RELACS — a modular software platform for closed-loop experiments

Jan Benda
Department Biologie II
Ludwig-Maximilian Universität München
Sensory electrophysiology

Electrosensory systems of weakly electric fish

Auditory system of grasshopper and crickets

How are sensory stimuli processed by sensory systems?
Closed-loop experiments with RELACS

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
- ...

relacs
Example: tuning curve measurement

Traditional:

![Graph showing tuning curve with firing rate on the y-axis and stimulus intensity on the x-axis. The graph has a step-like increase in firing rate as stimulus intensity increases.]
Example: tuning curve measurement

Traditional:

Firing rate [Hz] vs. Stimulus intensity

- n=5
- ΔI=8

either:
fast → low resolution
Example: tuning curve measurement

Traditional:

either:
fast $\rightarrow$ low resolution

or:
high resolution $\rightarrow$ slow
Example: tuning curve measurement

Traditional:

- Firing rate [Hz]
- Stimulus intensity
- n=17
- ∆I=2

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

Closed loop:

- Firing rate [Hz]
- Stimulus intensity
- n=5
- ∆I=8

1. start with low resolution
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!
Example: tuning curve measurement

Traditional:

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

Closed loop:

- Firing rate [Hz]
- Stimulus intensity
- $n=9$
- $\Delta I=2$

1. start with low resolution
2. increase resolution where necessary!
3. further increase resolution

either:
- fast $\rightarrow$ low resolution
or:
- high resolution $\rightarrow$ slow
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).
- ...

Relaxed Electrophysiological data Acquisition, Control, and Stimulation

RELACS is a framework for closed-loop experiments

In RELACS the closed-loop cycle can be freely programmed as a C++ plugin (“research protocol”).

The research-protocol plugins

- take recorded and pre-analyzed data
- perform analysis & display results
- generate next stimulus
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

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

- Synaptic conductances
- Voltage-gated conductances

Andrew A. Sharp, Michael B. ONeil, L. F. Abbott, & Eve Marder (1993) *J Neurophysiol*
\[ I_1 = g_{syn}(V_2) \cdot (V_1 - E) \]
\[ I_2 = g_{syn}(V_1) \cdot (V_2 - E) \]


- Artificially couple real neurons
- Couple with simulated neurons
Precision-switch by leak conductance

Dynamic clamp: leak current $I = g(V - E), E := V_{rest}$

Boucsein, Ammer, Benda (2010) in preparation
Modular design

RELACS core with flexible C++ Plugins for

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

RELACS integrates all hardware components.

Research 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
Options for research protocols

MembraneResistance Settings

version 1.0 (Nov 12, 2009)
by Jan Benda

Stimulus
- Amplitude of output signal (amplitude)
- Compute amplitude from vstep and estimated membrane resistance (userrm)
- Steady-state voltage amplitude induced by output signal (vstep)
- Duration of output (duration)
- Duration of pause between outputs (pause)
- Repetitions of stimulus (repeats)

Analysis
- Window length for steady-state analysis (sswidth)
- Fix steady-state potential for fit (nossfit)
- Plot standard deviation of membrane potential (plotstddev)
- Set results to the session variables (setdata)

Set values as default
Changes overwrite macro options

Project
Experiment

Ok  Apply  Run  Reset  Close
Macros

... execute research protocols with specific parameter settings:

$FIField startsession
FIField
SysLatency
FICurve: duration=40ms; pause=260ms;
detector Spikes−1: save
```c
int Example::main( void ) {
    double frequency = number( "frequency" );
    double duration = number( "duration", "s" );
    double amplitude = 0.0;
    OutData signal;
    signal.setTrace( "LeftSpeaker" );
    signal.sineWave( frequency, duration, amplitude );
    SampleDataD rate( 0.0, duration, 0.001 );
    for ( int counter=0; counter<20; counter++ ) {
        write( signal );
        sleep( duration + pause );
        EventData spikes( events( "Spikes-1" ), signalTime(), signalTime() + duration );
        double meanrate = spikes.rate( 0.3*duration, duration );
        spikes.addRate( rate, counter, GaussKernel( sigma ) );
        P.lock();
        P.clear();
        P.setXRange( 0.0, duration );
        P.plot( rate, 1000.0, Plot::Yellow, 2, Plot::Solid );
        P.draw();
        P.unlock();
        if ( meanrate < targetrate ) {
            amplitude *= 2.0;
            signal.sineWave( frequency, duration, amplitude );
        }
    }
    return Completed;
}
```
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 \( x \mid y \) 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)
Research protocols also run on simulated data:

• test closed-loop algorithms

• directly compare models with experimental data
Meta-data: the data-chain

\[ C = \int \log_2 \left( 1 + \frac{S(f)}{N(f)} \right) df \]
Meta-data: the data-chain

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

German neuroinformatics node

www.g-node.de
Meta-data: the data-chain

German neuroinformatics node
www.g-node.de

- All data transfer for analysis, management, and sharing requires talking about data.
Meta-data: the data-chain

- All data transfer for analysis, management, and sharing requires talking about data.
- How to exchange metadata?
- How to record metadata?

