JÜLICH
FORSCHUNGSZENTRUM

N M B U
Norwegian University
of Life Sciences

LFPy and **hybrid scheme** for Local Field Potentials

CNS 2015 Tutorial T2:
Modeling and analysis of extracellular potentials

July 18 2015 | Espen Hagen, e.hagen@fz-juelich.de

Collaborations

`LFPy`:

- Henrik Lindén
- Espen Hagen
- Szymon Łęski
- Eivind S. Norheim
- Klas H. Pettersen
- Gaute T. Einevoll

Hybrid LFP model:

- Espen Hagen
- David Dahmen
- Maria L. Stavrinou
- Tom Tetzlaff
- Henrik Lindén
- Sacha van Albada
- Markus Diesmann
- Sonja Grün
- Gaute T. Einevoll

Topics

- Why model extracellular potentials?
- Forward modeling scheme (brief repetition)
- **`LFPy`**:
  - Introduction
  - Requirements, installation
  - Class overview
  - Examples
- Hybrid LFP model (**`hybridLFPy`**):
  - Introduction
  - Components
  - Application with 2-population network
  - Application with cortical column model
- Summary & Outlook

Why model extracellular potentials?

- **Improve understanding of experimental measurements:**
  - Extracellular action-potential shapes:
    - Gold et al. *J Neurophysiol* (2006)
    - Pettersen&Einevoll. *Biophys J* (2008)
    - Hagen et al. *J Neurosci Methods* (2015)
    - Ness et al. *Neuroinform* (2015)
  - Spiking component of LFP:
    - Schomburg et al. *J. Neurosci* (2012)
  - Spectral content of LFP:
    - Lindén et al. *J Comput Neurosci* (2010)
    - Tomsett et al. *Brain Struct Funct* (2014)
  - Reach of LFP:
    - Lindén et al. *Neuron* (2011)
    - Łęski et al. *PLOS Comput Biol* (2013)
  - Role of active membranes:
    - Reimann et al. *Neuron* (2013)

Why model extracellular potentials?

- **Methods validation:**
  - Spike sorting:
    - Franke et al. *Proc IEEE Eng Med Biol Soc* (2010)
    - Einevoll et al. *Curr Op Neurobiol* (2012),
    - Thorbergsson et al. *J Neurosci Methods* (2013)
    - Hagen et al. *J Neurosci Methods* (2015)
  - Current-source density (CSD) reconstruction:
    - Pettersen et al. *J Comput Neurosci* (2008)
    - Łęski et al. *Neuroinform* (2011)
    - Głąbska et al. *PLOS One* (2014)
    - Ness et al. *Neuroinform* (2015)

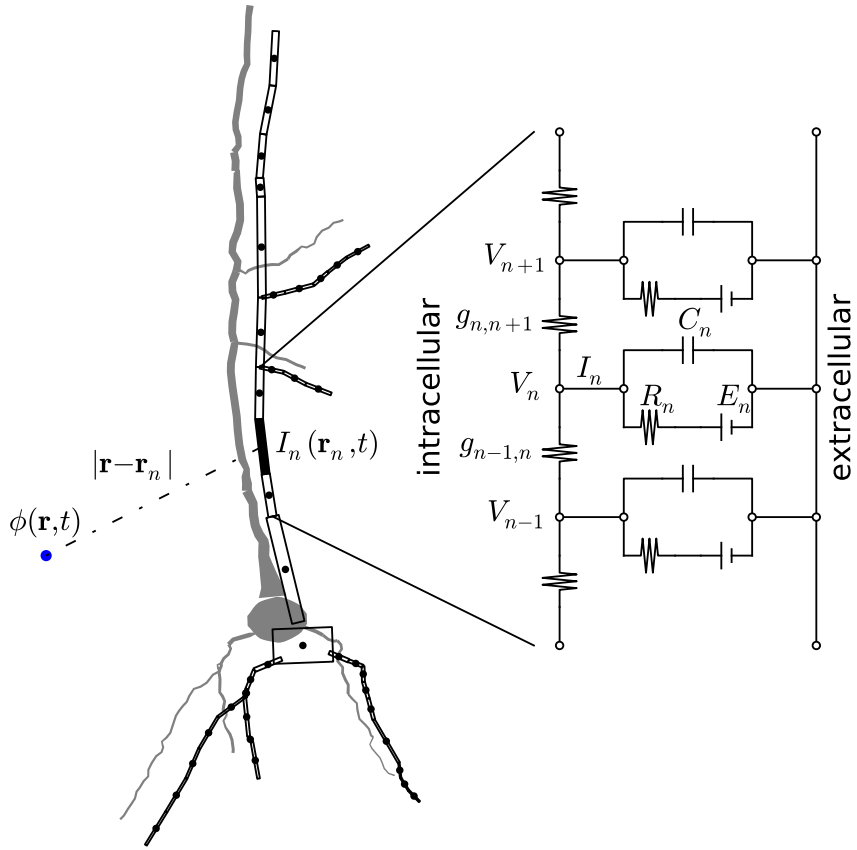
Forward modeling of extracellular potentials

Biophysical background:

- Current balance intracellular node point, compartment n:

$$I_n = C_n \frac{dV_n}{dt} + \sum_j I_n^j = g_{n,n+1}(V_{n+1} - V_n) - g_{n-1,n}(V_n - V_{n-1})$$

- Simulated using **NEURON** (neuron.yale.edu) Hines et al. (2009))
- Extracellular potentials are computed from I_n



Lindén et al. (2014)

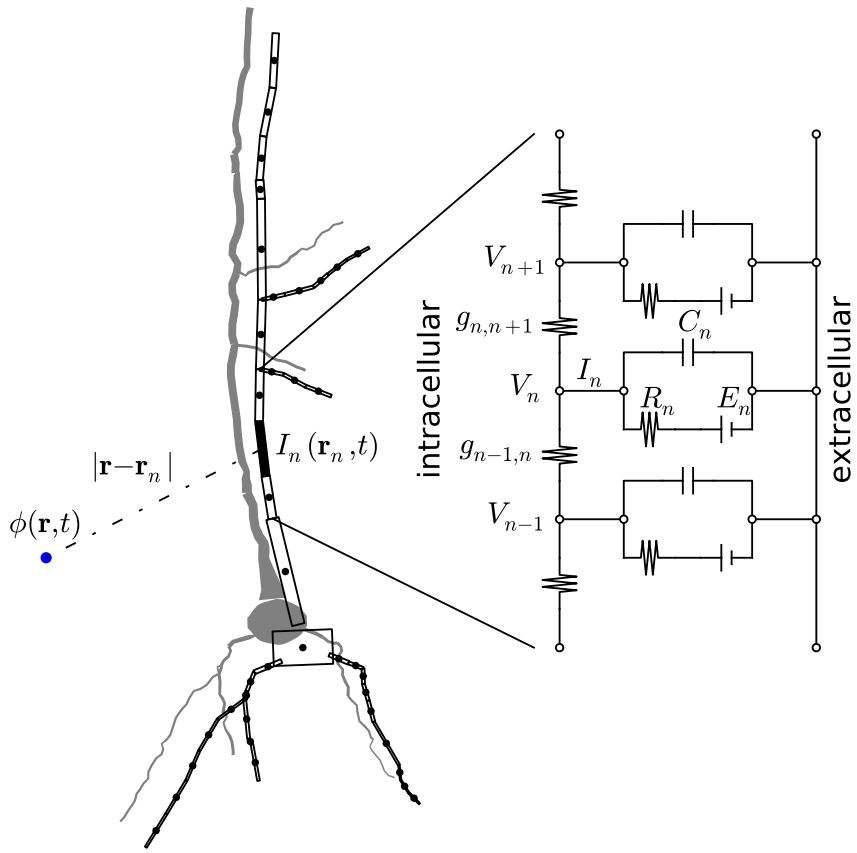
Forward modeling of extracellular potentials

Biophysical background:

- Poisson's equation in electrostatics

$$\nabla \cdot (\sigma \nabla \phi) = -\frac{\rho}{\sigma}$$

$\phi(\mathbf{r}, t)$ - electric potential
$\rho(\mathbf{r}, t)$ - current source density
$\sigma(\mathbf{r})$ - conductivity



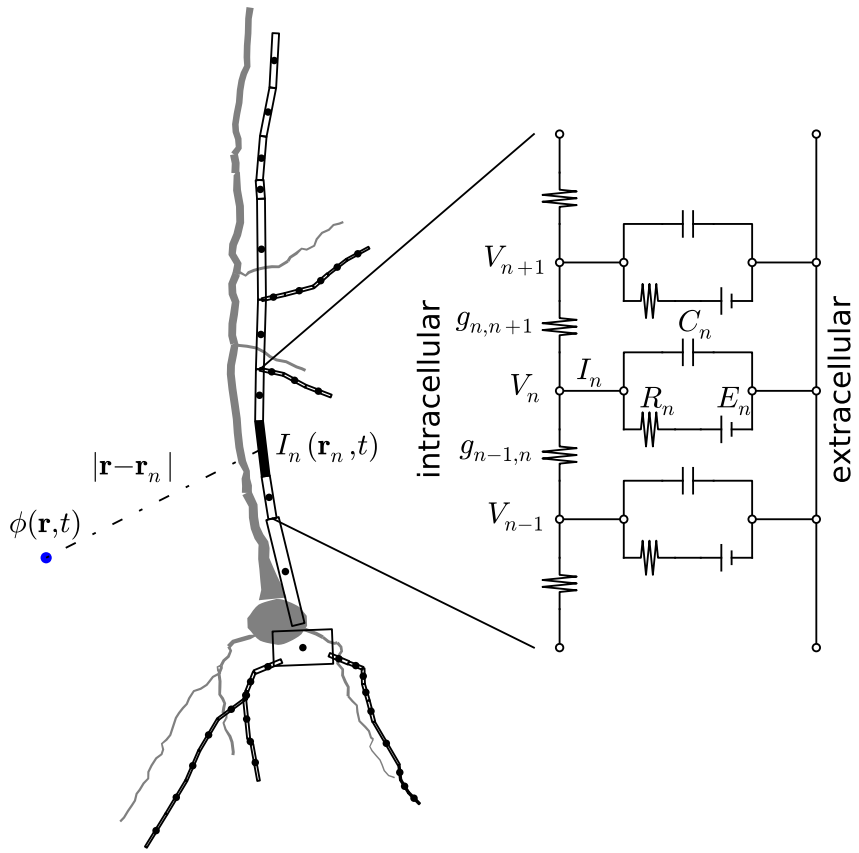
Lindén et al. (2014)

Forward modeling of extracellular potentials

Biophysical background:

- Assumptions:
  - Quasi-static approximation of Maxwell's equations
  - Extracellular medium:
    - linear
    - isotropic
    - homogeneous
    - ohmic

    (scalar, real σ)
  - Linear superposition
- $\phi(r \to \infty) = 0$



Lindén et al. (2014)

Forward modeling of extracellular potentials

Biophysical background:

- Quasi-static approximation of Maxwell's equations:

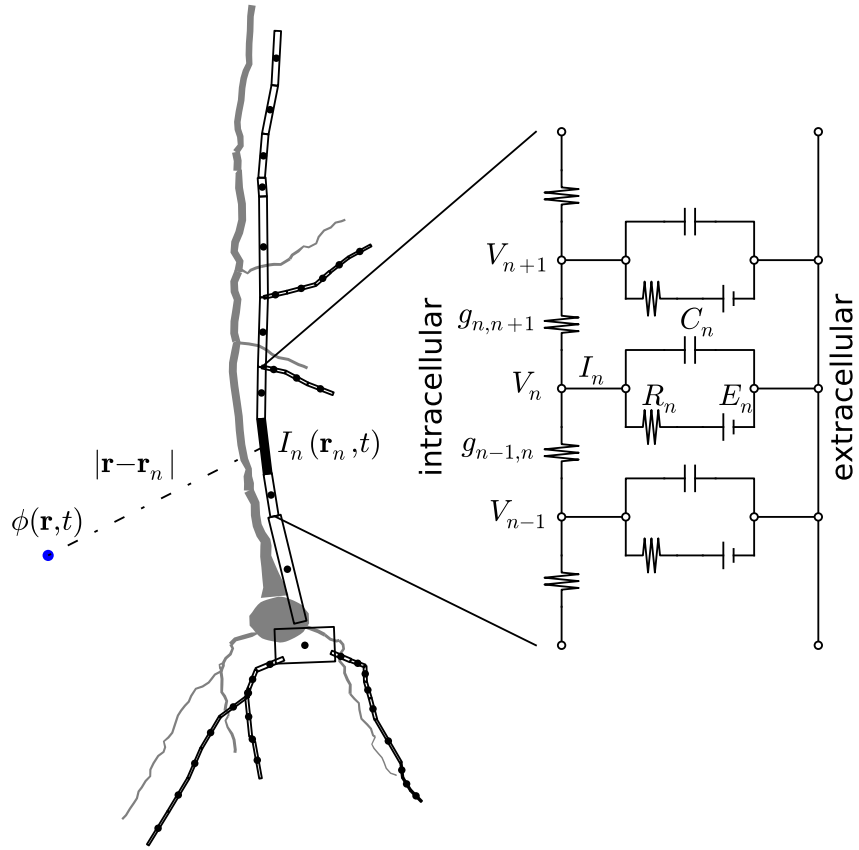
$$\nabla \cdot \mathbf{E} = \frac{q}{\epsilon_0}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \approx 0$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{B} = \mu(\mathbf{J} + \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}) \approx \mu \mathbf{J}$$

- **E** - electric field; q - charge density;
  ϵ_0 - free space permittivity;
  B - magnetic field; μ - permeability;
  J - ohmic + polarization currents



Lindén et al. (2014)

Forward modeling of extracellular potentials

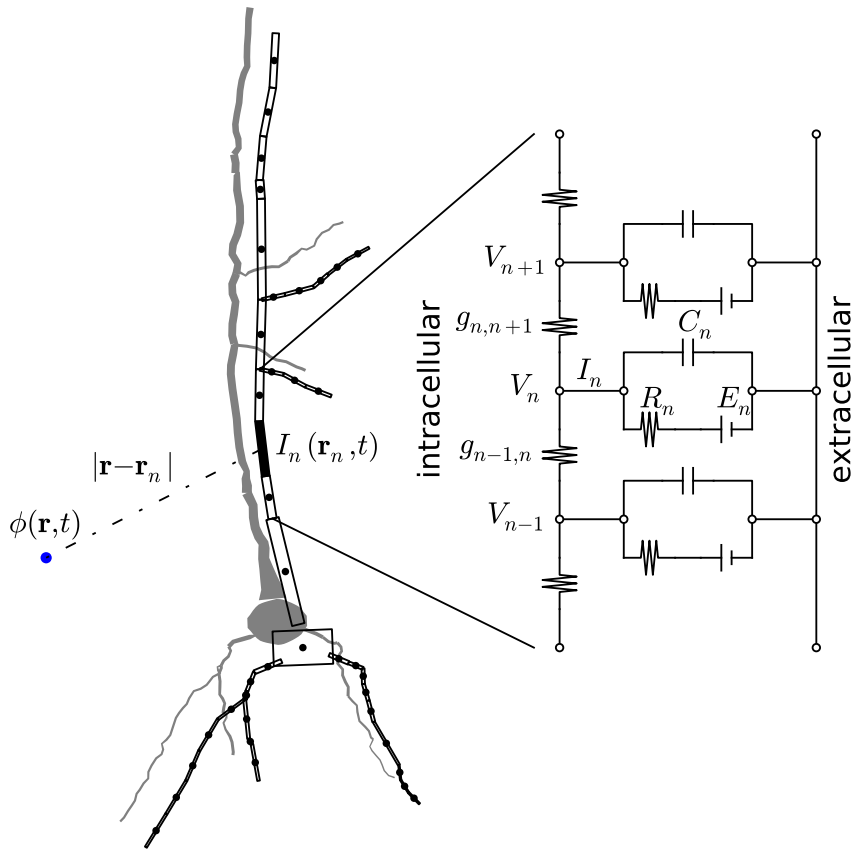
Biophysical background:

- Point current source

$$\phi(\mathbf{r}, t) = \frac{1}{4\pi\sigma} \frac{I_0(t)}{|\mathbf{r} - \mathbf{r}_0|}$$

- Linear summation N point sources

$$\phi(\mathbf{r}, t) = \frac{1}{4\pi\sigma} \sum_{n=1}^{N} \frac{I_n(t)}{|\mathbf{r} - \mathbf{r}_n|}$$



Lindén et al. (2014)

Forward modeling of extracellular potentials

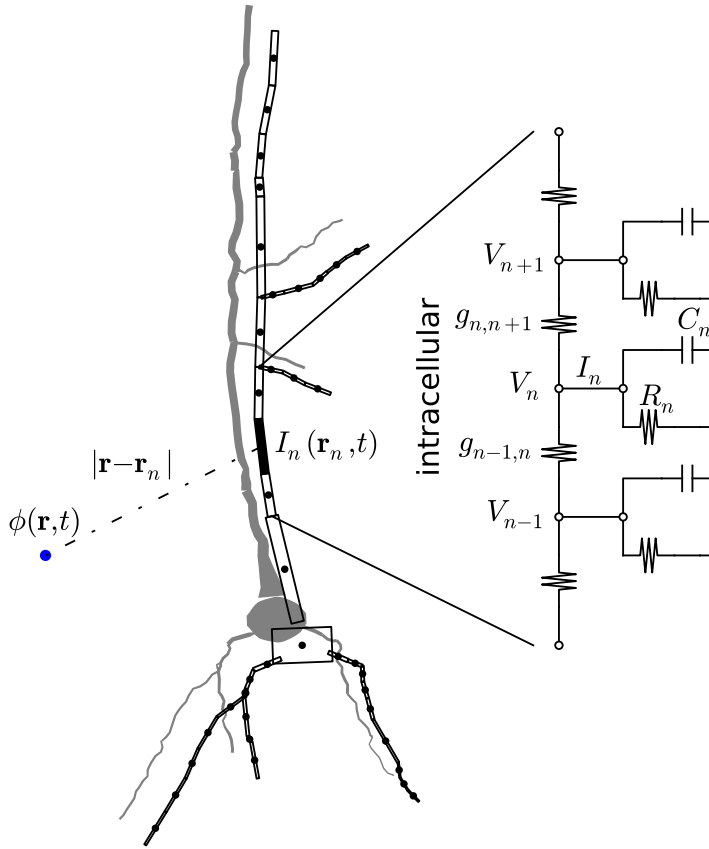
Biophysical background:

- Line sources

$$\phi(\mathbf{r},t) = \frac{1}{4\pi\sigma}\sum_{n=1}^{N} I_n(t)\int \frac{d\mathbf{r}_n}{|\mathbf{r}-\mathbf{r}_n|}$$

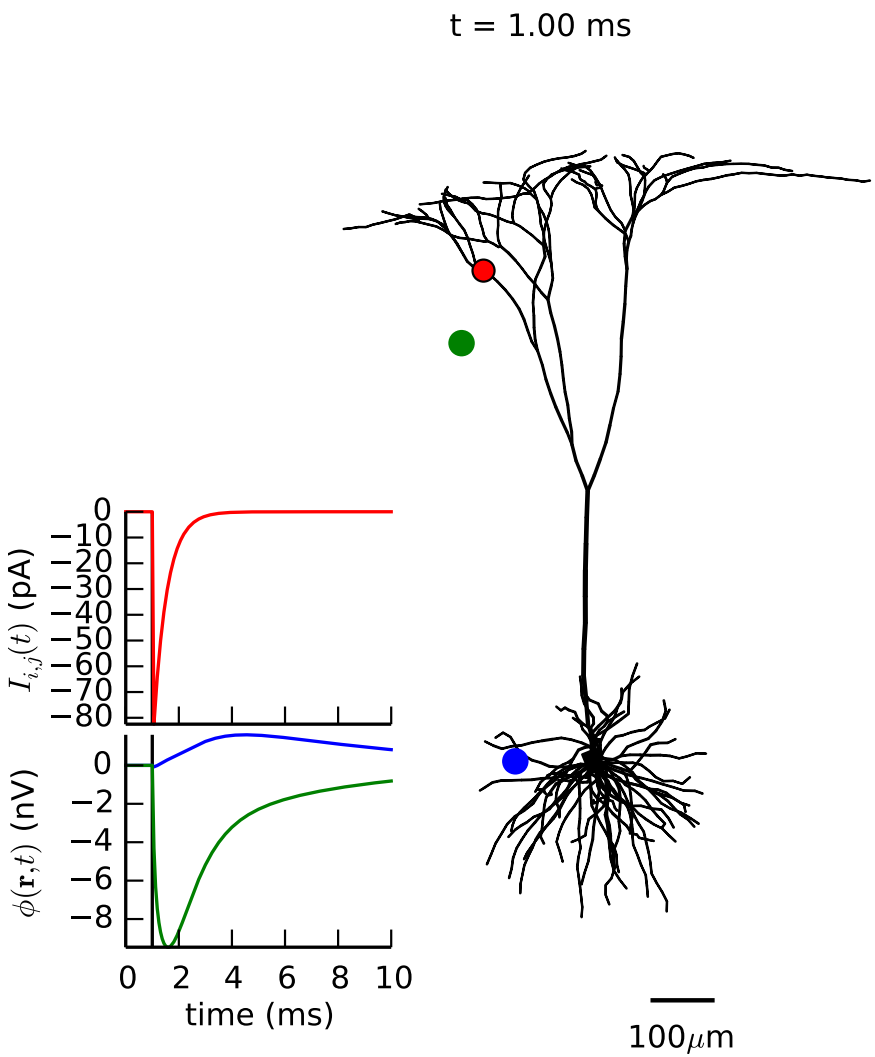
$$= \frac{1}{4\pi\sigma}\sum_{n=1}^{N} \frac{I_n(t)}{\Delta s_n}\ln\left|\frac{\sqrt{h_n^2+r_{\perp n}^2}-h_n}{\sqrt{l_n^2+r_{\perp j}^2}-l_n}\right|$$

- Δs_n - segment length; h_n - longitudal distance to one end of segment; $r_{\perp n}$ - perpendicular distance to segment axis; $l_n = \delta s_n + h_n$.
- see Holt & Koch. (1999), *J Comput Neurosci* 6:169-184



