traditional:

![Graph showing firing rate vs. stimulus intensity](image-url)
Example: tuning curve measurement

Traditional:

![Graph showing firing rate vs. stimulus intensity with n=5 and ΔI=8]

either:
fast $\rightarrow$ low resolution
Example: tuning curve measurement

Traditional:

![Graph showing a tuning curve with firing rate on the y-axis and stimulus intensity on the x-axis. The graph includes data points for n=17 and ΔI=2.]

either:
fast → low resolution
or:
high resolution → slow
Example: tuning curve measurement

Traditional:

![Graph showing firing rate vs. stimulus intensity for traditional method with n=17 and ∆I=2.]

either:
fast → low resolution

or:
high resolution → slow

Closed loop:

1. start with low resolution

![Graph showing firing rate vs. stimulus intensity for closed loop method with n=5 and ∆I=8.]

Jan Benda, LMU
Example: tuning curve measurement

**Traditional:**

- Firing rate [Hz]
- Stimulus intensity
- 
  - $n=17$
  - $\Delta I=2$

**Closed loop:**

- Firing rate [Hz]
- Stimulus intensity
- 
  - $n=7$
  - $\Delta I=4$

either:

- fast $\rightarrow$ low resolution
- or:
- high resolution $\rightarrow$ slow

1. start with low resolution
2. increase resolution where necessary!
Example: tuning curve measurement

Traditional:

either:
fast $\rightarrow$ low resolution

or:
high resolution $\rightarrow$ slow

Closed loop:

1. start with low resolution
2. increase resolution where necessary!
3. further increase resolution
Example: tuning curve measurement

Traditional:

- either:
  - fast → low resolution
- or:
  - high resolution → slow

Closed loop:

1. start with low resolution
2. increase resolution where necessary!
3. further increase resolution
Advanced closed-loop experiments

New experimental designs are possible:

- Optimal search for a neuron’s receptive field.
- Search for stimuli that drive a neuron in an "optimal" way.
- Find set’s of stimulus parameter that result in the same response (iso-response method).
- ...

Example: optimal stimulus ensembles

Dynamic clamp

- Read voltage
- Compute model: $I = f(V, t)$
- Write current

Diagram of a neuron with inputs and outputs.
Dynamic clamp

Current-clamp, with the current $I$ computed as a function of the measured membrane potential $V$.

Closed-loop at a per sample time scale (tens of kHz).
Artificial conductances

\[ I = g(t) \cdot (V - E) \]

Artificial conductances

\[ I = g(t) \cdot (V - E) \]

Andrew A. Sharp, Michael B. ONeil, L. F. Abbott, & Eve Marder (1993) *J Neurophysiol*

- Synaptic conductances
- Voltage-gated conductances
Artificial networks

\[ I_1 = g_{\text{syn}}(V_2) \cdot (V_1 - E) \]

\[ I_2 = g_{\text{syn}}(V_1) \cdot (V_2 - E) \]


- Artificially couple real neurons
- Couple with simulated neurons
Discontinuous CC and dynamic clamp

- Sampling rate $\leq$ switching frequency / 2
Discontinuous CC and dynamic clamp

- Sampling rate $\leq$ switching frequency / 2

$\Rightarrow$ Synchronize switching and dynamic clamp cycles!

Sampling rate = switching frequency

(in collaboration with R. Polder, npi electronic)
Discontinuous CC and dynamic clamp

- Sampling rate $\leq$ switching frequency/2
  $\Rightarrow$ Synchronize switching and dynamic clamp cycles!
  Sampling rate $=$ switching frequency
  (in collaboration with R. Polder, npi electronic)
- NPI SEC: variable switching frequency $\gg 10$ kHz
  independent of C compensation
relacs

... enjoy your recordings

www.relacs.net
Relaxed Electrophysiological data Acquisition, Control, and Stimulation

RELACS is a framework for closed-loop experiments

Modular design

RELACS core with flexible C++ Plugins for

- hardware abstraction
- data pre-processing (filter, spike detectors)
- experimental protocols
- passive controls
- model
Hardware independent protocols

RELACS integrates all hardware components.

