Closed-loop experiments and metadata management with *Relacs* and *LabLog*

Jan Benda & Jan Grewe
Department Biology II
Ludwig-Maximilian University
Munich, Germany
Content

Part I  Closed-loop experiments: RELACS

Part II  Data management: LabLog

Part III  Metadata exchange
Part I —
Closed loop experiments with RELACS

by Jan Benda

www.relacs.net
“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

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

Input: Stimulus
Output: Neuronal response

Input: Neuronal response
Output: Next stimulus
Closed-Loop Experiments

RELACS is designed as a framework for closed-loop experiments that

- considerably speed up this traditional approach
- offer novel experimental possibilities
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 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 closed-loop

- Set stimuli relative to the neuron’s dynamic range

- Automatically control motorized electrodes (great for dual unit recordings!)

- ...
Example 1: Tuning Curve Measurement

Traditional:

![Graph showing firing rate vs. stimulus intensity]

Firing rate [Hz] vs. Stimulus intensity
Example 1: Tuning Curve Measurement

Traditional:

\[
\begin{array}{c|c}
\text{Stimulus intensity} & \text{Firing rate [Hz]} \\
0 & 0 \\
8 & 0 \\
16 & 0 \\
24 & 0 \\
32 & 0 \\
\end{array}
\]

\(n=5\)

\(\Delta I=8\)

either:

fast \rightarrow low resolution
Example 1: Tuning Curve Measurement

Traditional:

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

either:
- fast $\rightarrow$ low resolution
or:
- high resolution $\rightarrow$ slow
Example 1: Tuning Curve Measurement

Traditional:

- Start with either:
  - fast → low resolution
  - high resolution → slow

Closed loop:

1. start with low resolution
Example 1: Tuning Curve Measurement

Traditional:

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

Closed loop:

1. start with low resolution
2. increase resolution where necessary!
Example 1: Tuning Curve Measurement

**Traditional:**

- Firing rate [Hz] vs. Stimulus intensity
- n=17
- \( \Delta I = 2 \)
- Either:
  - Fast \( \rightarrow \) low resolution
  - OR:
    - High resolution \( \rightarrow \) slow

**Closed loop:**

- Firing rate [Hz] vs. Stimulus intensity
- n=9
- \( \Delta I = 2 \)

1. Start with low resolution
2. Increase resolution where necessary!
3. Further increase resolution
Example 1: 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
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 2: Optimal Stimulus Ensembles

Example 3: Iso-Response

Relaxed Electrophysiological data Acquisition, Control, and Stimulation

... enjoy your recordings
Modular Design

RELACS core with flexible C++ Plugins for

- hardware abstraction
- data pre-processing (filter, spike detectors)
- **experimental protocols**
- passive control
- 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
Metadata Acquisition with RELACS

RELACS records many metadata:

- All RELACS-controlled hardware settings (e.g. sampling rate)
- All settings and version numbers of the experimental protocols
- Properties of the stimuli used by the experimental protocols
- Main characteristics of the recorded cell
- General infos about the experiment (from a dialog)
Metadata Acquisition with RELACS
Metadata Acquisition with RELACS
Metadata Acquisition with RELACS

Minimal manual input necessary!
How should the metadata be stored?
How should the metadata be stored?

Fileformat?
Vocabulary?
Part II — LabLog — the long-term memory of your lab —

by Jan Grewe
lablog.sourceforge.net
LabLog - the Laboratory Logbook

- Storage of project related information:
  - Lab-journal (ideas, diary)
  - Experimental setup and hardware
  - Projects

- Management of acquired data:
  - Storage of metadata directly from RELACS
  - Search for data within/across projects (by SQL queries)
  - Export search results for further analysis

- Platform independent Java frontend to a mySQL database
The structure of the underlying relational database (about 60 tables):
Screen Shots

[Image of LaboratoryLogbook software interface showing projects and data management features]
Screen Shots

![Screen Shot of Database Retrieval Interface]

SQL Command:

```
SELECT d.datasetID, concat(d.filename, d.datafolder, d.filename) as file, e.name
FROM datasets d, experiments e
WHERE d.experimentID = e.id AND e.name like '%Control 1%';
```
How should the metadata be imported into the database?
Part III —

Talking about data

an Extensible Framework for Metadata Exchange
The Data-Chain

\[ C = \int \log_2 \left( 1 + \frac{S(f)}{N(f)} \right) df \]
The Data-Chain

The Contribution of Noise to Contrast Invariance of Orientation Tuning in Cat Visual Cortex

Jeffrey S. Anderson, Ilan Lampl, Deda C. Gillespie, David Ferster
Science, 2000
The Data-Chain

\[
C = \int_{\log_2 \left( \frac{1 + S(f)}{N(f)} \right)} \text{df}
\]
All data transfer requires talking about data.