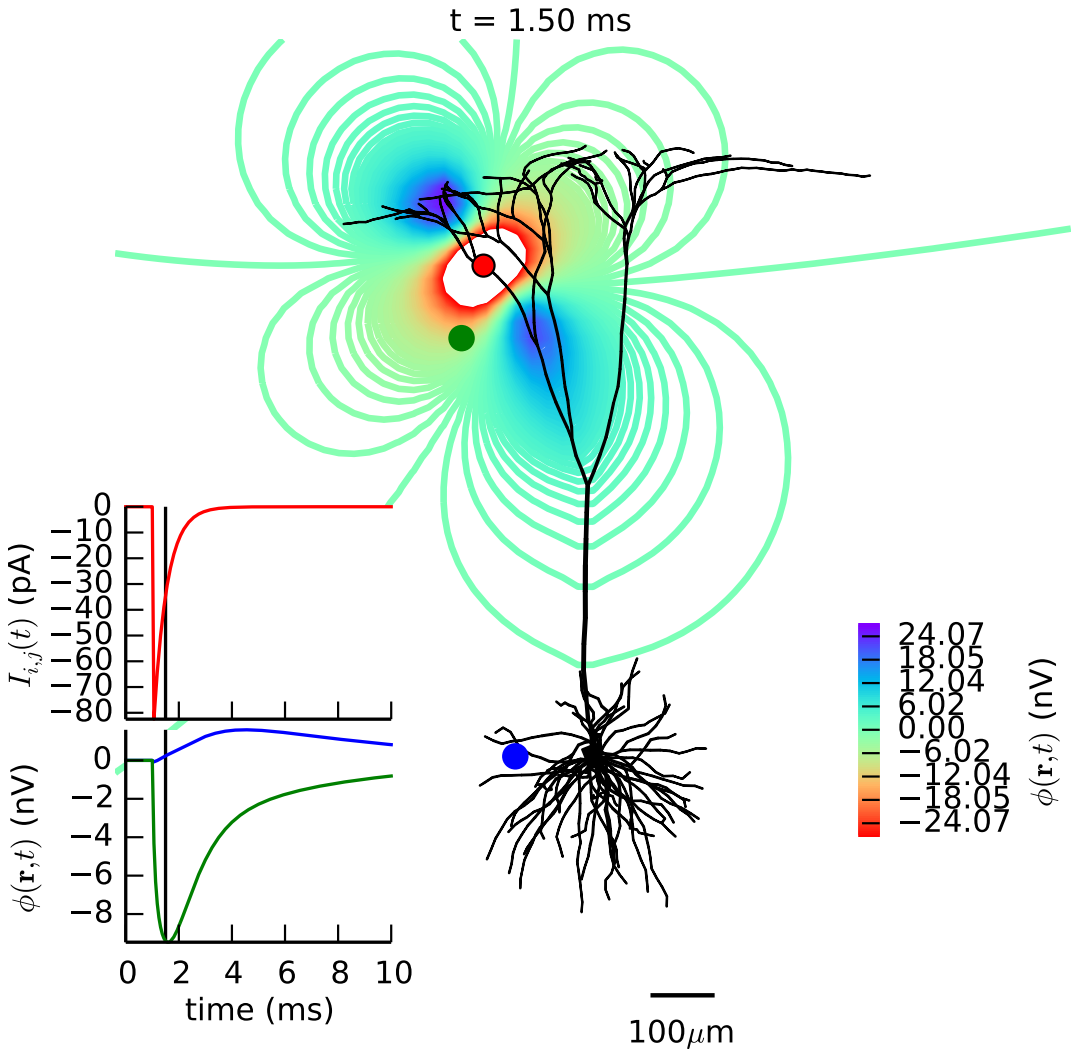
Lindén et al. (2014)

Forward modeling of extracellular potentials



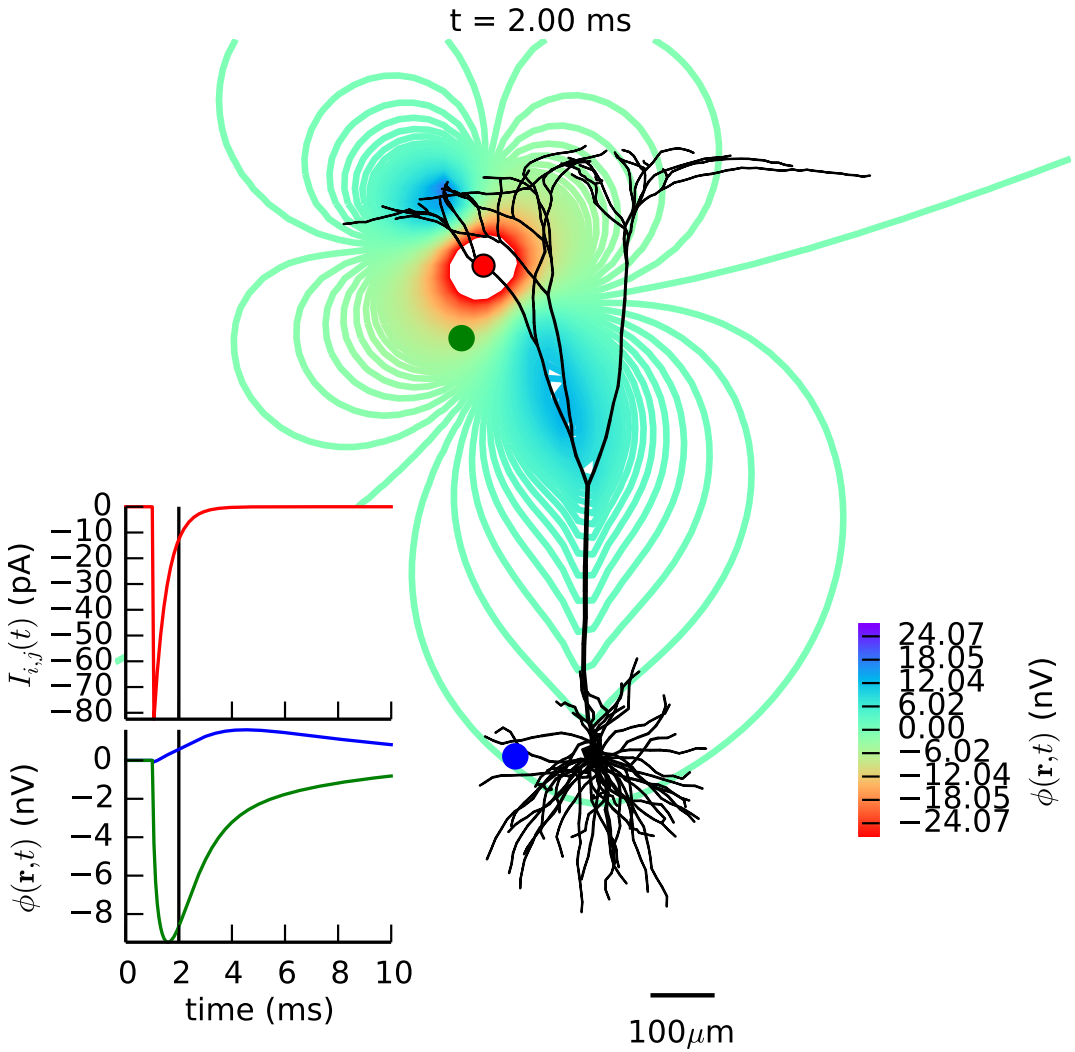
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



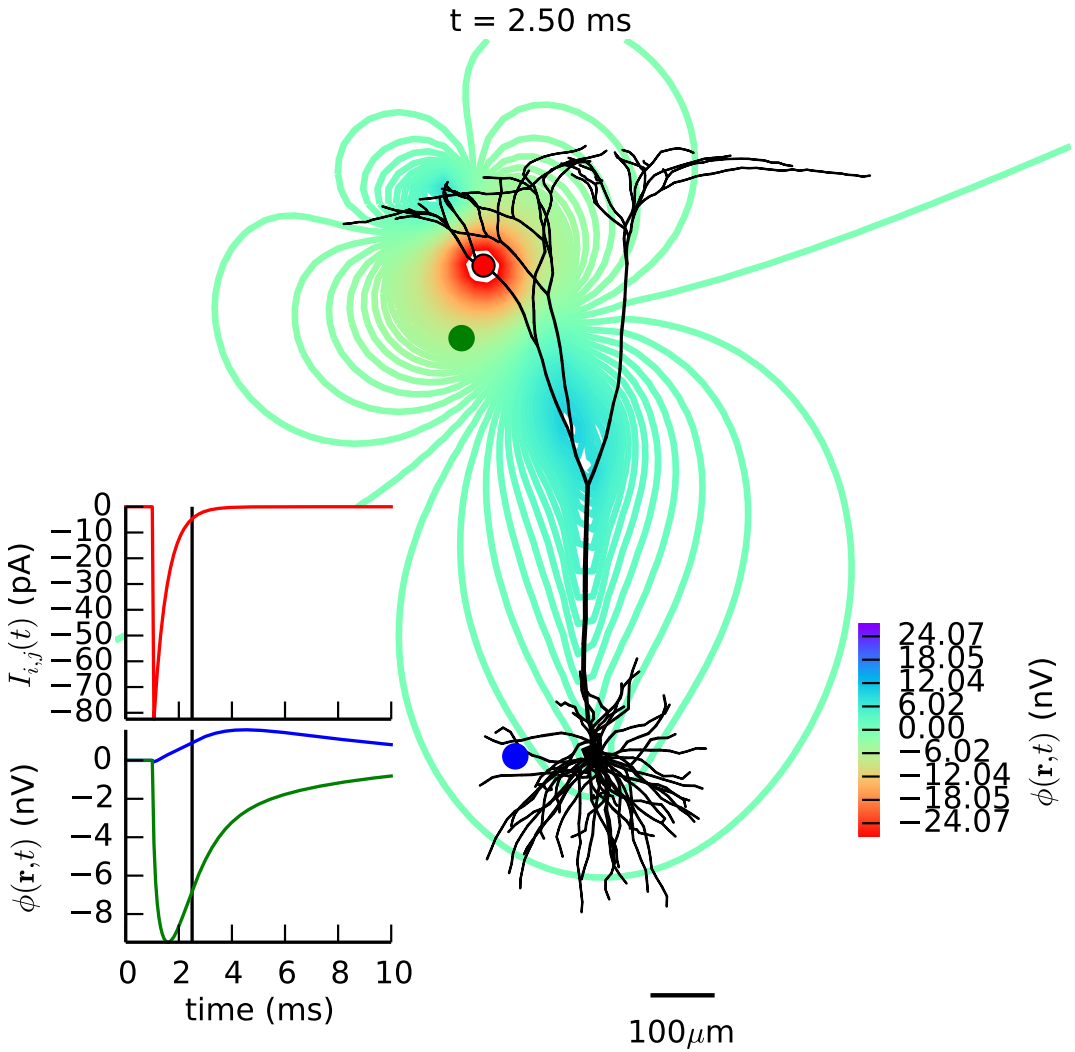
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



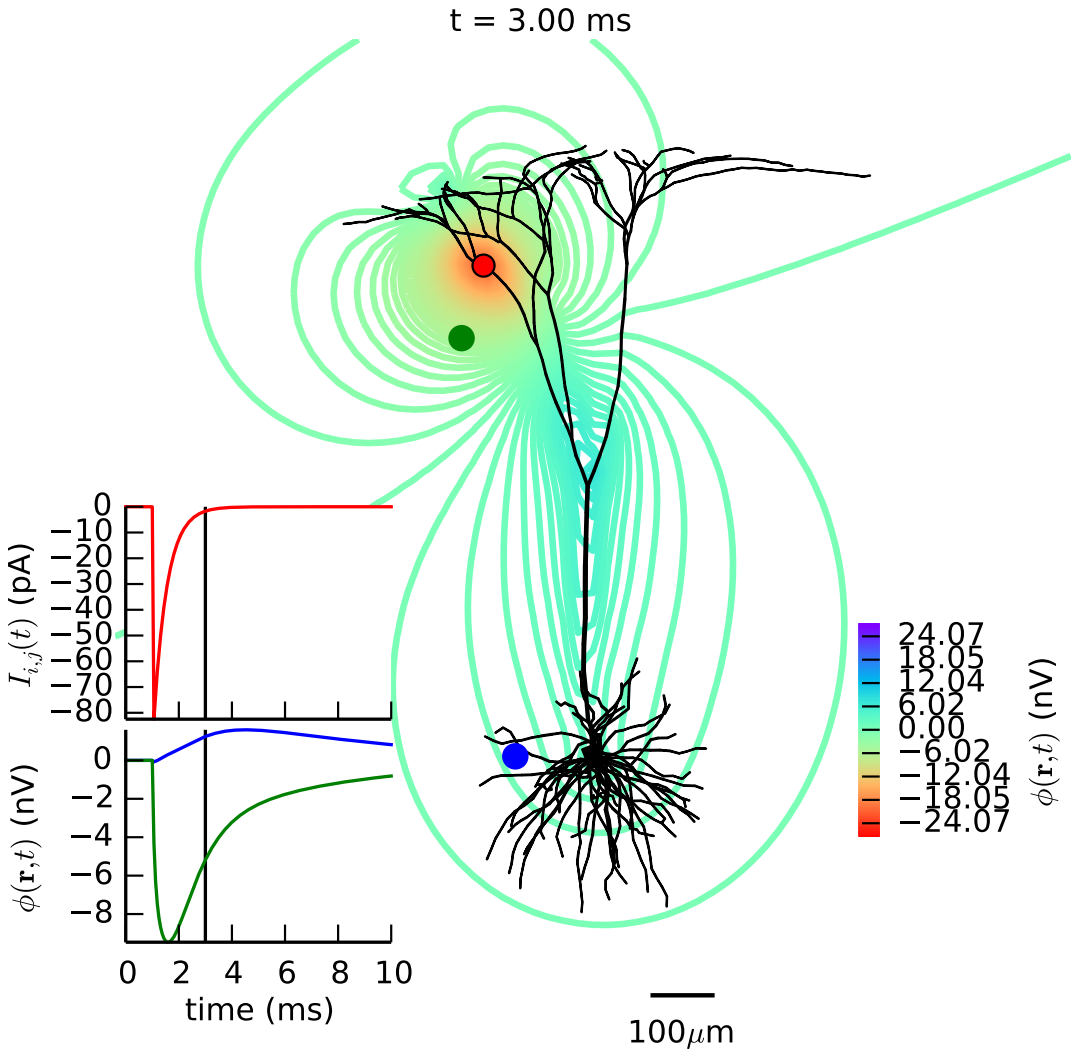
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



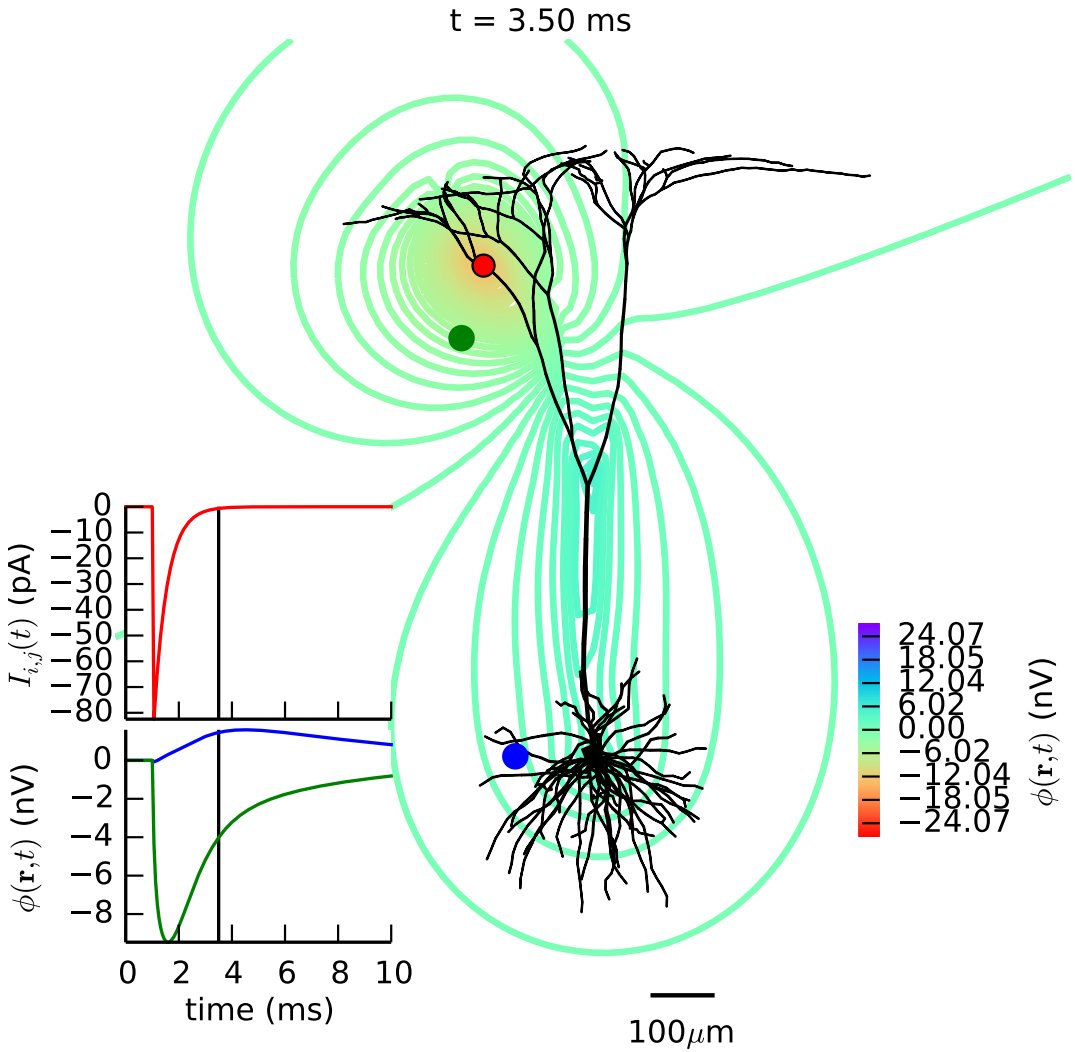
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



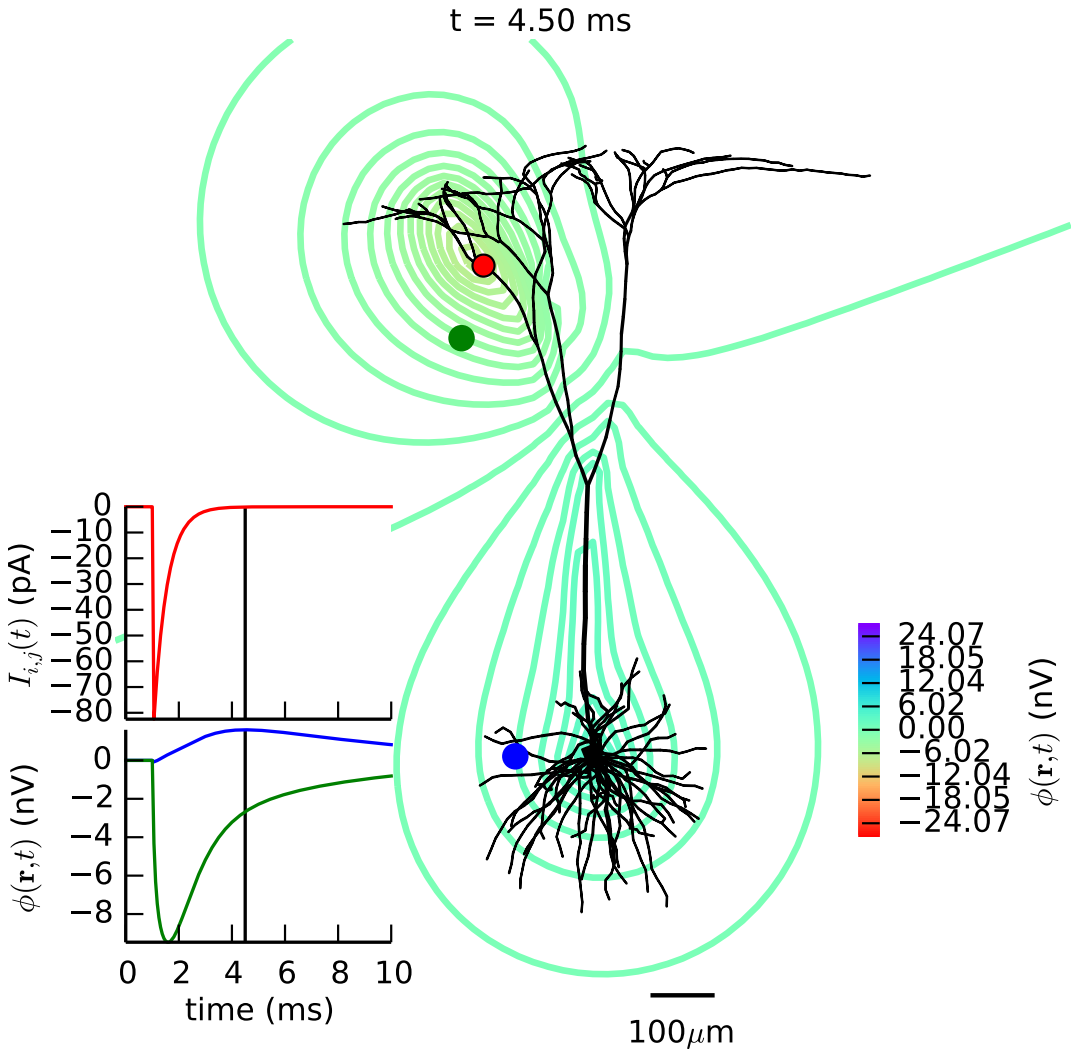
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



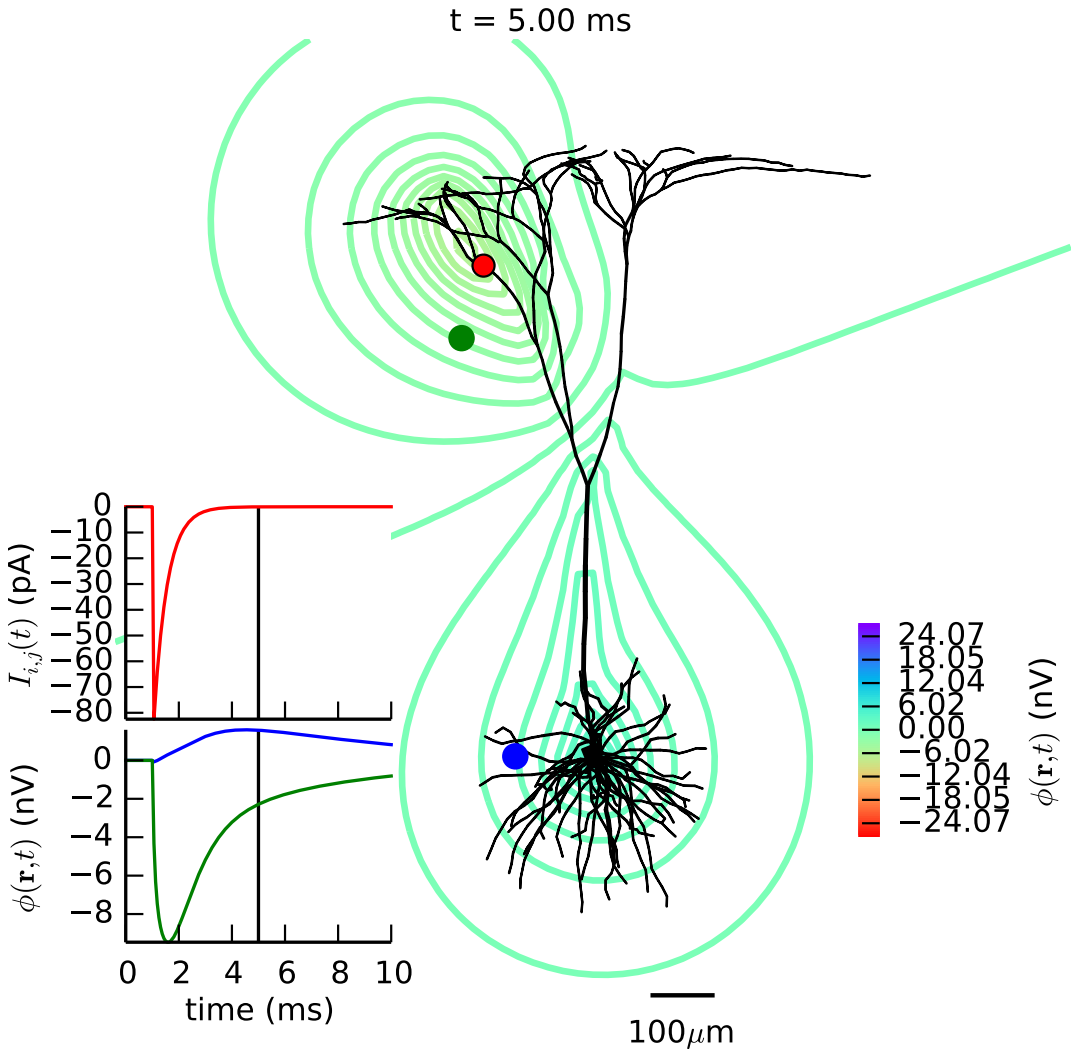
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