Experimental protocols for RELACS

- are implemented independently of specific hardware
- can be used on all the different experimental setups in your lab without any modifications
- can be shared with other labs
Free and open source software

RELACS is open source and free software distributed under the GNU General Public License (GPL).

- No hassle with licenses of commercial software.
- Add whatever new feature you need directly to the program.
- Share the program and your specific experimental protocols with your collaborators.
- Know what the data-analysis algorithms are doing!
Talking about data
an extensible framework
for metadata exchange
The data-chain

\[ C = \int \log \left( 1 + \frac{S(f)}{N(f)} \right) df \]
The data-chain

\[ C = \int \log \left( 1 + \frac{S(f)}{N(f)} \right) df \]

German neuroinformatics node

www.g-node.de
• All data transfer requires talking about data.
• How to exchange metadata?
Metadata

- is “data about data”.
- describe recording conditions.
- essential for data analysis, management, and sharing.

`stimulusType = white noise`
Metadata

- is “data about data”.
- describe recording conditions.
- essential for data analysis, management, and sharing.

**stimulusType** = **white noise**

```
name  value
```
The metadata problem

- is “data about data”.
- describe recording conditions.
- essential for data analysis, management, and sharing.

**stimulusType** = **white noise**

- What name to choose?
- What does it mean?
- How to organize metadata?
odML — open metadata markup language

Structure:

- Section 1
  - Property
    - name
    - value
    - error (optional)
    - unit (optional)
    - type (optional)
    - description (optional)

 Implemented as the odML XML Schema
Vocabularies: names & definitions

HardwareSettings:

Amplifier:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gain</td>
<td>float</td>
<td>The amplifier gain.</td>
</tr>
<tr>
<td>HighpassCutoff</td>
<td>float</td>
<td>The cutoff frequency of the amplifier’s highpass filter. Given in Hz.</td>
</tr>
<tr>
<td>LowpassCutoff</td>
<td>float</td>
<td>The cutoff frequency of the amplifier’s lowpass filter. Given in Hz.</td>
</tr>
<tr>
<td>Mode</td>
<td>string</td>
<td>The amplifier mode. E.g. Bridge, CC, VC etc.</td>
</tr>
</tbody>
</table>
How to use odML?

1. Assemble properties:
   
   • If you find an appropriate property in the odML-vocabularies, use it!
   
   • Ignore all properties that do not match.
   
   • Add your own properties that are not yet in the vocabulary, if possible with a description.
How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-vocabularies, use it!
   - Ignore all properties that do not match.
   - Add your own properties that are not yet in the vocabulary, if possible with a description.

2. Write them into an odML XML file
How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-vocabularies, use it!
   - Ignore all properties that do not match.
   - Add your own properties that are not yet in the vocabulary, if possible with a description.

2. Write them into an odML XML file

3. Transfer them to an analysis or database program
The data life-cycle

- Meta information tends to vanish with time.
- Thus, re-using of old data is a tedious business.
- Data should be annotated as early as possible (preferentially at the time of acquisition, e.g. with RELACS).
The data life-cycle

The Data Lifecycle

- odML provides a simple and flexible standard
- Well annotated data can be found and reused easily

⇒ Your data deserves it!
Summary

Closed-loop experiments
Novel experimental designs

Dynamic clamp
Artificial conductances and hybrid networks
— R. Polder, npi electronic, Tamm

RELACS www.relacs.net
Software platform for closed-loop and dynamic clamp experiments
— NeurOnline: S. Rotter, M. Ambard, A. Brandt, C. Boucsein, Freiburg