German neuroinformatics node
www.g-node.de
The meta-data problem

Name-value (+unit) pairs for:

- Stimuli
- Experimental settings
- Cell, preparation, experimental subject
- Hardware properties
- Analysis parameter
- etc.

```
stimulusType = white noise
```

name

value
The meta-data problem

Name-value (+unit) pairs for:

- Stimuli
- Experimental settings
- Cell, preparation, experimental subject
- Hardware properties
- Analysis parameter
- etc.

But:

- What name to choose?
- What does it mean?
- How to share meta-data?
odML — a proposal
— open metadata markup language —

• simple key-value based, hierarchical structure:

  • all meta-data can be immediately stored
    (e.g. no XML namespace extensions required)
  • independent of data-base schemas
  • standardization through terminologies

## odML terminologies

### names & definitions

**Hardware:**

**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>

Grewe, Wachtler, Benda (2010) submitted. [www.g-node.org/odml](http://www.g-node.org/odml)
How to use odML?

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

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

2. Write them into an odML XML file

3. Transfer them to an analysis or database program
How to use odML?

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

2. Write them into an odML XML file

3. Transfer them to an analysis or database program
   ⇒ odML flexibility: all available metadata can be immediately stored in a file
How to use odML?

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

2. Write them into an odML XML file

3. Transfer them to an analysis or database program

⇒ odML flexibility: all available metadata can be immediately stored in a file

⇒ odML standard: The G-Node electrophysiology database is based on odML: ww.g-node.org
How to record meta-data?

• Every online recording software knows about most of the important meta-data!
How to record meta-data?

- Every online recording software knows about most of the important meta-data!

⇒ All available Meta-data should be written to a file directly from the recording software, if possible using odML terminologies.
How to record meta-data?

- Every online recording software knows about most of the important meta-data!

⇒ All available Meta-data should be written to a file directly from the recording software, if possible using odML terminologies.

- Such automated meta-data storage is the basis for making public data bases, such as www.g-node.org, work.
Meta-data acquisition by RELACS
# Meta-data acquisition by RELACS

**Recording**
- Recording quality: Good
- Comment:
- Scientist: Jan Benda
- Temperature: 22 °C
- Humidity: 54 %

**Cell**
- Cell type: Low-frequency receptor
- Recording location: Auditory nerve
- Side: Left
- Depth: 0 μm

**Subject**
- Species: Locusta migratoria
- Sex: Female
- Age: Middle-aged
- Preparation: in vivo dorsal

---

**Figure Description**

- **Threshold (dB):**
  - Frequency range: 10-80 kHz
  - Intensity range: 0-100 dB SPL

- **Firing rate (Hz):**
  - Intensity range: 0-100 dB SPL

- **Spectrum:**
  - Frequency range: 10-80 kHz

- **Stop Session Dialog:**
  - Meta-data acquisition settings:
    - Recording quality
    - Comment
    - Scientist
    - Temperature
    - Humidity
    - Cell type
    - Recording location
    - Side
    - Depth
    - Species
    - Sex
    - Age
    - Preparation

---

**Timeline:**
- **00:01**
- **2008-12-02-ab/**
- **12%** Frequency 4 kHz, Increment 4, Intensity 55 dB SPL, Loop 2

**RELACS - Relaxed Electrophysiological data Acquisition, Control, and Stimulation: Version 0.9.6 (11/24/06)**

---

**Note:**
- The graphical interface is designed to facilitate data acquisition, control, and stimulation in electrophysiological research.
- The data acquisition process includes setting up recording parameters, specifying the cell and subject details, and adjusting the stimulus conditions.
- The interface allows for real-time monitoring and control of the experiment, ensuring accurate data collection and analysis.
RELACS records many meta-data:

- General infos about the experiment (from the dialog)
- Main characteristics of the recorded cell
- All RELACS-controlled hardware settings (e.g. sampling rate)
- All settings and version numbers of the research protocols
- Properties of the stimuli
Closed-loop experiments
→ Dynamic clamp
→ Simulation mode
→ Hardware independent
→ Data analysis libraries
→ Meta-data storage
→ Open source, GPL, Linux
→ \sim 160 000 lines of C++ code

by Jan Benda

www.relacs.net