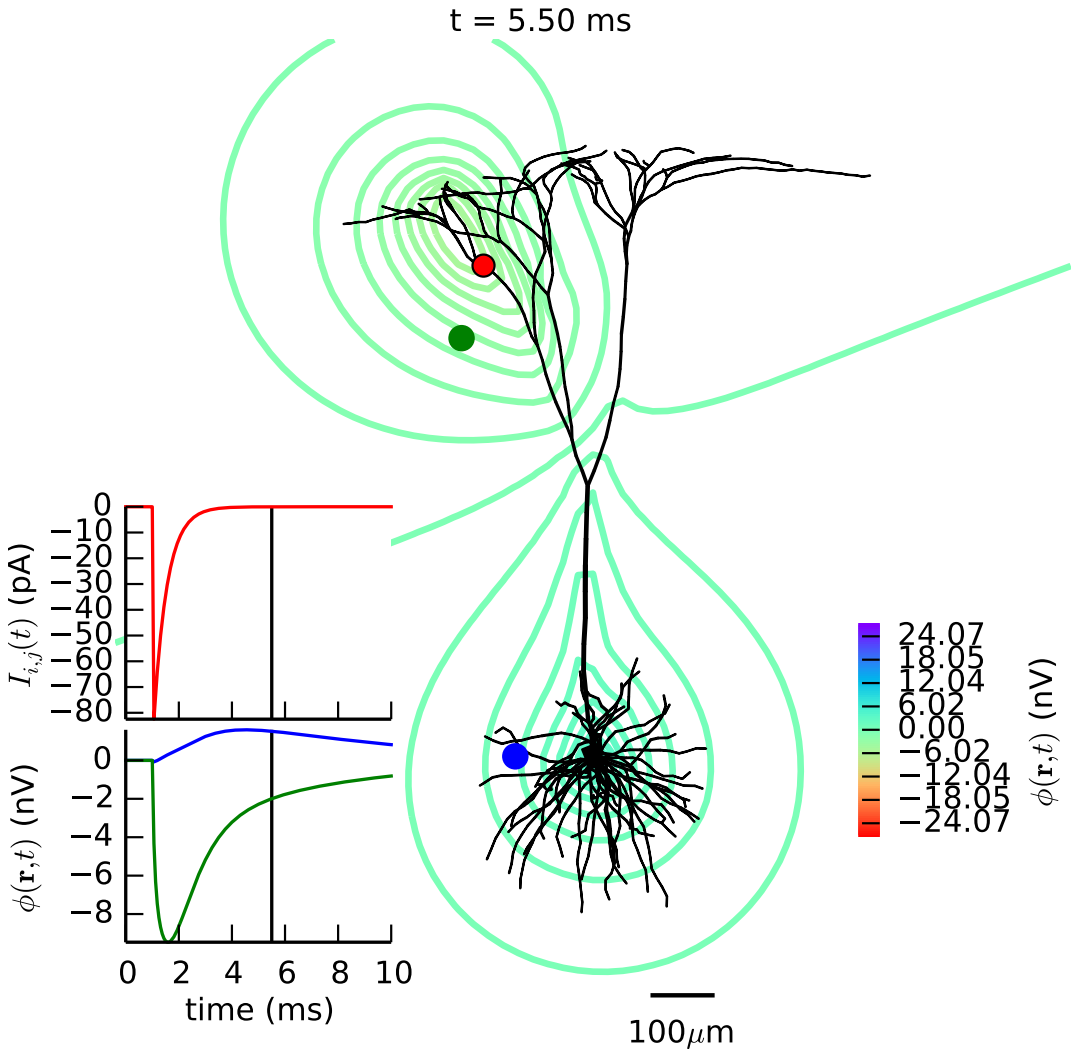
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



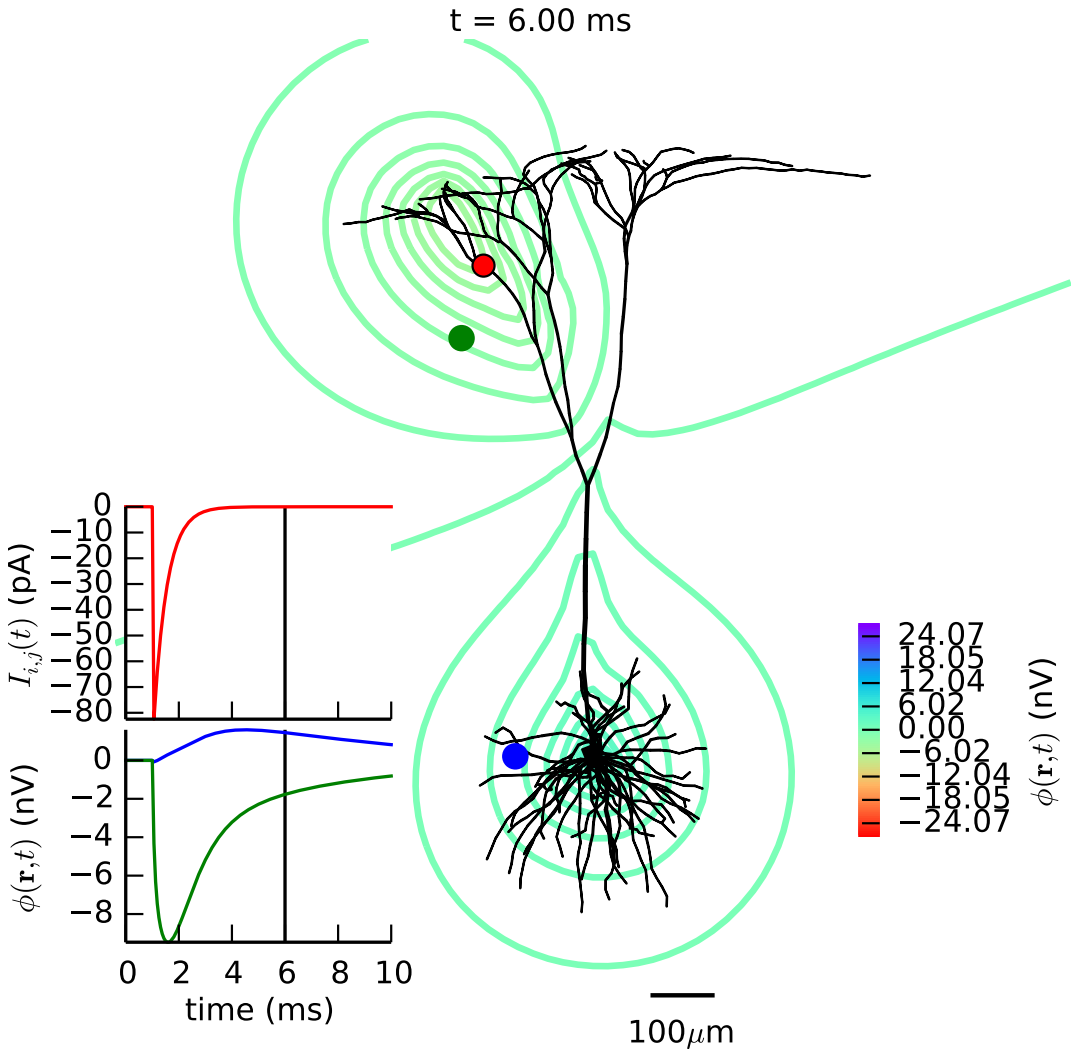
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



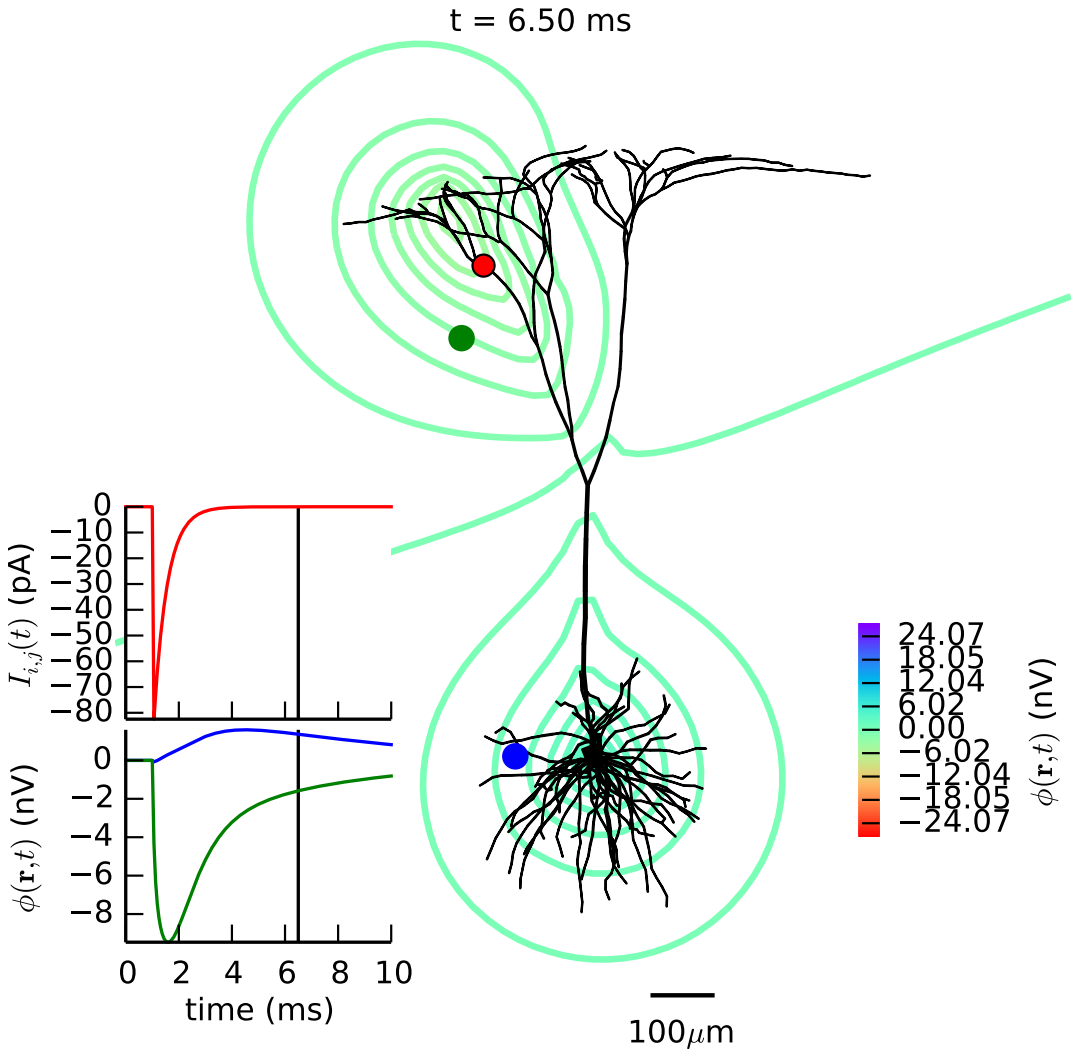
Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials



Passive propagation of synapse current input in passive cable model

Forward modeling of extracellular potentials

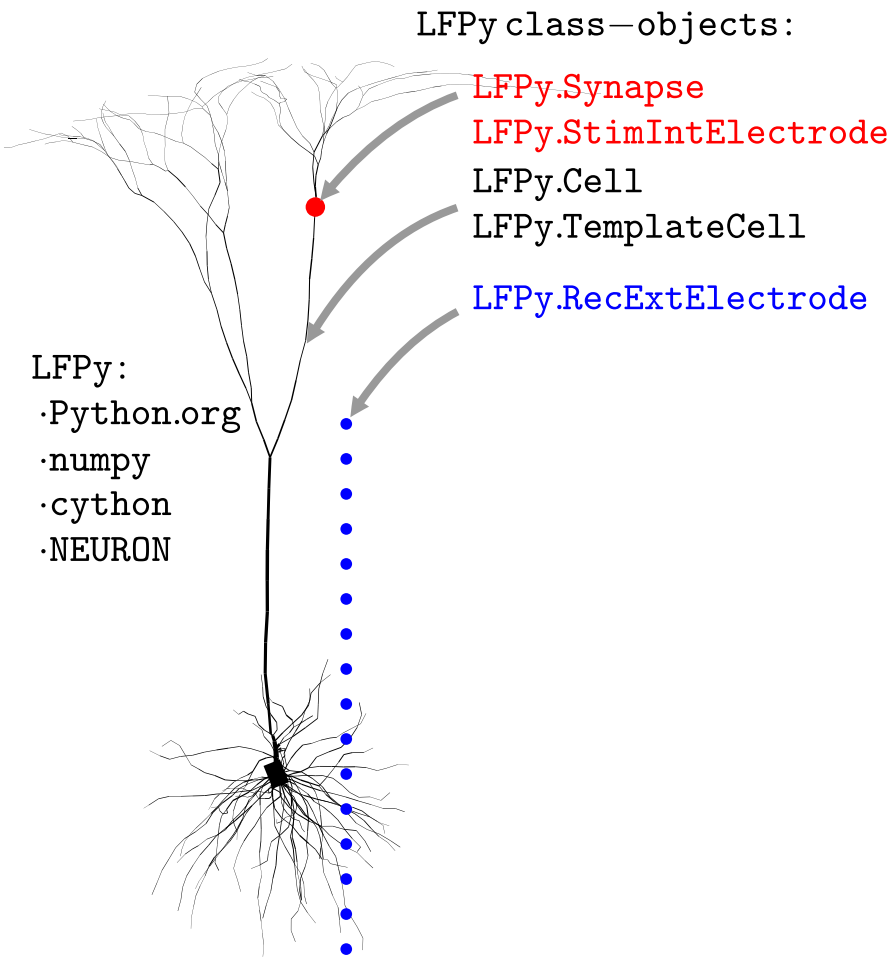


Passive propagation of synapse current input in passive cable model

LFPy - Introduction

Methods implementation:

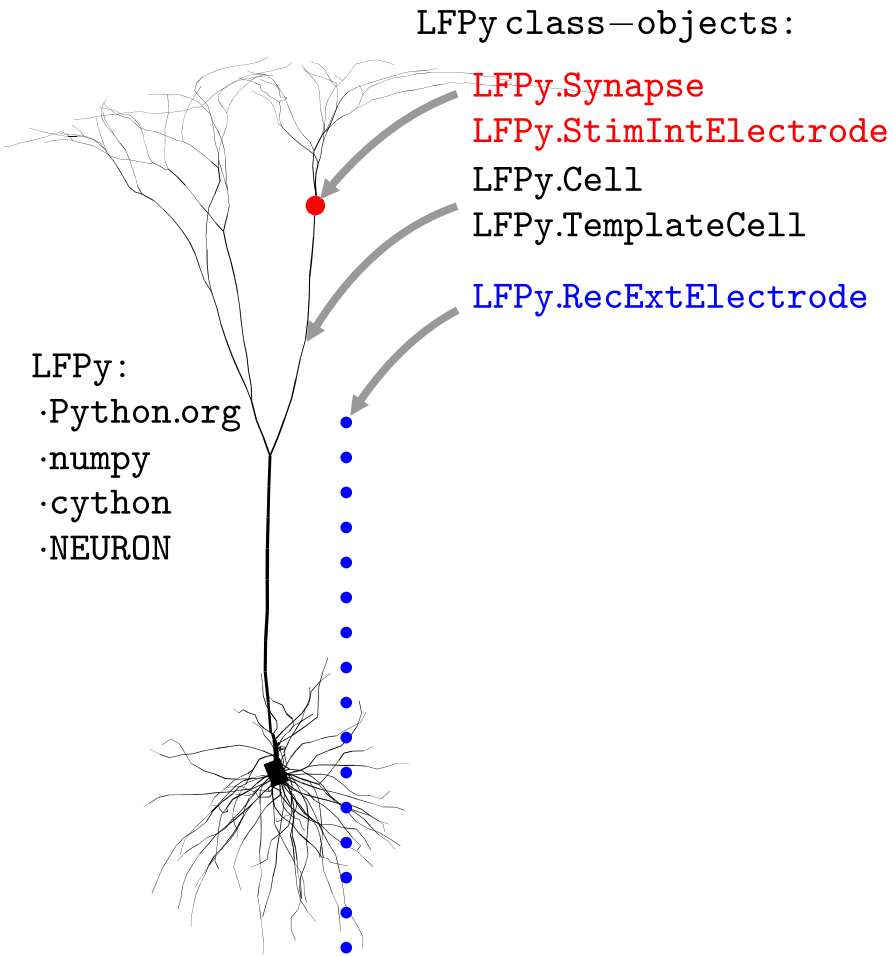
- Implemented in `Python`
- Uses `NEURON` under the hood
- Class objects represent:
  - cells
  - synapses
  - intracellular electrodes
  - extracellular electrodes
- Homepages w. documentation:
  - http://LFPy.github.io
  - http://github.com/LFPy



LFPy - Introduction

Why Python?

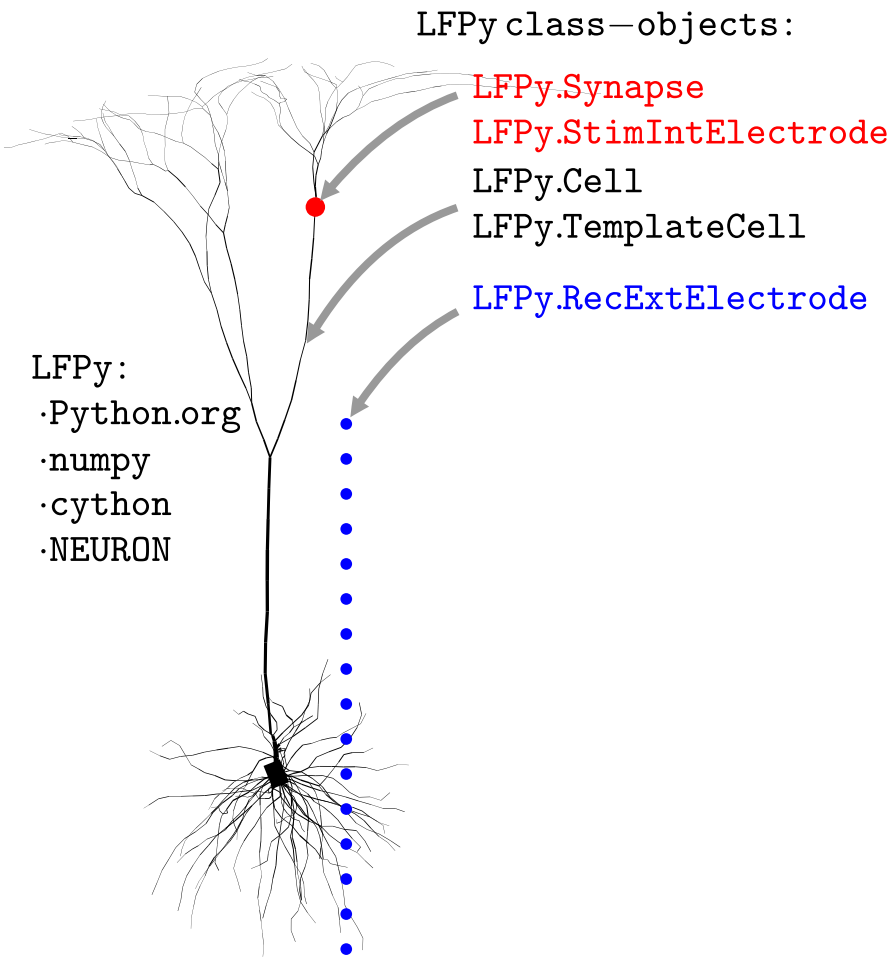
- Open source
- Easy, flexible coding
- Plethora of available packages for visualizations and analysis
- http://pypi.python.org:
  $\sim$ 63000 packages
- Interfacing other programs/languages:
  - NEURON (www.neuron.yale.edu)
  - NEST (www.nest-initiative.org)
  - BRIAN (http://briansimulator.org)
  - ...