Metadata — odML
A standard for sharing data
— J. Grewe & G-node, LMU Munich
Experimental protocol example

```cpp
int Example::main( void ) {
    // some initialization ...
    OutData signal;
    signal.setTrace( "LeftSpeaker" );
    signal.sineWave( frequency, duration, amplitude );
    SampleDataD rate( 0.0, duration, 0.001 );
    for ( int counter=0; counter<Repeats; counter++ ) {
        write( signal );
        sleep( duration + pause );
       EventData spikes( events( SpikeEvents[0] ),
            events( SpikeEvents[0] ).signalTime(),
            events( SpikeEvents[0] ).signalTime() + duration );
        double meanrate = spikes.rate( 0.3*duration, duration );
        spikes.addRate( rate, counter, GaussKernel( sigma ) );
        P.clear();
        P.plot( rate, 1000.0, Plot::Yellow, 2, Plot::Solid );
        P.draw();
        if ( meanrate < targetrate ) {
            amplitude *= 2.0;
            signal.sineWave( frequency, duration, amplitude );
        }
    }
}
```
RELACS C++ library for data analysis

Data structures (classes, container):

- *Array* — Basic 1-D vector
- *SampleData* — 1-D data vector with regularly sampled time axis
- *Map* — Sequence of $x|y$ data pairs

Algorithms:

- basic statistics (moments, quartiles, histogram)
- power spectra, coherence, transfer function
- linear fits
- non-linear fits (Simplex, Levenberg-Marquardt)
RELACS C++ library for data analysis

Data structures (classes, container):
- *EventData* — Spikes and other point process data
- *EventList* — Multi-trial spike trains

Algorithms:
- firing rates (mean, PSTH binned/kernel, 1/ISI)
- CV, Fano factor, ISI correlation
- vector strength, reliability, jitter
- mutual information (lower and upper bound)
Current odML developments

Done:

- Schema definition converges to version 1.
- Java and MatLab library to read, write and manipulate odML-files.
- Editor for odML metadata.
Current odML developments

Done:

- Schema definition converges to version 1.
- Java and MatLab library to read, write and manipulate odML-files.
- Editor for odML metadata.

On the way:

- Definition of the vocabularies.

Planned:

- Libraries for C/C++, Python ...
- collaboration with various initiatives (CRCNS, NIF, etc.)
### odML — Details

#### odML-Property:

<table>
<thead>
<tr>
<th>element name</th>
<th>description</th>
<th>occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of this property.</td>
<td>1</td>
</tr>
<tr>
<td>value</td>
<td>The value of this property.</td>
<td>1 - ∞</td>
</tr>
<tr>
<td>error</td>
<td>An error estimate of the value(s).</td>
<td>0 - ∞</td>
</tr>
<tr>
<td>unit</td>
<td>The unit of the value(s).</td>
<td>0 - 1</td>
</tr>
<tr>
<td>type</td>
<td>The datatype of the value e.g. int, float, string, date, time, binary, etc.</td>
<td>1</td>
</tr>
<tr>
<td>id</td>
<td>An identifier for each value, e.g. for a database.</td>
<td>0 - ∞</td>
</tr>
<tr>
<td>nameDefinition</td>
<td>Defines the property.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>valueDefinition</td>
<td>Defines each individual value.</td>
<td>0 - ∞</td>
</tr>
<tr>
<td>parent</td>
<td>This property is only meaningful if the parent property exists.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>parentValue</td>
<td>This property is only meaningful for a specific parent value.</td>
<td>0 - 1</td>
</tr>
</tbody>
</table>
## odML — Details

### odML-Section:

<table>
<thead>
<tr>
<th>element name</th>
<th>description</th>
<th>occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of this section.</td>
<td>1</td>
</tr>
<tr>
<td>alias</td>
<td>An alias name for this section.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>id</td>
<td>An identifier e.g. from a database by which the information of this section can be found.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>definition</td>
<td>Defines the section.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>vocabulary</td>
<td>The URI of the vocabulary which defines this section.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>parent</td>
<td>This section might be meaningful only if it is child of a parent section.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>parentURI</td>
<td>The URI of the parent section’s definition.</td>
<td>0 - 1</td>
</tr>
<tr>
<td>odML-Property</td>
<td>A section can contain properties ...</td>
<td>0 - ∞</td>
</tr>
<tr>
<td>odML-Section</td>
<td>... and subsections.</td>
<td>0 - ∞</td>
</tr>
</tbody>
</table>