How to exchange metadata?
Metadata

- is "data about data".
- describe recording conditions.

stimulus.type = white noise
Metadata

- is “data about data”.
- describe recording conditions.

\[
\text{stimulus.type} = \text{white noise}
\]

- name
- value
Metadata

• is “data about data”.
• describe recording conditions.

cell.baselineRate = 50 +/- 5 Hz
Metadata

- is “data about data”.
- describe recording conditions.

\[
\text{cell.baselineRate} = 50 \pm 5 \text{ Hz}
\]

- name
- value
- error
- unit
The Metadata Problem

- is “data about data”.
- describe recording conditions.

\[ \text{cell.baselineRate} = 50 \pm 5 \text{ Hz} \]

- What name to choose?
- What does it mean?
- How to organize metadata?
**Structure — How to organize metadata?**

<table>
<thead>
<tr>
<th>Property</th>
<th>name</th>
<th>value</th>
<th>error (optional)</th>
<th>unit (optional)</th>
<th>type (optional)</th>
<th>description (optional)</th>
</tr>
</thead>
</table>

### odML - metadata

<table>
<thead>
<tr>
<th>Section 1</th>
<th>name</th>
<th>description (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Property 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Property n</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Section 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Section n</td>
<td></td>
</tr>
</tbody>
</table>

- Implemented as the odML XML Schema
“normal” XML:

- A schema defines a fixed set of valid tags (names) and their relation
- Unknown properties invalidate the XML file
- The schema needs to be updated
The odML Schema

“normal” XML:

- A schema defines a fixed set of valid tags (names) and their relation
- Unknown properties invalidate the XML file
- The schema needs to be updated

“odd” XML — odML (open metadata markup language):

- The schema just defines the section and property structure (the grammar)
- Unknown properties do not invalidate the XML file
- The vocabulary (definition of properties) is inherently extensible
Definition — What name to choose?

- Sections group properties logically. E.g.:
  - Stimulus
  - Experiment description
  - Cell, experimental subject, preparation
  - Hardware properties, hardware settings
  - Dataset
  - Analysis parameter
  - etc.

Implementation: eVoc XML file conform to odML
## 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>
Definition — Examples

**HardwareSettings:**

**DataAcquisition:**

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIUsedChannelCount</td>
<td>int</td>
<td>The number of used analog input channels.</td>
</tr>
<tr>
<td>AISampleRate[n]</td>
<td>int</td>
<td>The sample rate with which an individual input channel was sampled. Given in Hz.</td>
</tr>
<tr>
<td>AIChannelGain[n]</td>
<td>float</td>
<td>The gain of an input channel.</td>
</tr>
<tr>
<td>AIReference[n]</td>
<td>string</td>
<td>The reference to which voltages were measured. Usually either &quot;common ground&quot; or &quot;differential&quot;.</td>
</tr>
<tr>
<td>AIPolarity[n]</td>
<td>string</td>
<td>The polarity of the measurement &quot;unipolar&quot; or &quot;bipolar&quot;.</td>
</tr>
</tbody>
</table>
# Definition — Examples

## Cell:

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>string</td>
<td>An identifier of this cell.</td>
</tr>
<tr>
<td>Type</td>
<td>string</td>
<td>The type of the recorded cell.</td>
</tr>
<tr>
<td>BrainRegion</td>
<td>string</td>
<td>The Region the cells are located in. For example Retina, Cortex, Cerebellum</td>
</tr>
<tr>
<td>BrainSubRegion</td>
<td>string</td>
<td>For example CA1 in hippocampus.</td>
</tr>
<tr>
<td>Layer</td>
<td>string</td>
<td>For example layer 4 in CA1.</td>
</tr>
<tr>
<td>Ganglion</td>
<td>string</td>
<td>Like the pro-, meta- and mesothoracal ganglion in invertebrates.</td>
</tr>
<tr>
<td>RecLocation</td>
<td>string</td>
<td>The recording location in the cell. Axonal, dendritic, somatic?</td>
</tr>
<tr>
<td>RestingPotential</td>
<td>float</td>
<td>What is the cell’s resting potential. An indiator for the recording quality.</td>
</tr>
<tr>
<td>BaselineRate</td>
<td>float</td>
<td>For spiking cells, the spontaneous activity which might be an indicator for the cell’s health status and thus recording quality.</td>
</tr>
</tbody>
</table>
How to use eVoc?

1. Assemble properties:

- If you find an appropriate property in eVoc, use it!
- Ignore all properties that do not match.
- Add your own properties that are not yet in eVoc, if possible with a description.
How to use eVoc?

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

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

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

2. Write them into an odML XML file

3. Transfer them to an analysis or database program

This is all done by RELACS and LabLog automatically!
Vision

- **odML** and **eVoc** are publicly available on the G-Node server.

- Discussion forum for extensions to vocabulary and schema.

- Recording, datamanagement, and analysis software should use **eVoc** for metadata exchange.

- Data can be published on G-Node database which can also import this kind of metadata.
Vision

- odML and eVoc are publicly available on the G-Node server.
- Discussion forum for extensions to vocabulary and schema.
- Recording, datamanagement, and analysis software should use eVoc for metadata exchange.
- Data can be published on G-Node database which can also import this kind of metadata.

⇒ You can easily share your data!
⇒ Your data can be found!
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).
The Data Life-Cycle

- Well annotated data can be found and reused easily

⇒ Your data deserves it!
Summary

• Closed-loop experiments may considerably speed up electrophysiological experiments and allow for new experimental designs.

• RELACS is a software platform for closed-loop experiments and automatically annotates your data (www.relacs.net).

• LabLog helps you keeping track of your data (lablog.sourceforge.net)

• A unifying framework for metadata exchange is needed for data sharing between data acquisition, analysis, and management software and services.
int Example::main( void ) {
    // some initialization ...
    OutData signal;
    signal.setTrace( "LeftSpeaker" );
    signal.sineWave( frequency, duration, amplitude );
    SampleData 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 );
        }
    }
}
Why C++

- well structured (object oriented)
- platform independent
- efficient and controllable memory usage
- very fast
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)
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)
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!