LFPy - Introduction

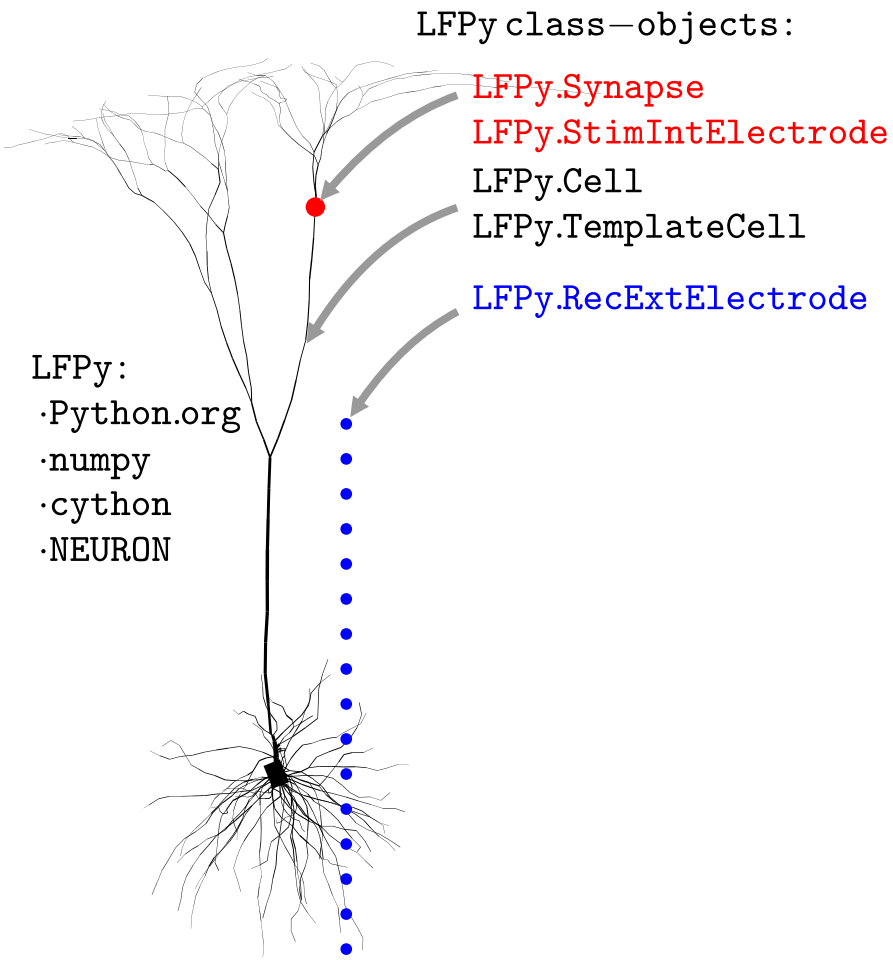
Python dependencies:

- Hard:
  - neuron
  - Cython
  - numpy
  - scipy
  - matplotlib
- Soft:
  - ipython (+notebook)
  - h5py
  - mpi4py



LFPy - Introduction

- Written for Python v2.6.* - 3.4.*
- Python distributions:
  - Anaconda
  - Enthought Canopy
  - Python(x,y)
  - ...
- **LFPy** supported on:
  - Unix, OS X
  - Linux
  - Windows



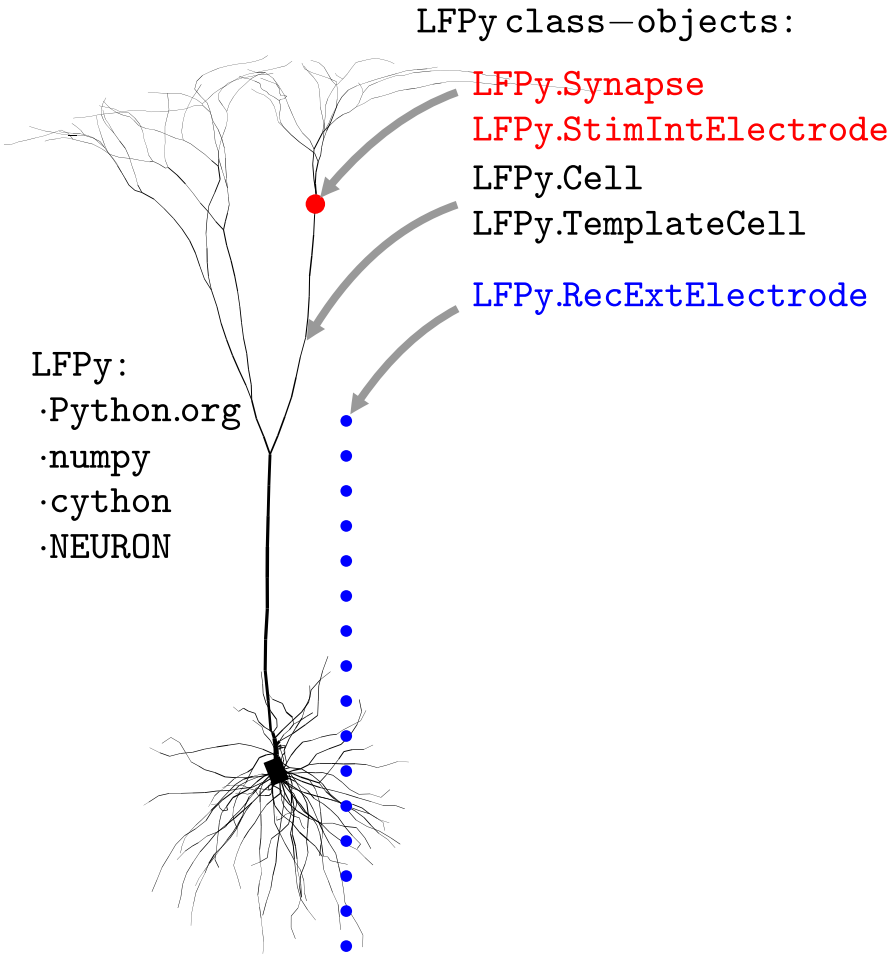
LFPy - Introduction

Easy installation method:

- `pip install LFPy --user`
- `easy_install LFPy --user`

From sources:

- Tar.gz-archive: https://pypi.python.org/packages/source/L/LFPy/LFPy-1.1.0.tar.gz
- From git (https://git-scm.com/): https://github.com/LFPy/LFPy.git
- Install:
  ```
  python setup.py install --user
  ```



LFPy - Installation

Test installation:

With `Python`:

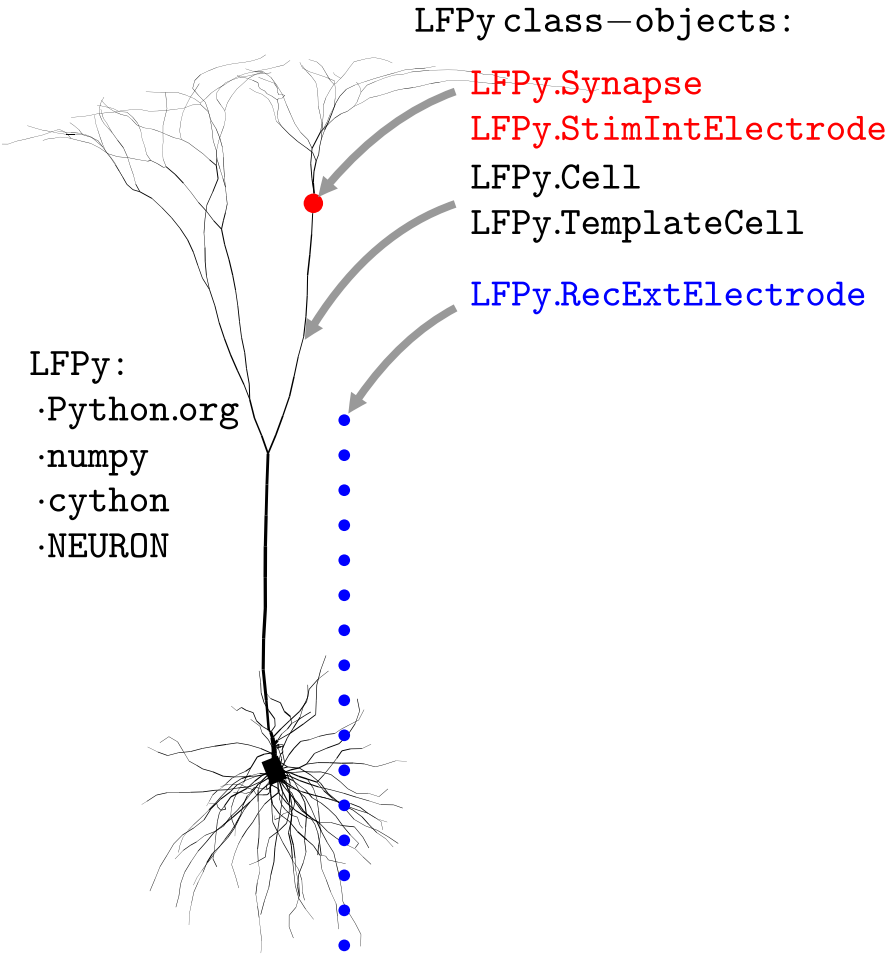
- `python -c "import LFPy"`
  `NEURON -- Release 7.3`
  `(1089:ecf32eddfbc7)...`

With `NEURON`:

- `nrngui -python -c "import LPFy"`
  `NEURON -- Release 7.3`
  `(1089:ecf32eddfbc7)...`

Unit test suite:

```
import LFPy
LFPy.test()
```



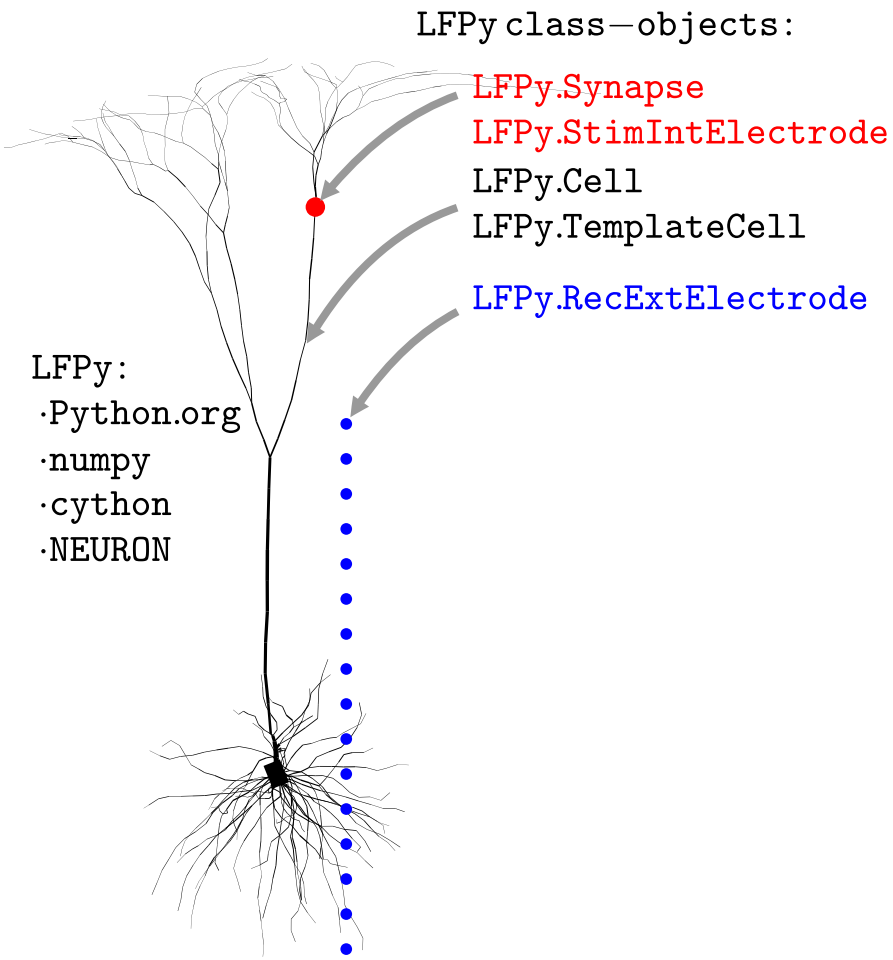
LFPy - Class overview

Main LFPy classes:

- Cell
- Synapse
- StimIntElectrode
- RecExtElectrode

Auxiliary classes and functions:

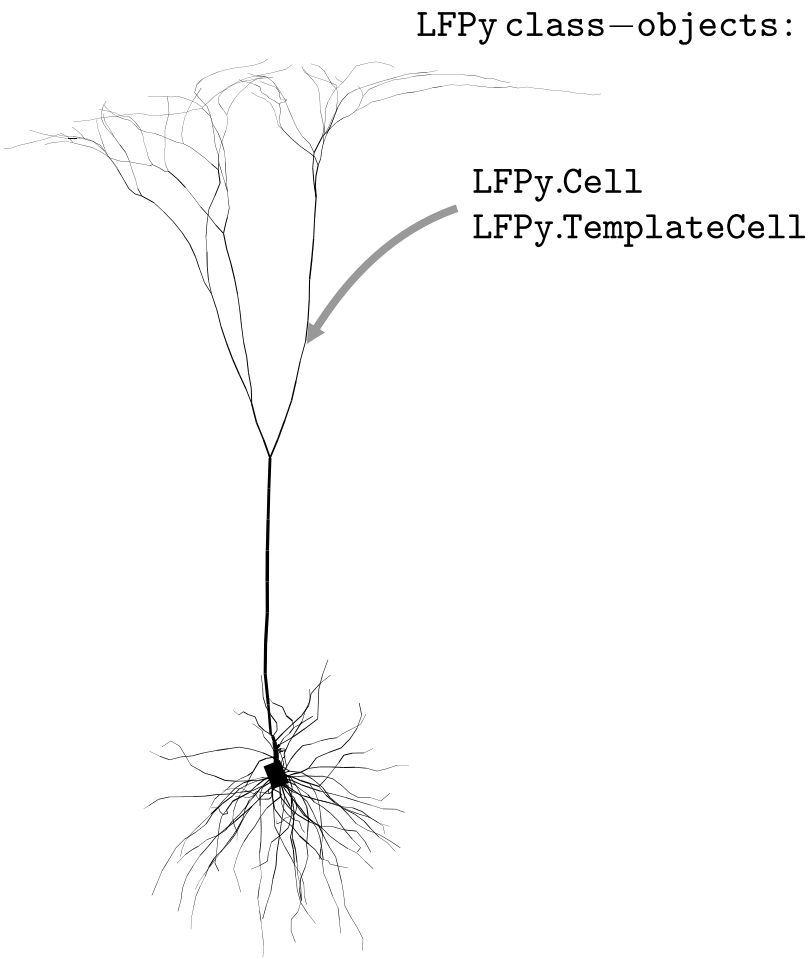
- TemplateCell
- lfpcalc.calc_lfp.*
- inputgenerators.*
- tools.*



`LFPy` - Class overview

`LFPy.Cell`:

- Uses `NEURON` under the hood
- Sets neuron properties:
  - neuron geometry
  - membrane mechanisms (`'pas'`, `'hh'`,...)
  - number of compartments (`'d_lambda'` rule; Hines&Carnevale. *Neuroscientist* (2001))
  - Sets cell location and rotation
- Simulation control
  - duration
  - record variables

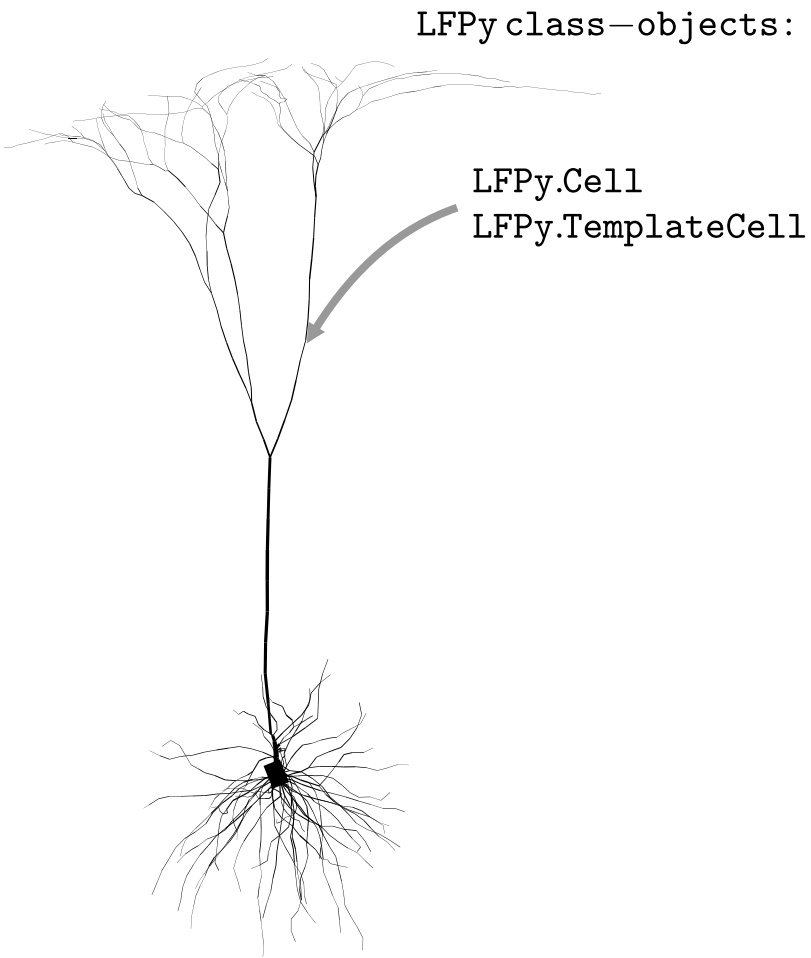


LFPy - Class overview

LFPy.Cell:

- Create parameter dictionary

```
# Define cell parameters
cell_parameters = dict(
    morphology='j4a.hoc',
    rm = 30000.,    #ohm cm2
    cm = 1.,        #uF cm-2
    Ra = 150.,      #ohm cm
    v_init = -65., #mV
    e_pas = -65.,   #mV
    tstopms = 100.,#ms
    custom_code = None,
    )
```



LFPy - Class overview

LFPy.Cell:

- Create `cell` object:

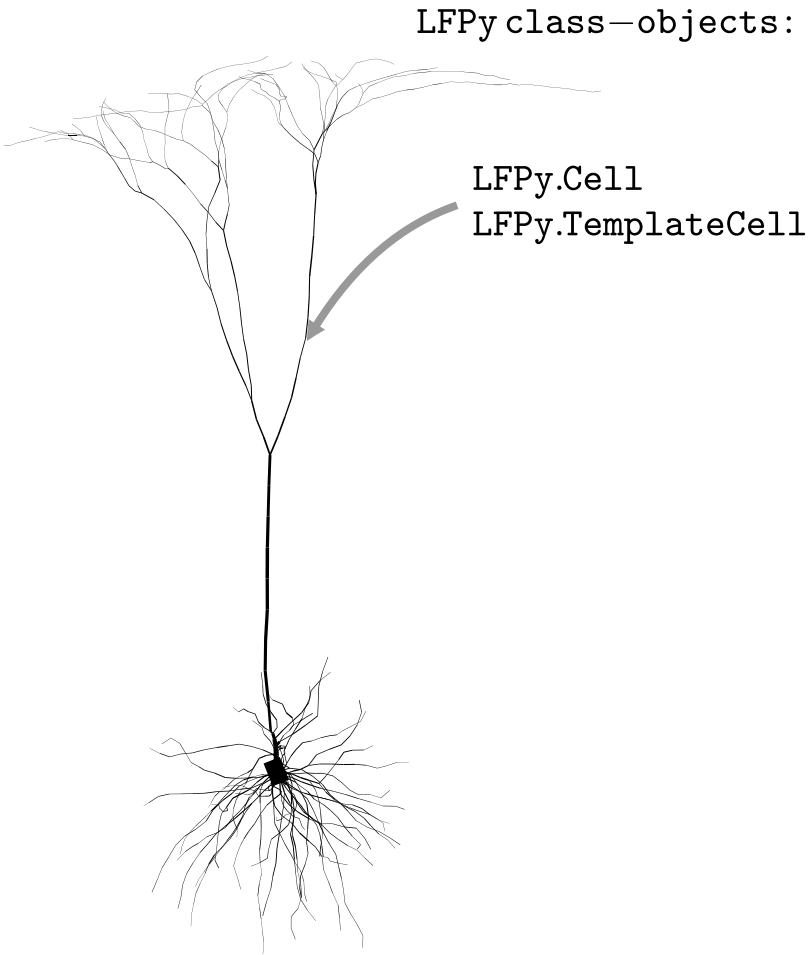
```
cell = LFPy.Cell(
          **cell_parameters)
```

- Position and align cell:

```
cell.set_pos(0., 0. ,0.)
cell.set_rotation(x=4.99,
          y=-4.33, z=3.14)
```

- (`cell` stimulation)
- simulate & plot `cell` response

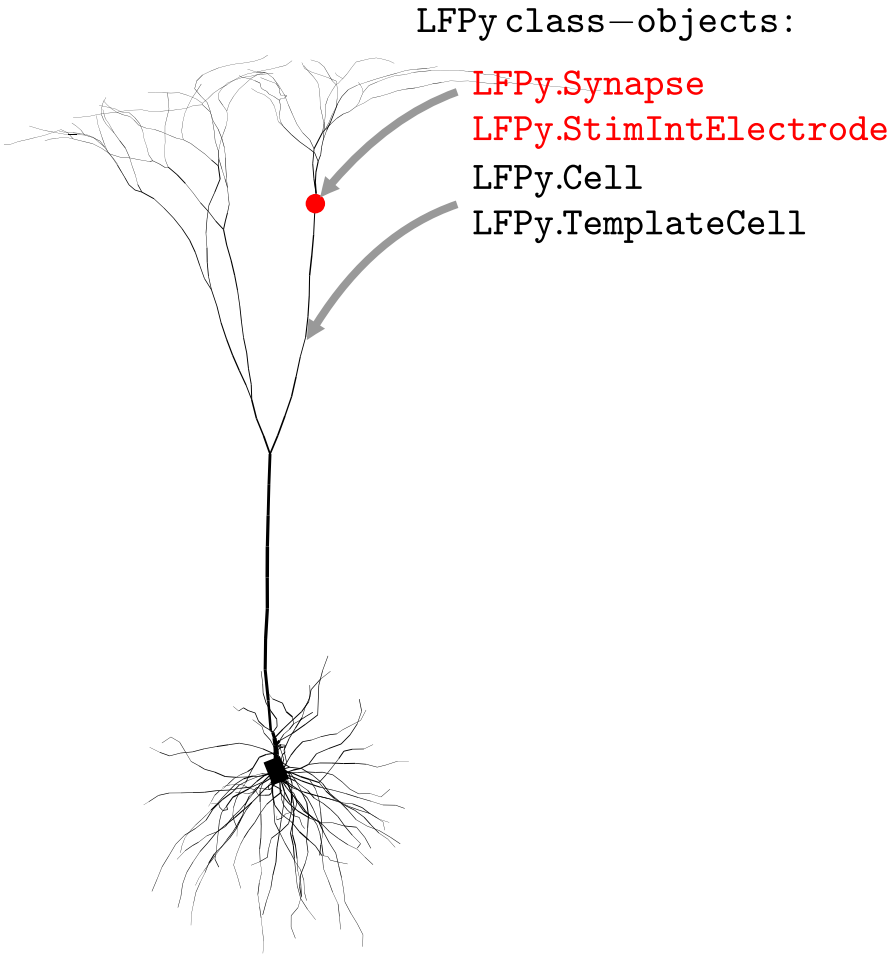
```
cell.simulate(rec_isyn=True/False,
              rec_istim=True/False)
plt.plot(cell.tvec, cell.somav)
```



LFPy - Class overview

LFPy.Synapse:

- Attach synapse-objects onto `cell`
- event-activated point currents
- Keyword arguments:
  - `cell`-object
  - compartment index (`idx`)
  - synapse type (`ExpSyn`, `Exp2syn`, `AlphaSynapse`)
  - mechanism arguments (`e`, `tau`, `weight`,...)
  - record synapse current (`record_current`)
- Feed in activation times (`np.array([20., ...])`)



LFPy - Class overview

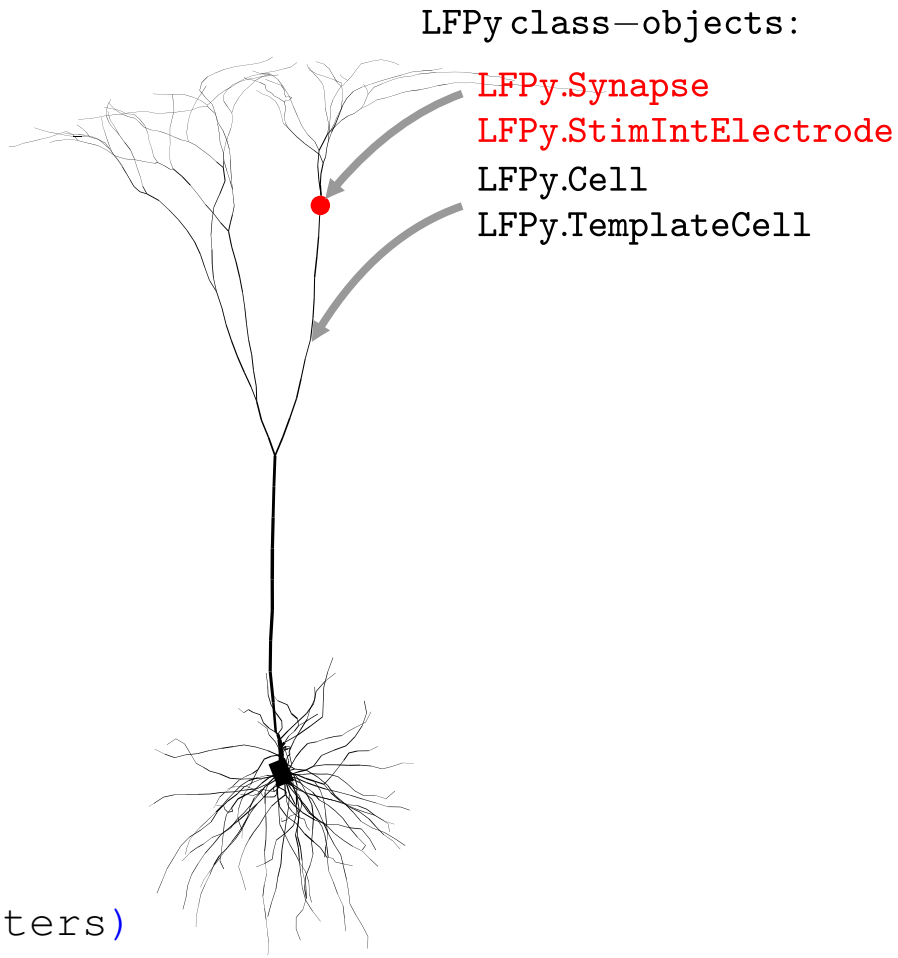
LFPy.Synapse:

- Define synapse parameters

```
synapse_parameters = dict(
    idx = cell.get_closest_idx(
                x=-200.,
                y=0.,
                z=800.),
    syntype = 'ExpSyn',
    e = 0.,
    tau = 5.,
    weight = .001,
    record_current = True,)
```

- Create synapse, set activation time

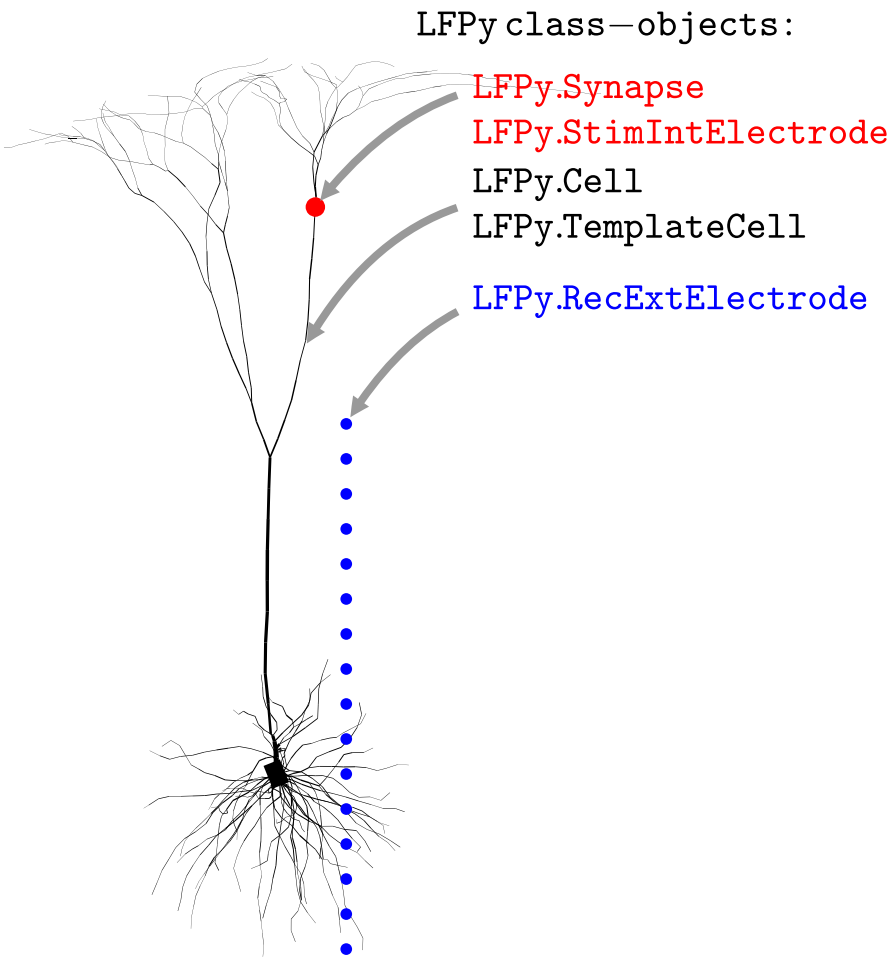
```
syn = LFPy.Synapse(cell,
                   **synapse_parameters)
syn.set_spike_times(np.array([20.]))
```



`LFPy` - Class overview

LFPy.RecExtElectrode:

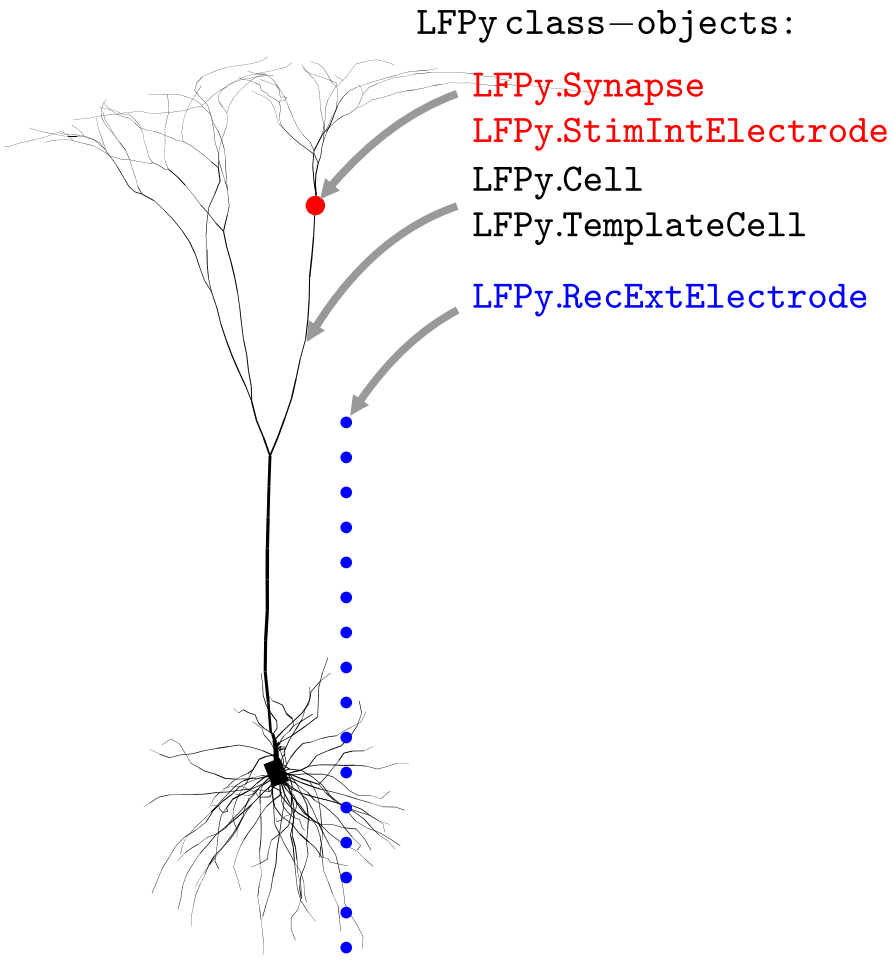
- Extracellular recording devices
- Main arguments:
  - `cell` objects (geometry, currents)
  - contact point coordinates `x,y,z`
  - extracellular conductivity `sigma`
  - `method` (pointsource/linesource)
- Optional:
  - contact radius `r`
  - contact surface normals `N`
  - *n*-point surface area averaged potential `n`



LFPy - Class overview

LFPy.RecExtElectrode:

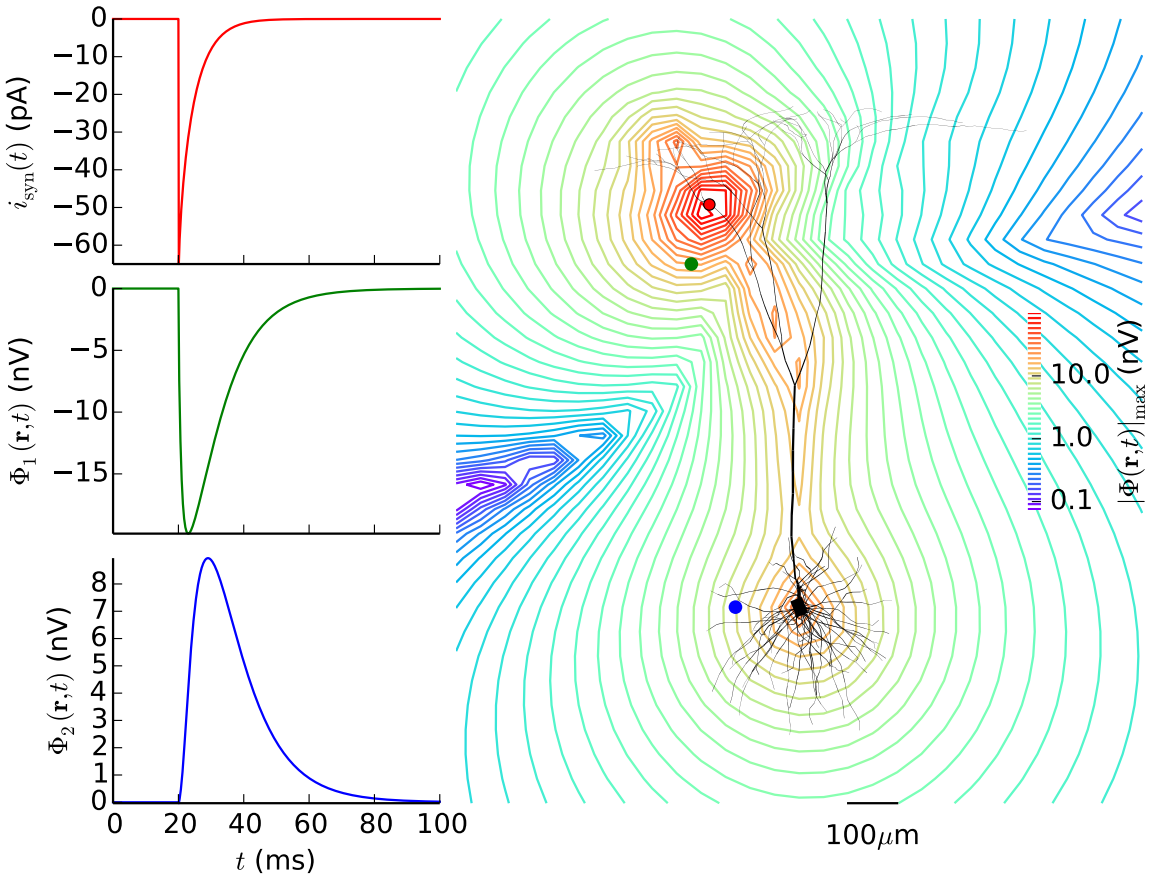
```
# Run simulation, record currents
cell.simulate(rec_imem=True,
              rec_isyn=True)
# Define electrode parameters
electrode_parameters = {
    'sigma' : 0.3,
    'x' : [-130., -220.],
    'y' : [   0.,    0.],
    'z' : [   0.,  700.],
}
# Create electrode object
electrode = LFPy.RecExtElectrode(
            cell,
            **electrode_parameters)
# Calculate LFPs
electrode.calc_lfp()
plt.plot(cell.tvec, electrode.LFP.T)
plt.show()
```



LFPy - Examples

`/path/to/LFPy/examples/example1.py`

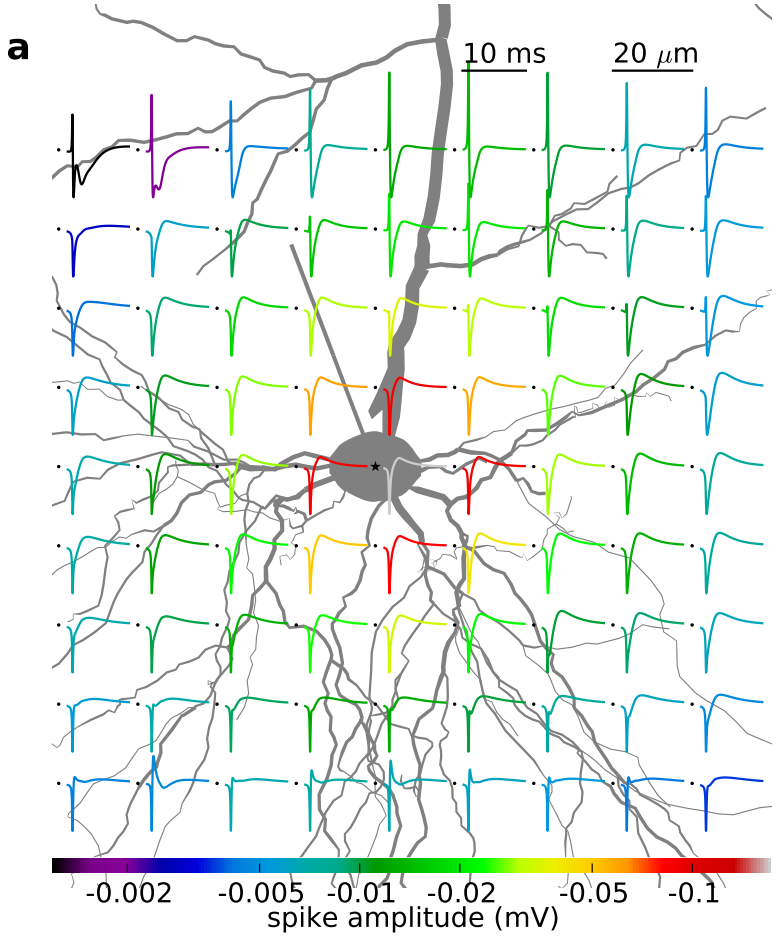
Apical synapse response, passive cable model

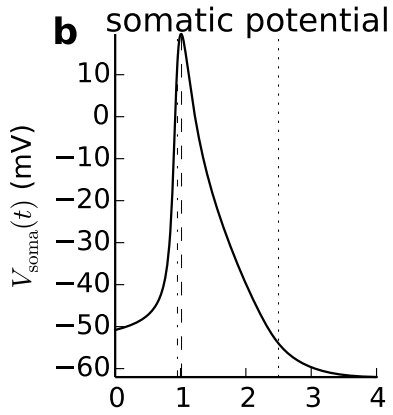


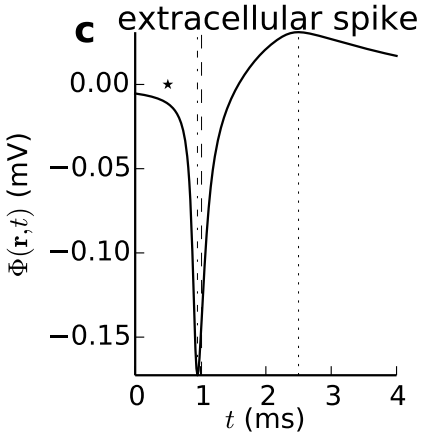
LFPy - Examples

`/path/to/LFPy/examples/example2.py`

Layer 5b action potential (Hay et al. 2011), `LFPy.TemplateCell`



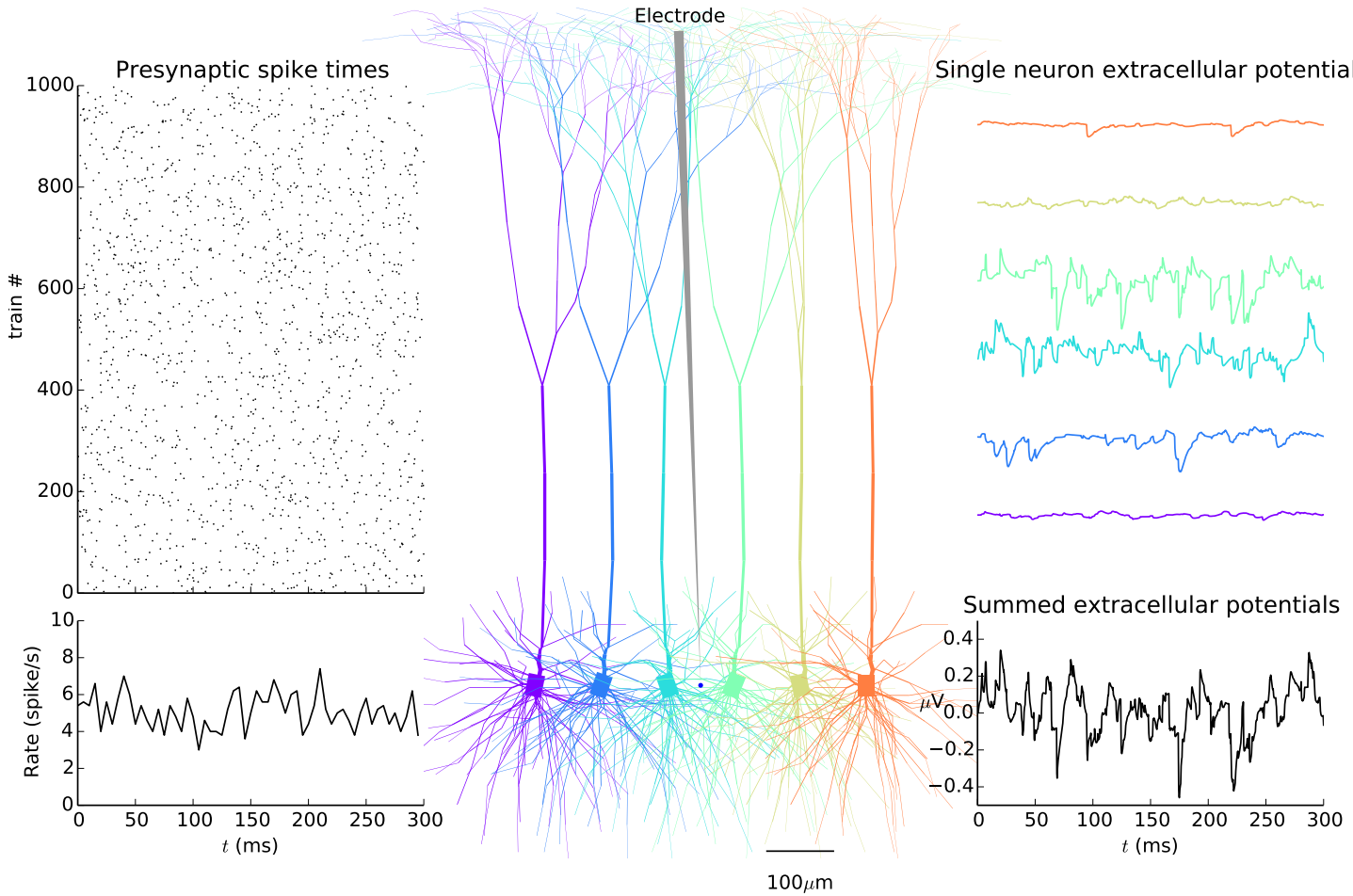




LFPy - Examples

`/path/to/LFPy/examples/example3.py`

Extracellular potentials of small model population, shared input



LFPy - Further reading and material

frontiers in
NEUROINFORMATICS

ORIGINAL RESEARCH ARTICLE
published: 16 January 2014
doi: 10.3389/fninf.2013.00041

LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons

Henrik Lindén[1,2†], Espen Hagen[1†], Szymon Łęski[1,3], Eivind S. Norheim[1], Klas H. Pettersen[1,4] and Gaute T. Einevoll[1*]

[1] *Department of Mathematical Sciences and Technology, Norwegian University of Life Sciences, Ås, Norway*
[2] *Department of Computational Biology, School of Computer Science and Communication, Royal Institute of Technology (KTH), Stockholm, Sweden*
[3] *Department of Neurophysiology, Nencki Institute of Experimental Biology, Warsaw, Poland*
[4] *CIGENE, Norwegian University of Life Sciences, Ås, Norway*

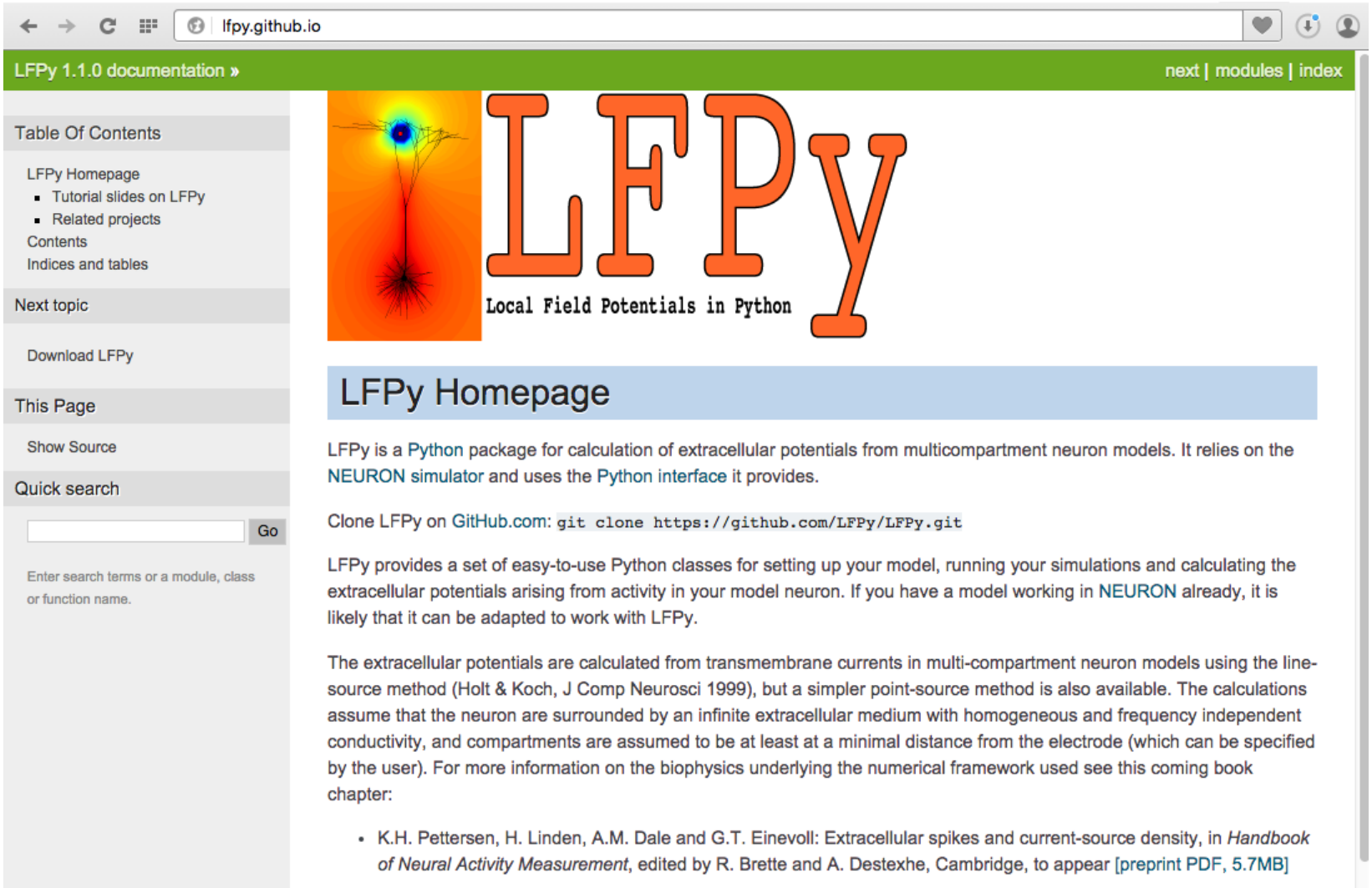
Edited by:
Andrew P. Davison, Centre National de la Recherche Scientifique, France

Reviewed by:
Nicholas T. Carnevale, Yale University School of Medicine, USA
Shyam Diwakar, Amrita University, India

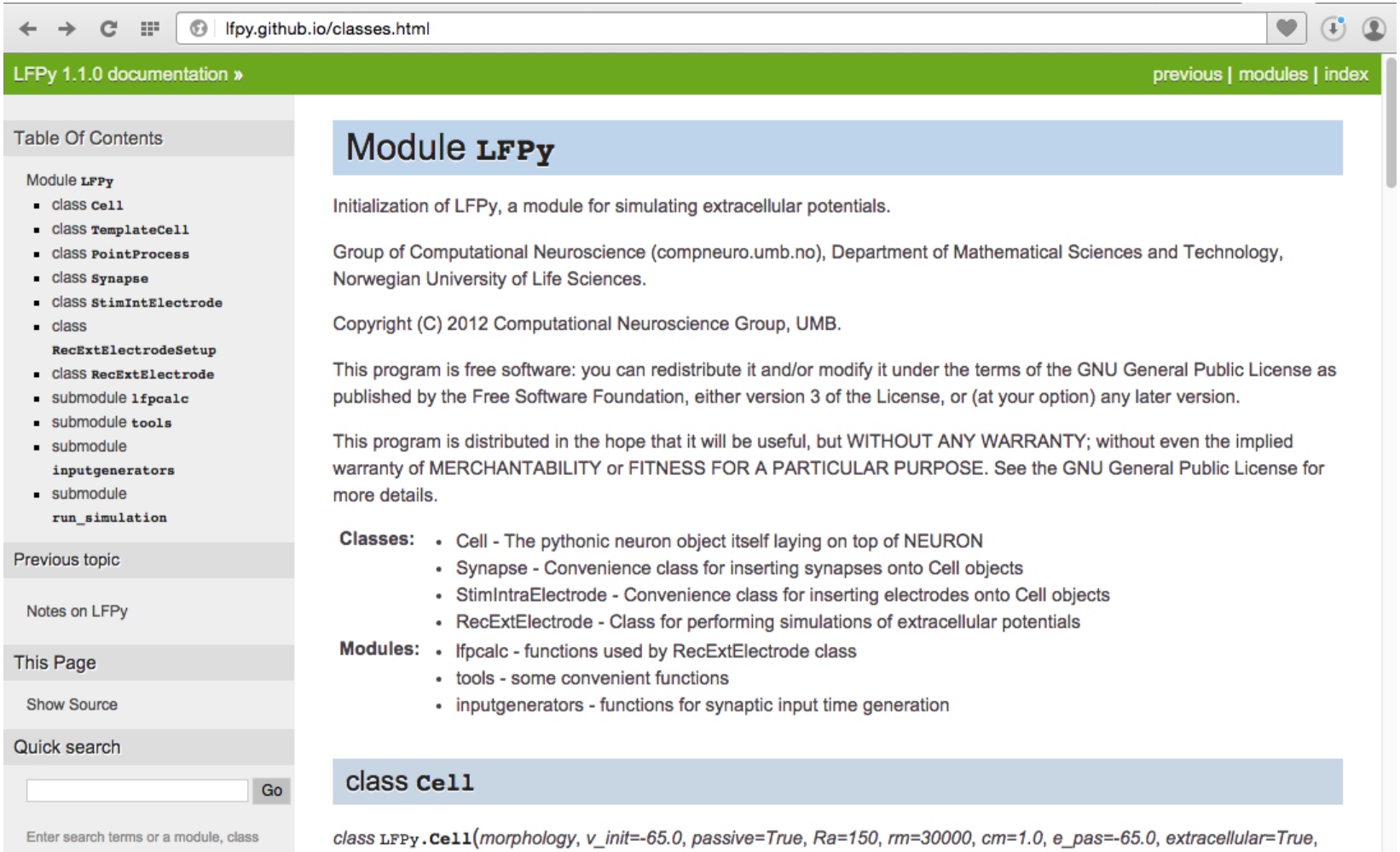
Electrical extracellular recordings, i.e., recordings of the electrical potentials in the extracellular medium between cells, have been a main work-horse in electrophysiology for almost a century. The high-frequency part of the signal ($\gtrsim$500 Hz), i.e., the *multi-unit activity (MUA)*, contains information about the firing of action potentials in surrounding neurons, while the low-frequency part, the *local field potential (LFP)*, contains information about how these neurons integrate synaptic inputs. As the recorded extracellular signals arise from multiple neural processes, their interpretation is typically ambiguous and

http://

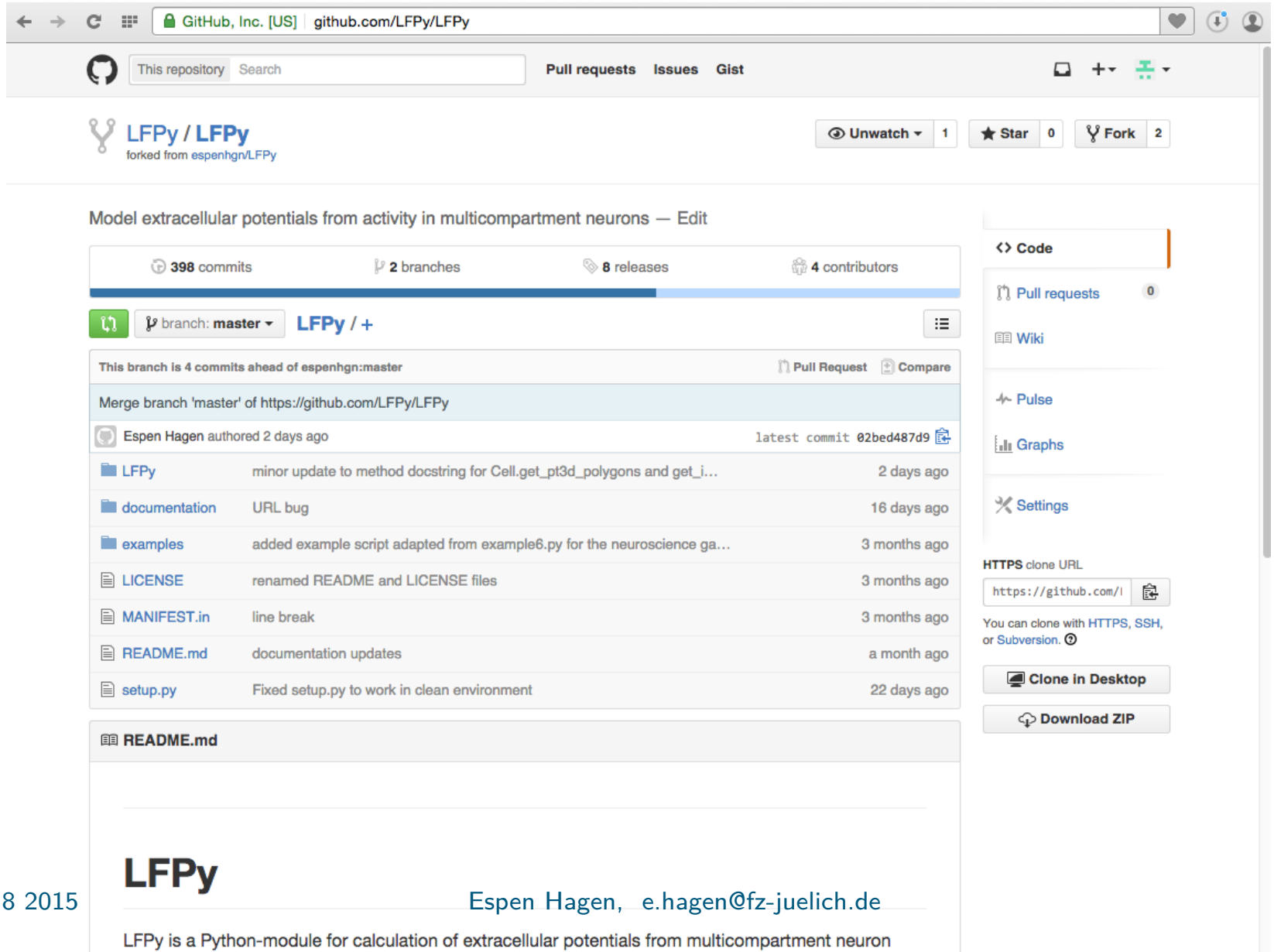
LFPy - Further reading and material



LFPy - Further reading and material



LFPy - Further reading and material



Questions?

Why hybrid model scheme? - `hybridLFPy`

Extracellular potentials in neural tissue

- Low-frequency part; the local field potential (LFP, $f \lesssim 100$ Hz)
  - Highly ambiguous, difficult to analyze
  - Large number of contributing sources
  - Reflect integration of synaptic inputs, synchrony, ...
  - Local and non-local network interactions
- High-frequency part ($f \gtrsim 500$ Hz)
  - Contain information of spiking activity
  - Single-unit activity (spikes)
  - Multi-unit activity (MUA)
  - Fewer contributing sources
  - Easier to interpret
  - (channel noise ++)

Why hybrid model scheme? - `hybridLFPy`

Extracellular potentials from electrophysiology

- Point-neuron network models:
  - Accurate predictions of population spiking activity
  - Efficient, easy to constrain models
  - Poor predictors of extracellular signals (e.g., `rate != LFP`)
- Biophysically detailed network models
  - Demanding to implement
  - Difficult to constrain
  - Computationally expensive
  - Extracellular signal predictions rare

Why hybrid model scheme? - `hybridLFPy`

Extracellular potentials from electrophysiology

- Large-scale models necessary:
  - LFP reflects synaptic input also generated by remote populations (cortical & subcortical areas)
  - Theoretical description of the LFP needs to account for:
    - Anatomical and electrophysiological features of proximal neurons
    - Activity in the local microcircuitry
    - Large-scale (O(brain)) neuronal circuitry generating synaptic input
  - Reducibility of asynchronous networks is fundamentally limited (**O1**: Albada et al. (2015), http://arxiv.org/abs/1411.4770v3):
    - Methods to conserve 1st order statistics of network dynamics exist
    - Limitations arise if also 2nd-order statistics are to be maintained
    - Preserving correlations require preserving effective connectivity
    - Adjust synapse strength $j \propto 1/K$ and background input variance

Why hybrid model scheme? - `hybridLFPy`

Extracellular potentials from electrophysiology

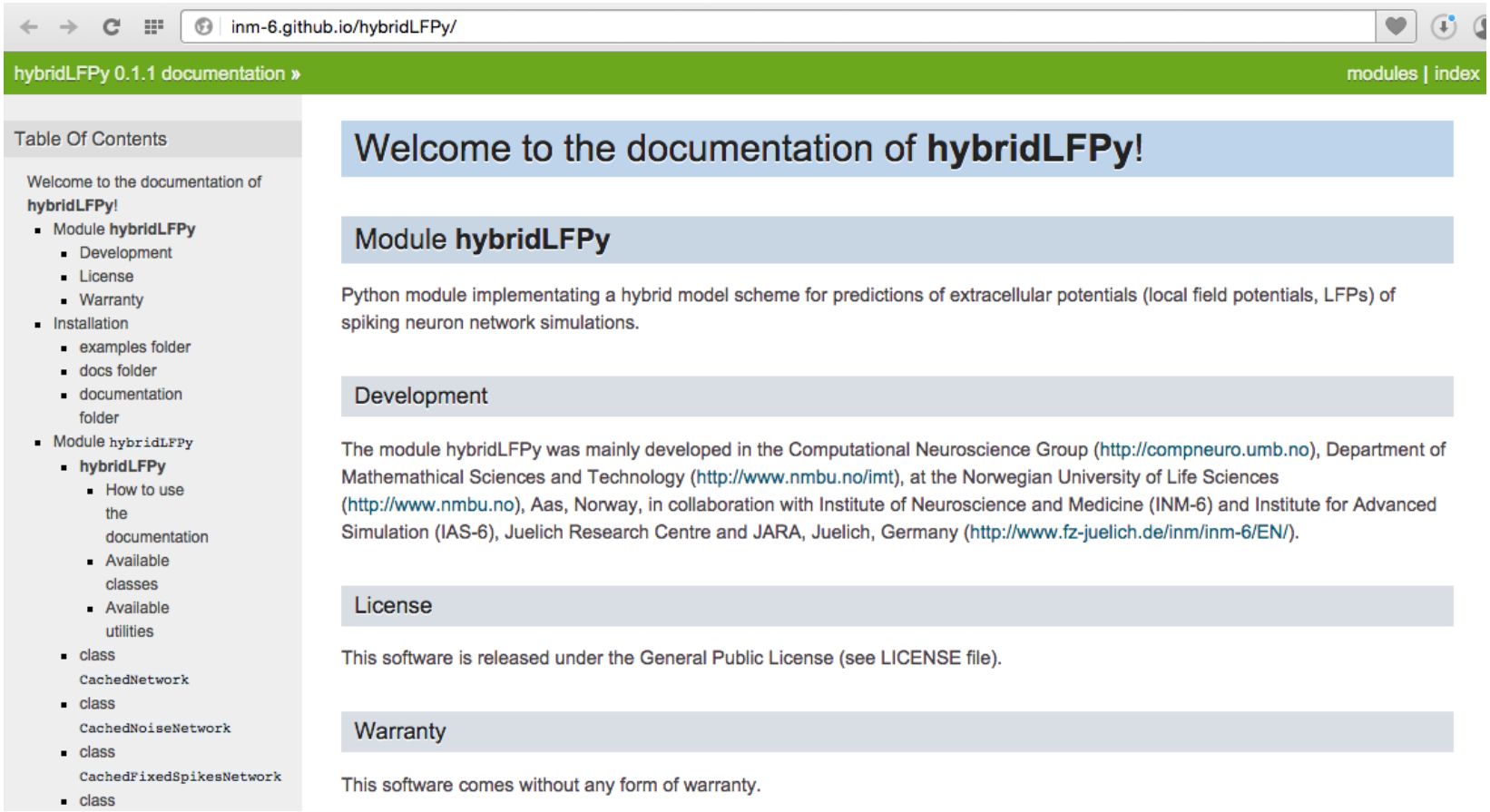
- Here:
  Hybrid scheme interfacing point-neuron network models with biophysically justified forward modelling scheme for extracellular potentials
  - LFP, CSD
  - (EEG, MEG, VSDi, ...)
- Benefits of hybrid scheme:
  - Relate network spiking activity to LFPs
  - Introduce spatial features (morphology, connectivity)
  - Simplified, passive membrane model
  - Preserve network features (cell count, synapse model ...)
  - Massive parallelism not necessary

`hybridLFPy` - Python package overview

Our hybrid scheme for LFP predictions is public:

- Documentation: http://inm-6.github.io/hybridLFPy
- Sources: https://github.com/INM-6/hybridLFPy
- Main classes and functions:
  - `hybridLFPy.CachedNetwork`
  - `hybridLFPy.Population`
  - `hybridLFPy.Postprocess`
  - `hybridLFPy.setup_file_dest`
- Example files - `/path/to/hybridLFPy/examples`
  - Network model: `/brunel_alpha_nest.py`
  - Hybrid model application: `/example_brunel.py`
- Python dependencies: `LFPy`, `nest`, `mpi4py`, `h5py`, `sqlite3`, `NeuroTools`

hybridLFPy - Python package overview



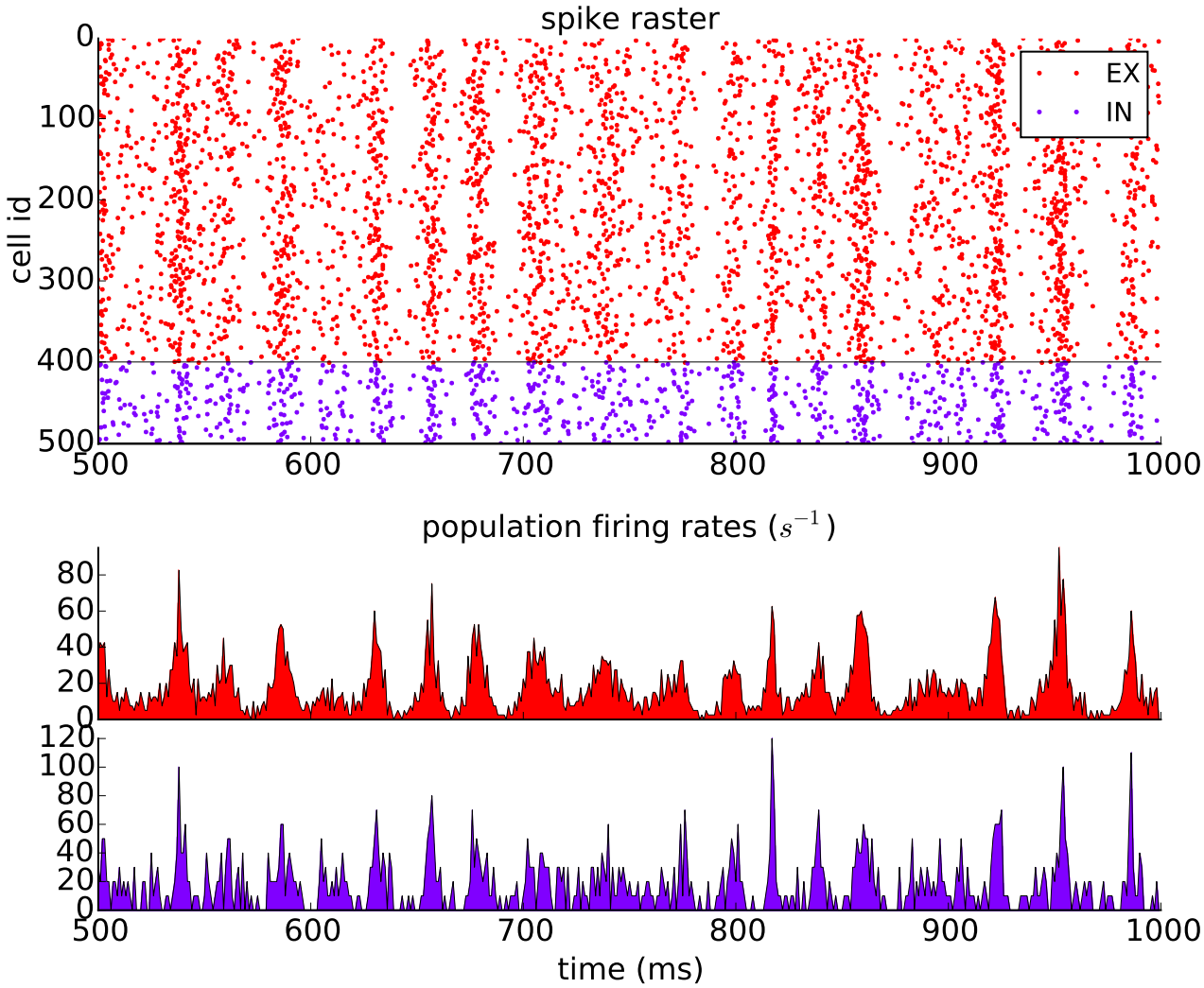
http://inm-6.github.io/hybridLFPy

`hybridLFPy` - Application with E-I network

Network model

- Two-population E-I network (Brunel, *J Comput Neurosci* (2000))
  - leaky integrate-and-fire (LIF) neurons
  - current based synapses
  - alpha-shaped PSCs
  - adapted from NEST (github.com/nest/nest-simulator) example: `/pynest/examples/brunel_alpha_nest.py`
- Modifications:
  - $N_{\mathrm{E}} + N_{\mathrm{I}} = 500$ neurons
  - $J = 1.$ mV
  - $g = -6.$
  - External Poisson spike generators removed
  - DC current input ($I_{\mathrm{DC}} \approx 300$ nA)
  - All spike events dumped to disk

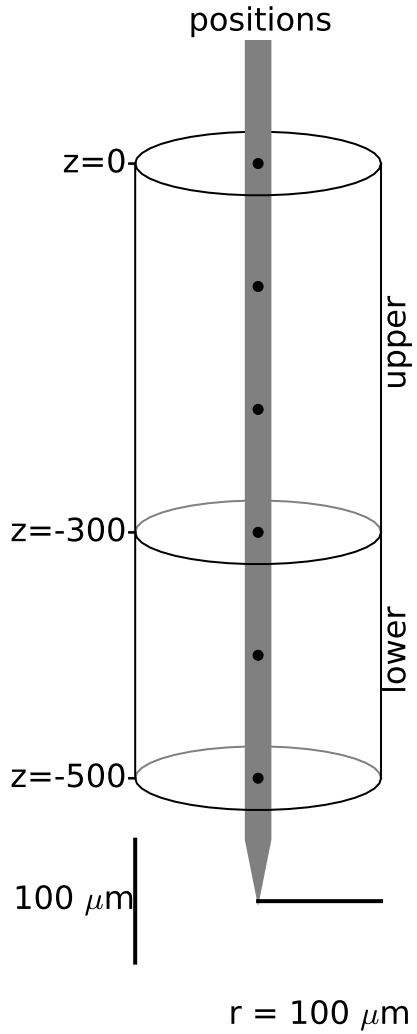
hybridLFPy - Application with E-I network



`hybridLFPy` - Application with E-I network

Model configuration

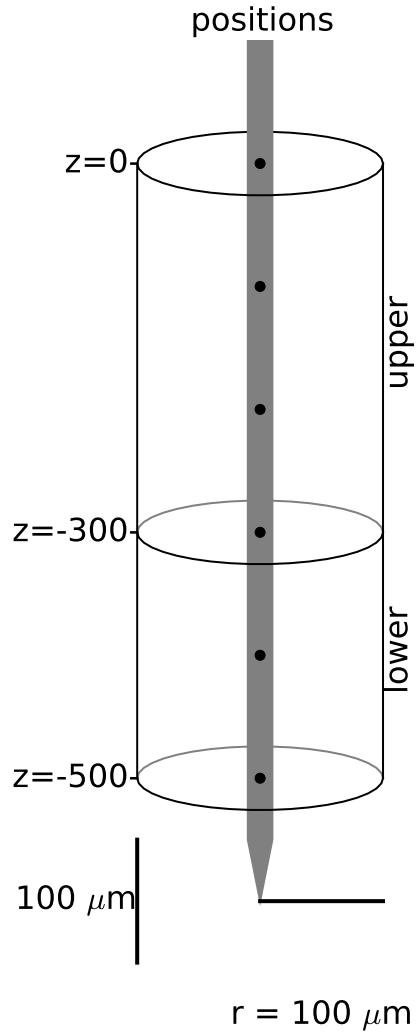
- "Layers":
  - upper $z \in [-300, 0]$ μm
  - lower $z \in [-500, -300]$ μm



`hybridLFPy` - Application with E-I network

Model configuration

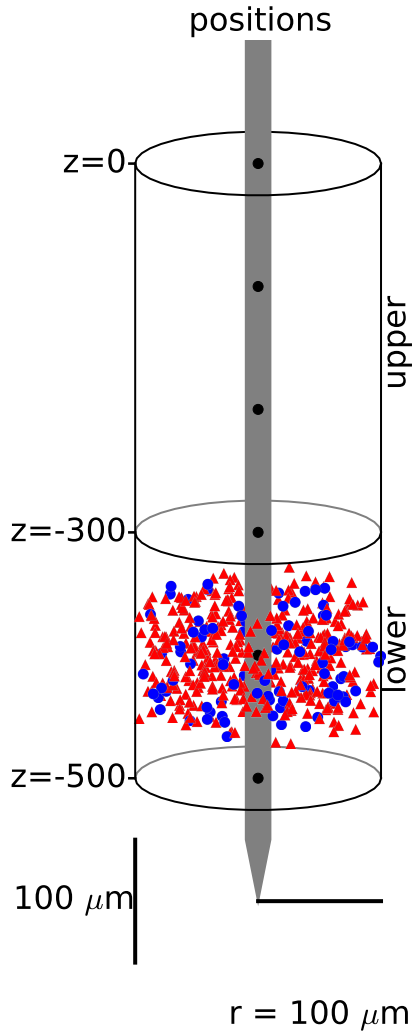
- "Layers":
  - upper $z \in [-300, 0]$ μm
  - lower $z \in [-500, -300]$ μm
- Measurements:
  - 6-channel laminar "electrode"
  - 100 μm between contacts
  - (laminar) current-source density (CSD)
  - local field potentials (LFP)



`hybridLFPy` - Application with E-I network

Model configuration

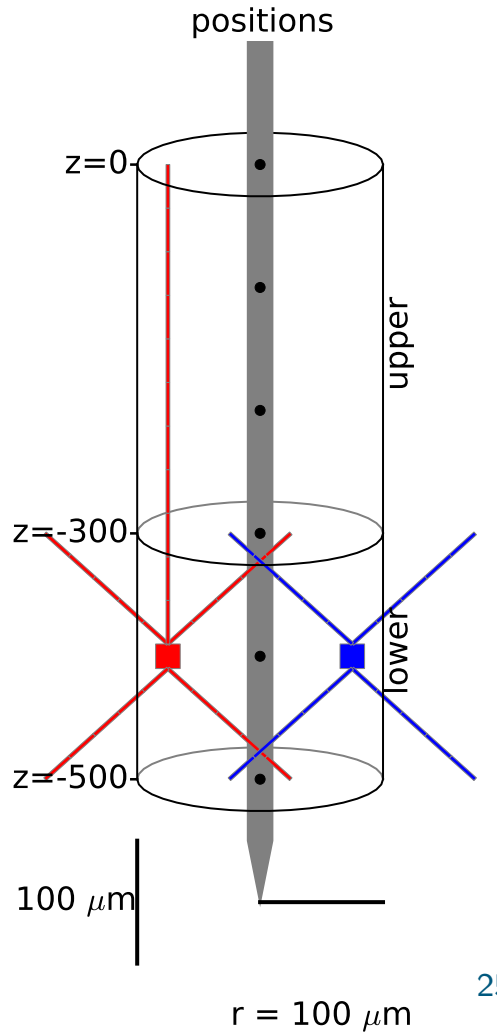
- "Layers":
  - upper $z \in [-300, 0]$ μm
  - lower $z \in [-500, -300]$ μm
- Measurements:
  - 6-channel laminar "electrode"
  - 100 μm between contacts
  - (laminar) current-source density (CSD)
  - local field potentials (LFP)
- Random cell positions
  - $z \in [-450, -350]$ μm
  - $R < 100$ μm



`hybridLFPy` - Application with E-I network

Model configuration

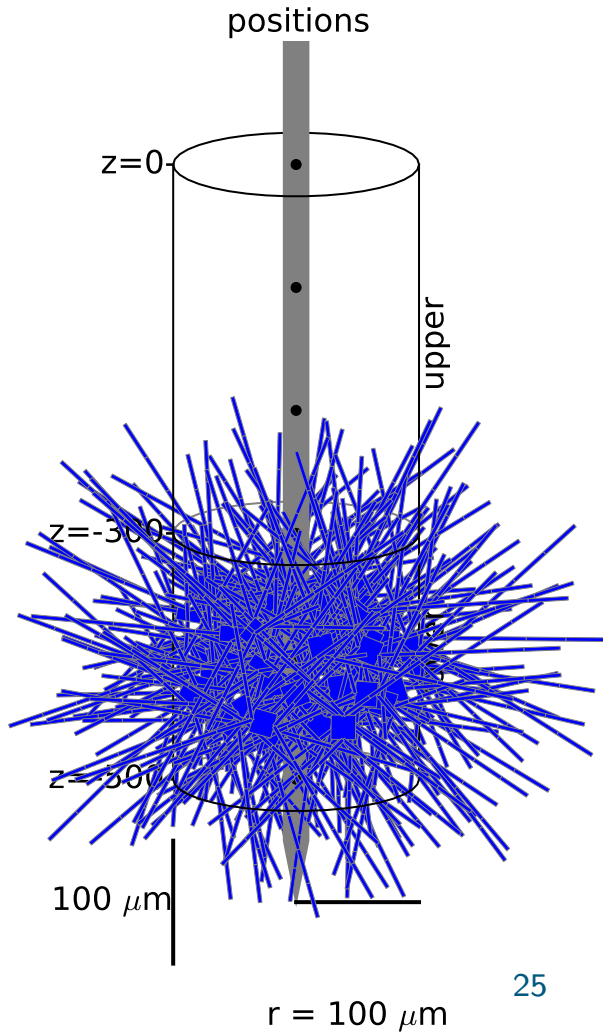
- Simplified morphologies:
  - "EX" - "pyramidal neuron"
  - "IN" - "interneuron"
  - passive cable models
  - membrane time constant of LIF neurons
  - spatially discretized into compartments



`hybridLFPy` - Application with E-I network

Model configuration

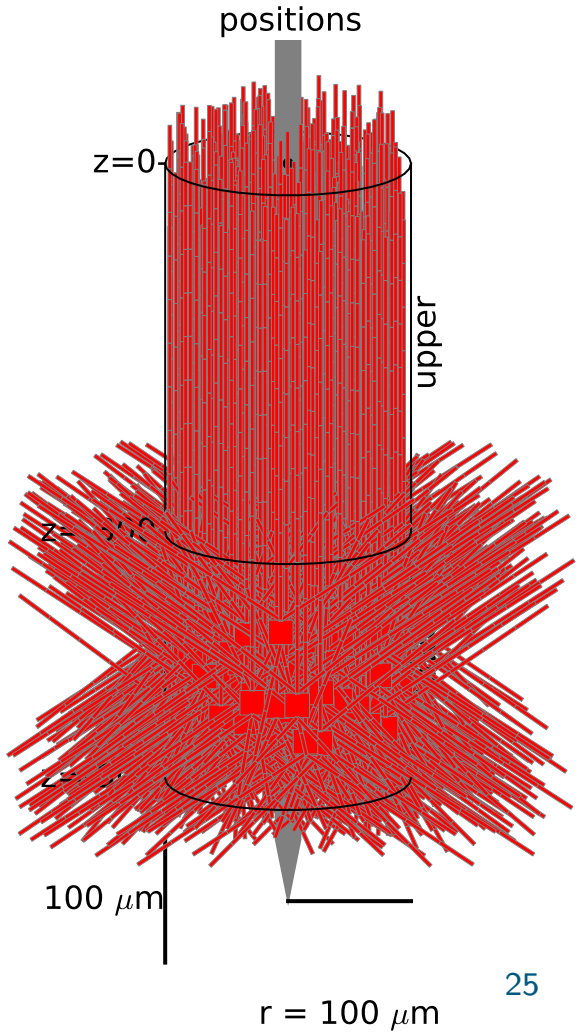
- Simplified morphologies:
  - "EX" - "pyramidal neuron"
  - "IN" - "interneuron"
  - passive cable models
  - membrane time constant of LIF neurons
  - spatially discretized into compartments
- IN - Random rotation around x, y, z-axis



`hybridLFPy` - Application with E-I network

Model configuration

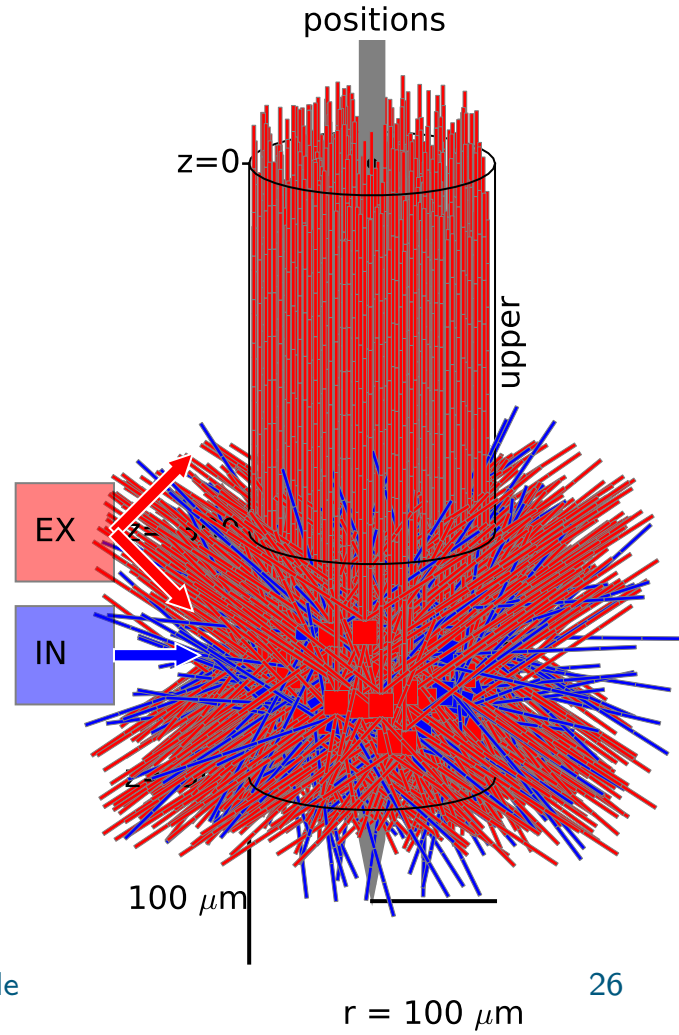
- Simplified morphologies:
  - "EX" - "pyramidal neuron"
  - "IN" - "interneuron"
  - passive cable models
  - membrane time constant of LIF neurons
  - spatially discretized into compartments
- IN - Random rotation around x, y, z-axis
- EX - Vertically aligned apical stick
  random rotation around z-axis



`hybridLFPy` - Application with E-I network

Model configuration

- Connectivity:
  - Mean in-degree from the network
  - EX→EX: 50/50% in upper/lower layer
  - EX→IN: 100% in lower layer
  - IN→EX: 100% in lower layer
  - IN→IN: 100% in lower layer
  - Only IN inputs on soma
  - Within layers - conn.-prob. normalized by surface area
- Synapse model
  - inherited from network
  - `NEURON NMODL` language `examples/alphaisyn.mod`



hybridLFPy - Application with E-I network

example_brunel.py initialization:

```
#import necessary classes and functions
...
from hybridLFPy import PostProcess, Population, CachedNetwork
from hybridLFPy import setup_file_dest
from NeuroTools.parameters import ParameterSet
from mpi4py import MPI

#MPI Initialization
COMM = MPI.COMM_WORLD
SIZE = COMM.Get_size()
RANK = COMM.Get_rank()

#Parameters defined in pynest example script,
#brunel_alpha_nest.py, adapted from NEST v2.4.1 release:
import brunel_alpha_nest as BN
#note: will not execute model
```

hybridLFPy - Application with E-I network

File hierarchy:

- simulation_output_example_brunel/
  - simscripts/
  - cells/
  - populations/
  - spiking_output_path/
  - figures/

hybridLFPy - Application with E-I network

example_brunel.py file hierarchy:

```
PS = ParameterSet(dict()) #initialize
savefolder = 'simulation_output_example_brunel'
PS.update(
    #Main destination destination
    savefolder = savefolder,
    #copy of simulation files
    sim_scripts_path = os.path.join(savefolder, 'sim_scripts'),
    #single-cell output
    cells_path = os.path.join(savefolder, 'cells'),
    #destination compound signals
    populations_path = os.path.join(savefolder, 'populations'),
    #spike output from the network model
    spike_output_path = BN.spike_output_path,
    #figure destination
    figures_path = os.path.join(savefolder, 'figures')
))
#set up file destination, clear old results
setup_file_dest(PS, clearDestination=True)
```

hybridLFPy - Application with E-I network

example_brunel.py parameter setup:

```
#population (and cell type) specific parameters
PS.update(dict(
    #population names
    X = ["EX", "IN"],
    #population-specific LFPy.Cell parameters
    cellParams = dict(
        #excitory cells
        EX = dict(
            morphology = 'morphologies/ex.hoc',
            v_init = BN.neuron_params['E_L'],
            rm = BN.neuron_params['tau_m'] * 1E3 / 1.,
            cm = 1.0, Ra = 150,
            e_pas = BN.neuron_params['E_L'],
            timeres_NEURON = BN.dt,
            timeres_python = BN.dt,
            tstopms = BN.simtime,),
        #inhibitory cells
        IN = dict(
            morphology = 'morphologies/in.hoc', ...))
```

hybridLFPy - Application with E-I network

example_brunel.py parameter setup:

```
#population (and cell type) specific parameters
PS.update(dict(
    #cylindrical model populations
    populationParams = dict(
        EX = dict(
            number = BN.NE,
            radius = 100,
            z_min = -450,
            z_max = -350,
            min_cell_interdist = 1.,),
        IN = dict(number = BN.NI, ...),
    ),
    #set the boundaries between the
    #"upper" and "lower" layer:
    layerBoundaries = [[0.,   -300],
                       [-300, -500]]),
```

hybridLFPy - Application with E-I network

example_brunel.py parameter setup:

```
#set the geometry of the virtual recording device
PS.update(dict(
    electrodeParams = dict(
            #contact locations:
            x = [0]*6,
            y = [0]*6,
            z = [x*-100. for x in range(6)],
            #extracellular conductivity:
            sigma = 0.3,
            #contact surface normals, radius, n-point averaging
            N = [[1, 0, 0]]*6,
            r = 5,
            n = 20,
            seedvalue = None,
            #dendrite line sources, soma as sphere source (Linden2014)
            method = 'som_as_point',
            #no somas within the constraints of the "electrode shank":
            r_z = [[-1E199, -600, -550, 1E99],[0, 0, 10, 10]],)))
```

hybridLFPy - Application with E-I network

`example_brunel.py` parameter setup:

```
#layer- and population-specific
#connection parameters
PS.update(dict(
    #number of connections per layer per cell
    #from each presynaptic population
    k_yXL = dict(
        EX = [[int(0.5*BN.CE), 0],
              [int(0.5*BN.CE), BN.CI]],
        IN = [[0,      0],
              [BN.CE, BN.CI]],),

    #Connection weights (current amplitudes)
    J_yX = dict(
        EX = [BN.J_ex*1E-3, BN.J_in*1E-3],
        IN = [BN.J_ex*1E-3, BN.J_in*1E-3],),
```

hybridLFPy - Application with E-I network

example_brunel.py parameter setup:

```
#set up synapse parameters as derived from the network
PS.update(dict(
    synParams = dict(
        EX = dict(
            section = ['apic', 'dend'],
            tau = BN.tauSyn,
            syntype = 'AlphaISyn'
        ),
        IN = dict(section = ['dend', 'soma'],
                  ...),),
    #fixed delays of network
    synDelayLoc = dict(
        EX = [BN.delay, BN.delay],
        IN = [BN.delay, BN.delay],
    ),
    #no distribution of delays
    synDelayScale = dict(
        EX = [None, None],
        IN = [None, None],),
```

hybridLFPy - Application with E-I network

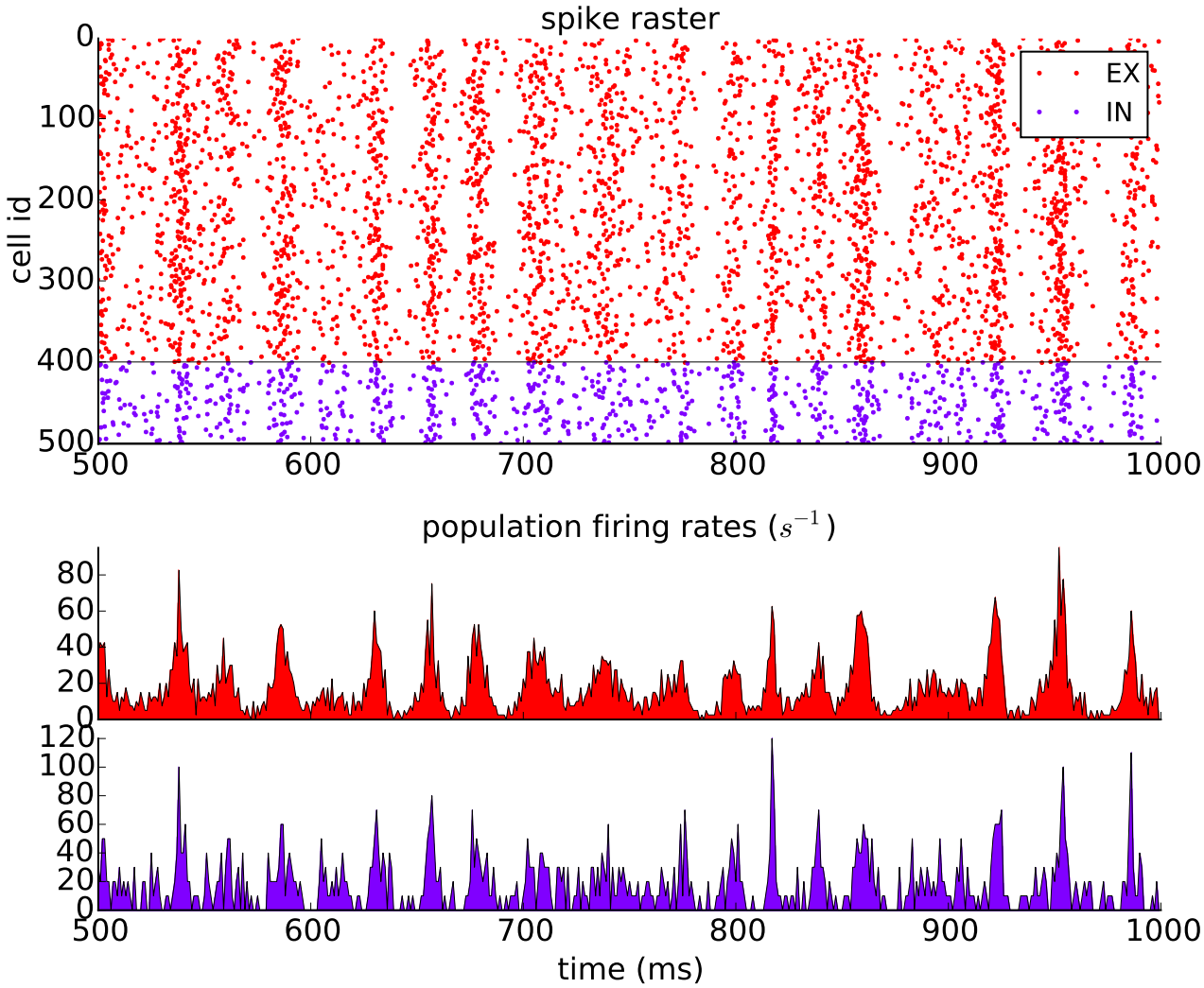
example_brunel.py network spikes:

```
#execute network simulation
BN.simulate()

#wait for the network simulation to finish, resync MPI threads
COMM.Barrier()

#Create an object representation containing the spiking activity of
#the network simulation output that uses sqlite3. Again, kwargs are
#derived from the brunel network instance.
networkSim = CachedNetwork(
    simtime = BN.simtime,
    dt = BN.dt,
    spike_output_path = BN.spike_output_path,
    label = BN.label,
    ext = 'gdf',
    GIDs = {'EX' : [1, BN.NE],
            'IN' : [BN.NE+1, BN.NI]},
)
```

hybridLFPy - Application with E-I network

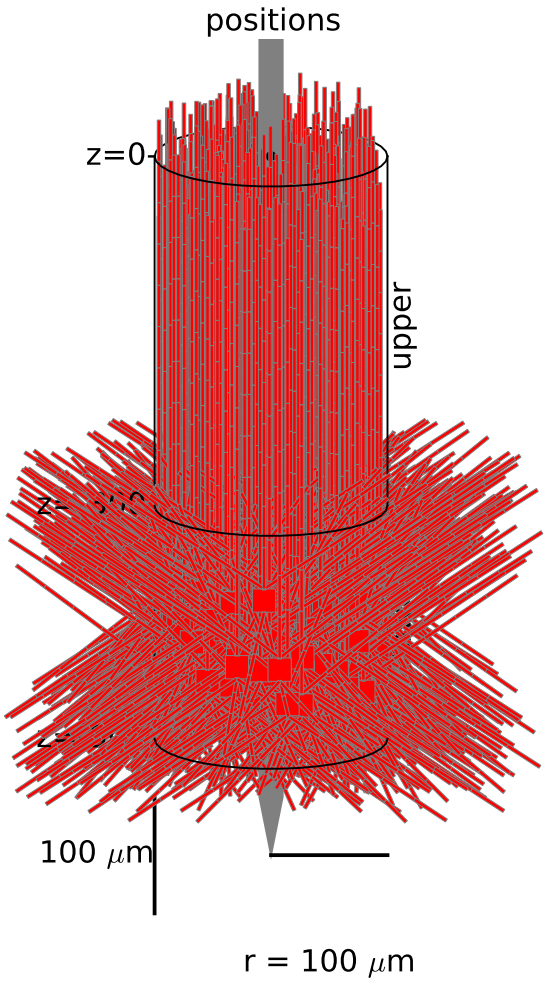


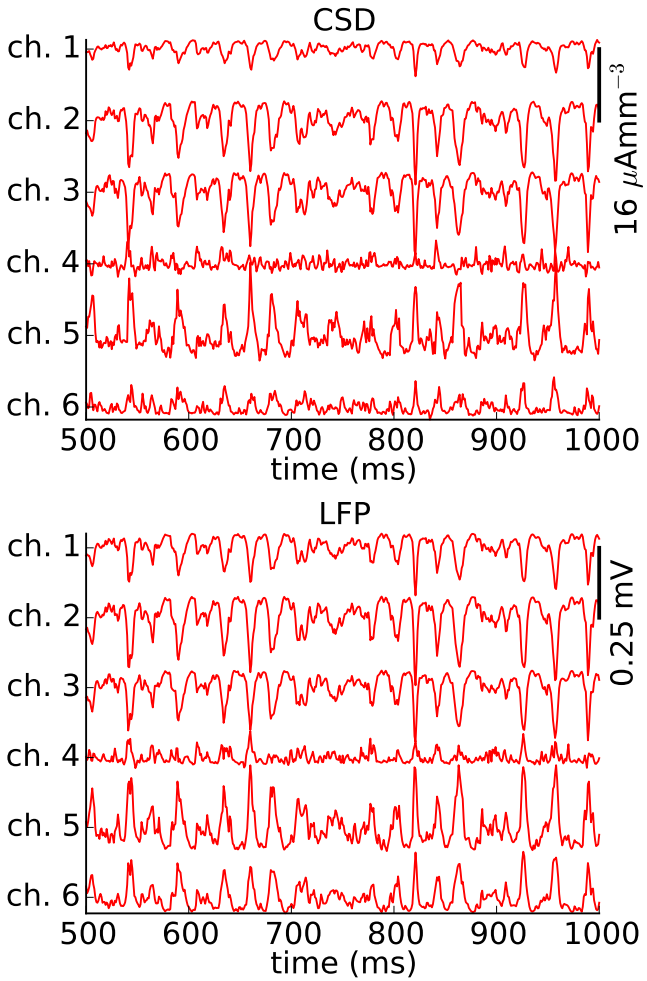
hybridLFPy - Application with E-I network

example_brunel.py runnig single-cell simulations:

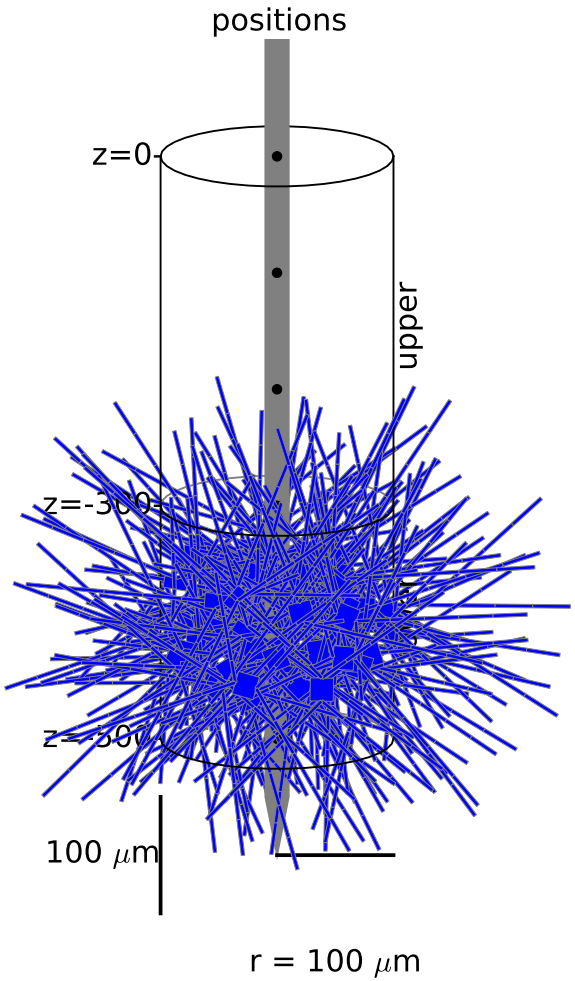
```
for i, Y in enumerate(PS.X):
    #create population:
    pop = Population(
            cellParams = PS.cellParams[Y],
            rand_rot_axis = PS.rand_rot_axis[Y],
            simulationParams = PS.simulationParams,
            populationParams = PS.populationParams[Y],
            layerBoundaries = PS.layerBoundaries,
            electrodeParams = PS.electrodeParams,
            ...
            networkSim = networkSim,
            k_yXL = PS.k_yXL[Y],
            synParams = PS.synParams[Y],
            synDelayLoc = PS.synDelayLoc[Y],
            synDelayScale = PS.synDelayScale[Y],
            J_yX = PS.J_yX[Y],)
    #run simulation and process single-cell data
    pop.run()
    pop.collect_data()
```

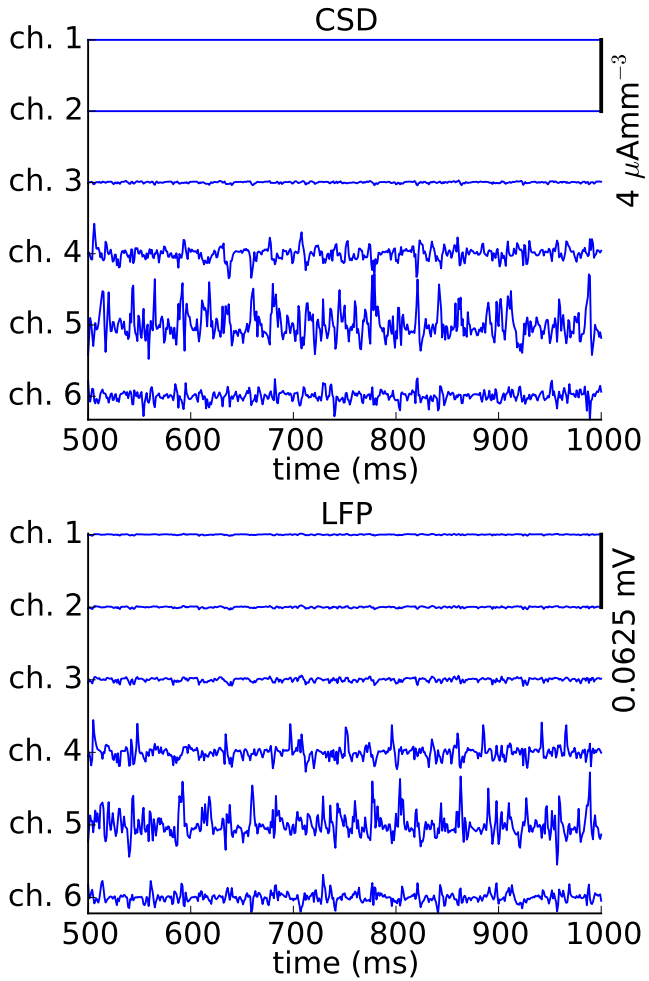
hybridLFPy - Application with E-I network





hybridLFPy - Application with E-I network



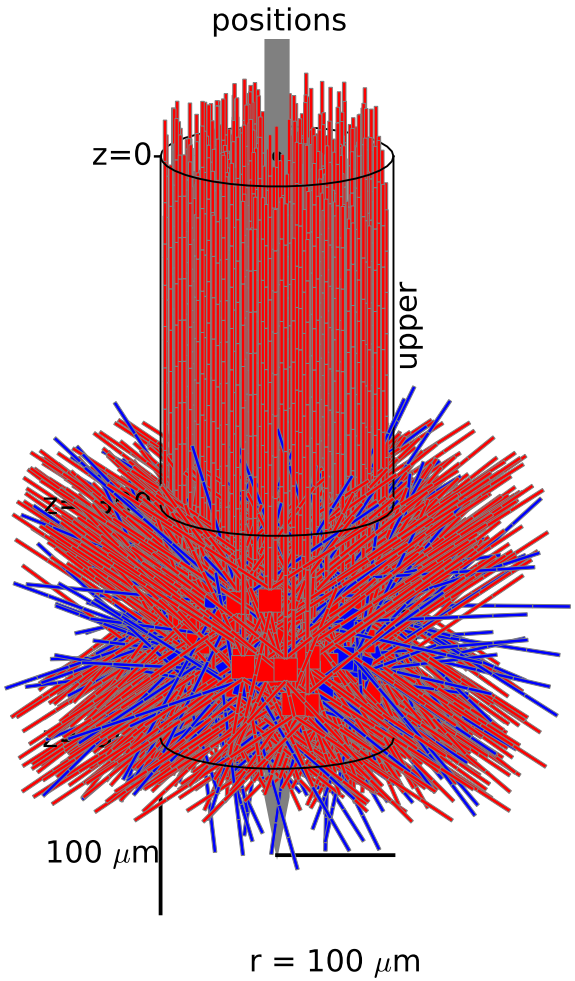


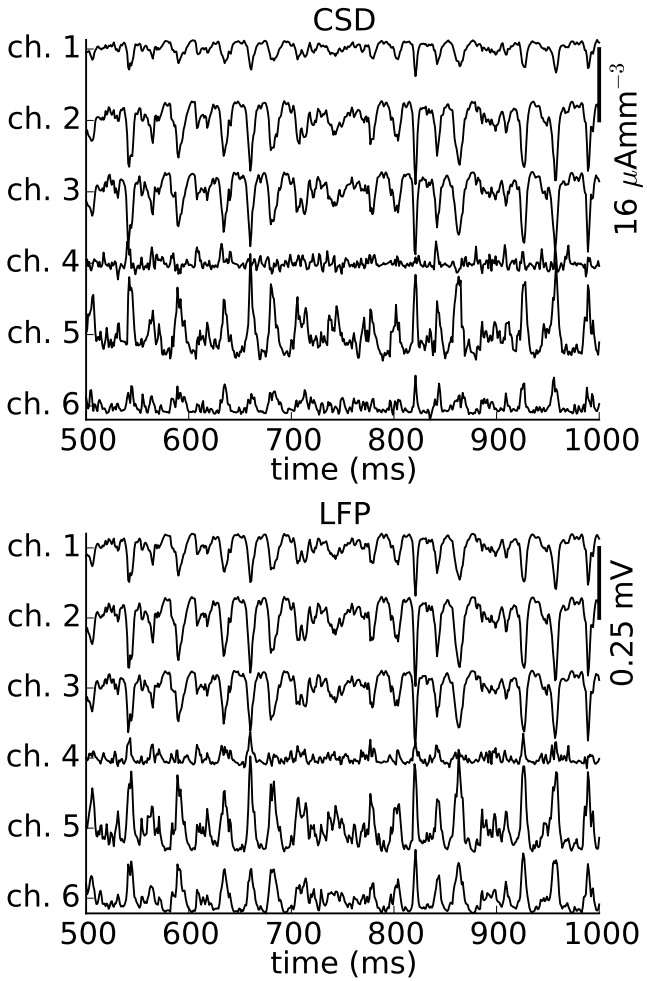
hybridLFPy - Application with E-I network

example_brunel.py creating compound signals:

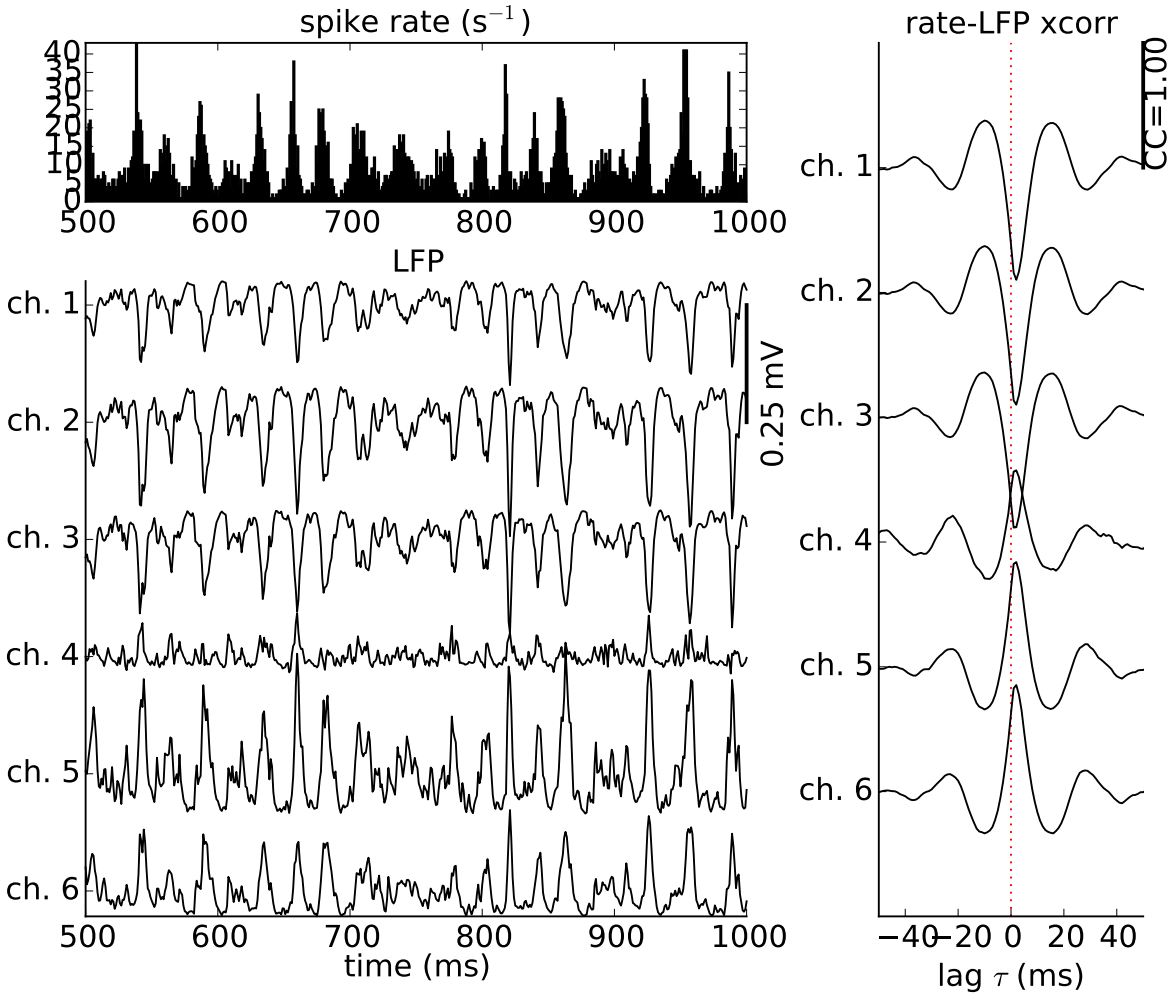
```
#Postprocessing of population output,
#(superposition of population LFPs, CSDs)
postproc = PostProcess(y = PS.X,
                       dt_output = PS.dt_output,
                       savefolder = PS.savefolder,
                       mapping_Yy = PS.mapping_Yy,
                       )
#run procedure
postproc.run()
```

hybridLFPy - Application with E-I network



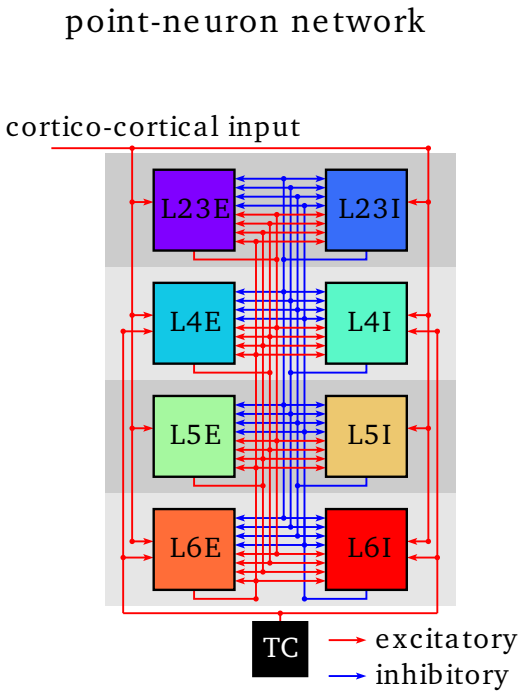


hybridLFPy - Application with E-I network



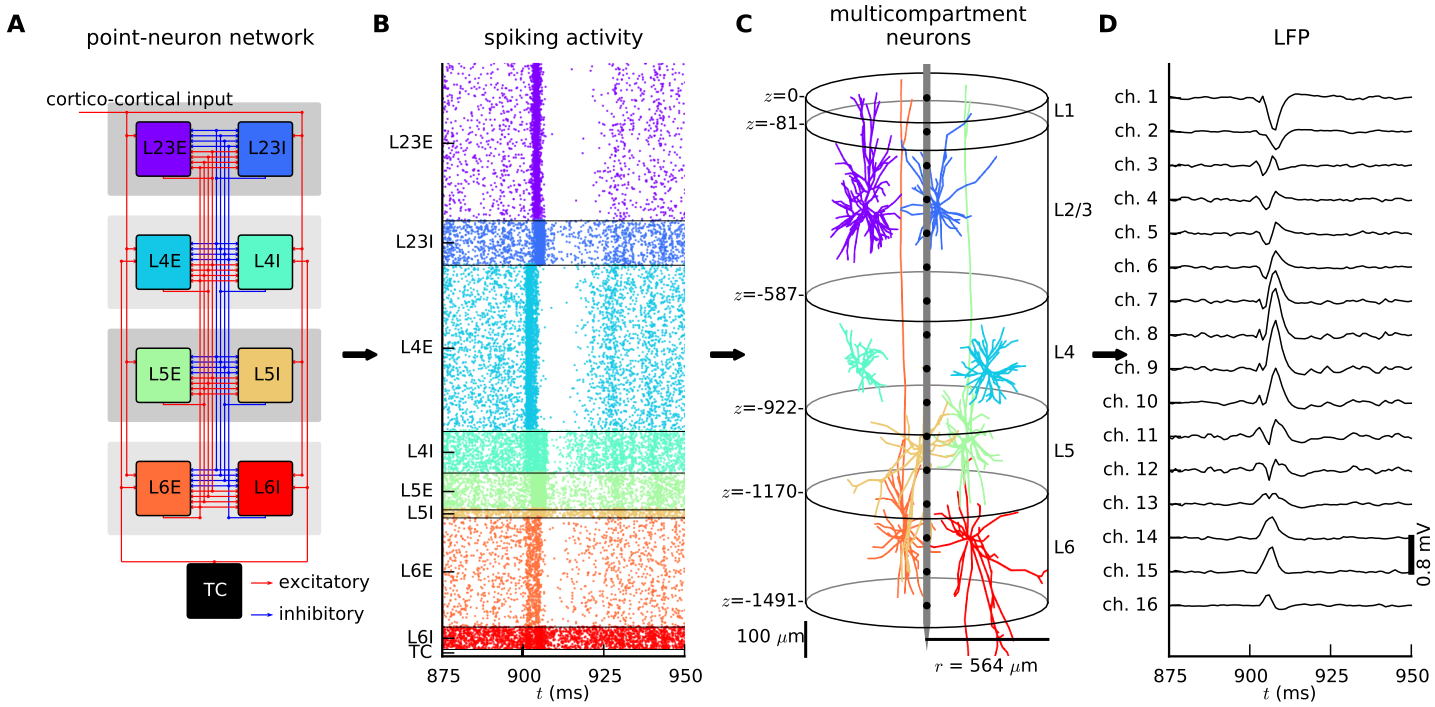
`hybridLFPy` - Application with microcircuit model

- microcircuit model:
  - local circuitry under 1mm^2 (cat VC)
  - 80k LIF point neurons
  - 300M current-based synapses
  - 4 layers
  - 2 populations (E,I) per layer
  - equal dynamics of E-I neurons
  - layer- and type-specific random connectivity



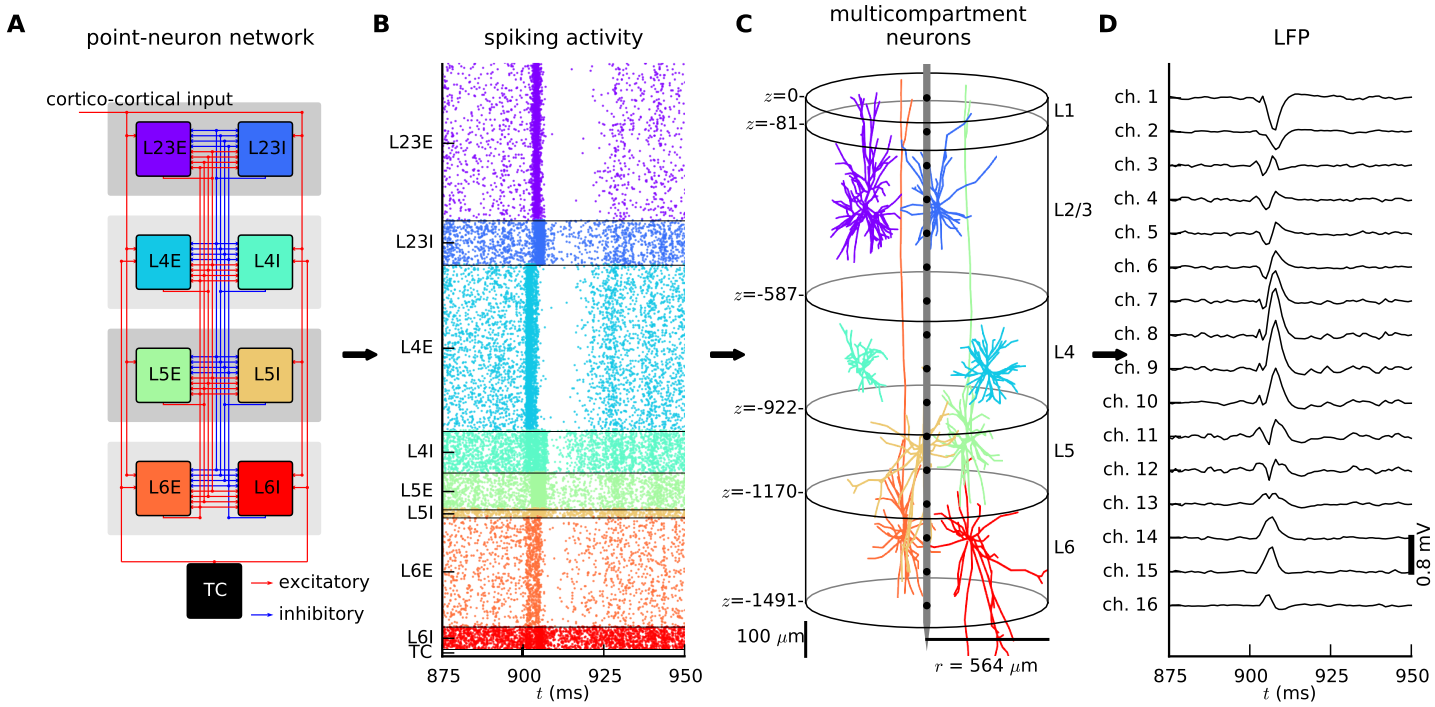
Potjans&Diesmann,
Cereb Cortex (2014)

`hybridLFPy` - Application with microcircuit model



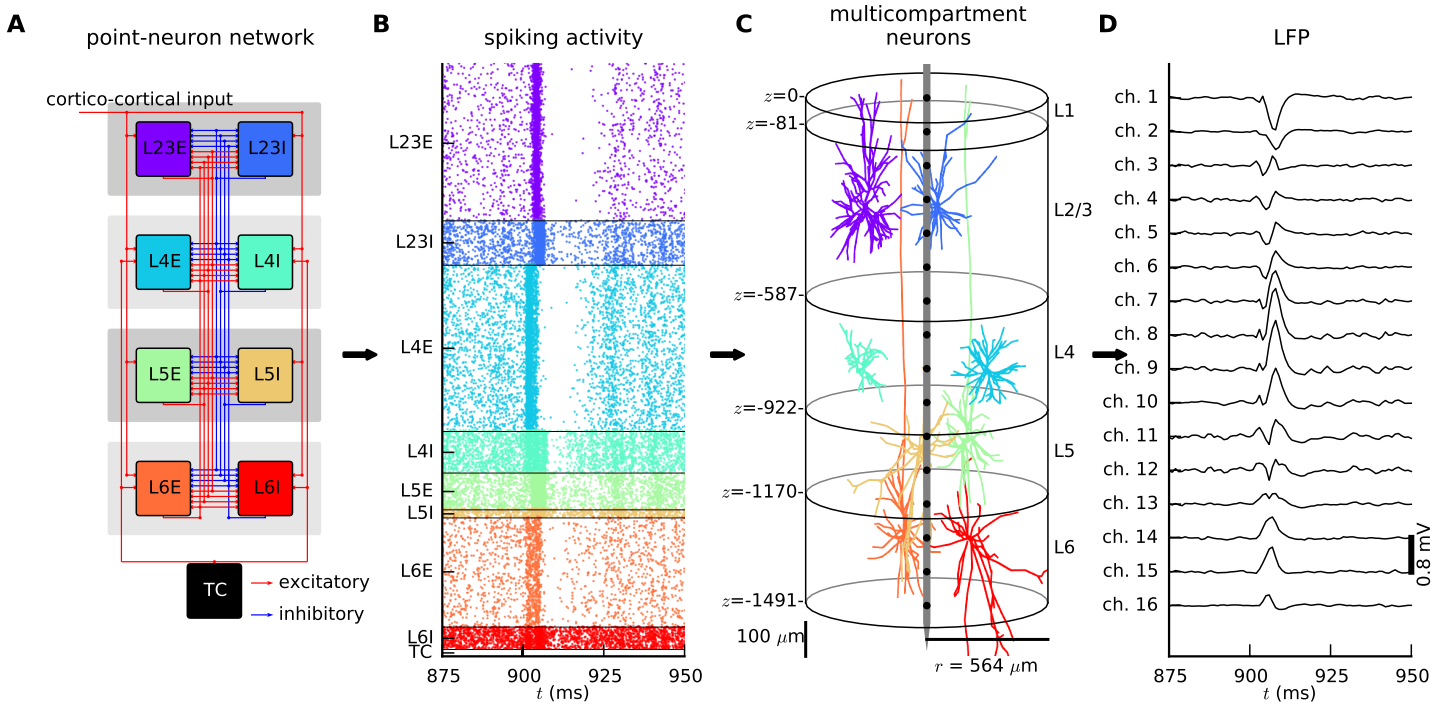
- point-neuron network, Potjans&Diesmann, *Cereb Cortex* (2014) (http://www.opensourcebrain.org)

`hybridLFPy` - Application with microcircuit model



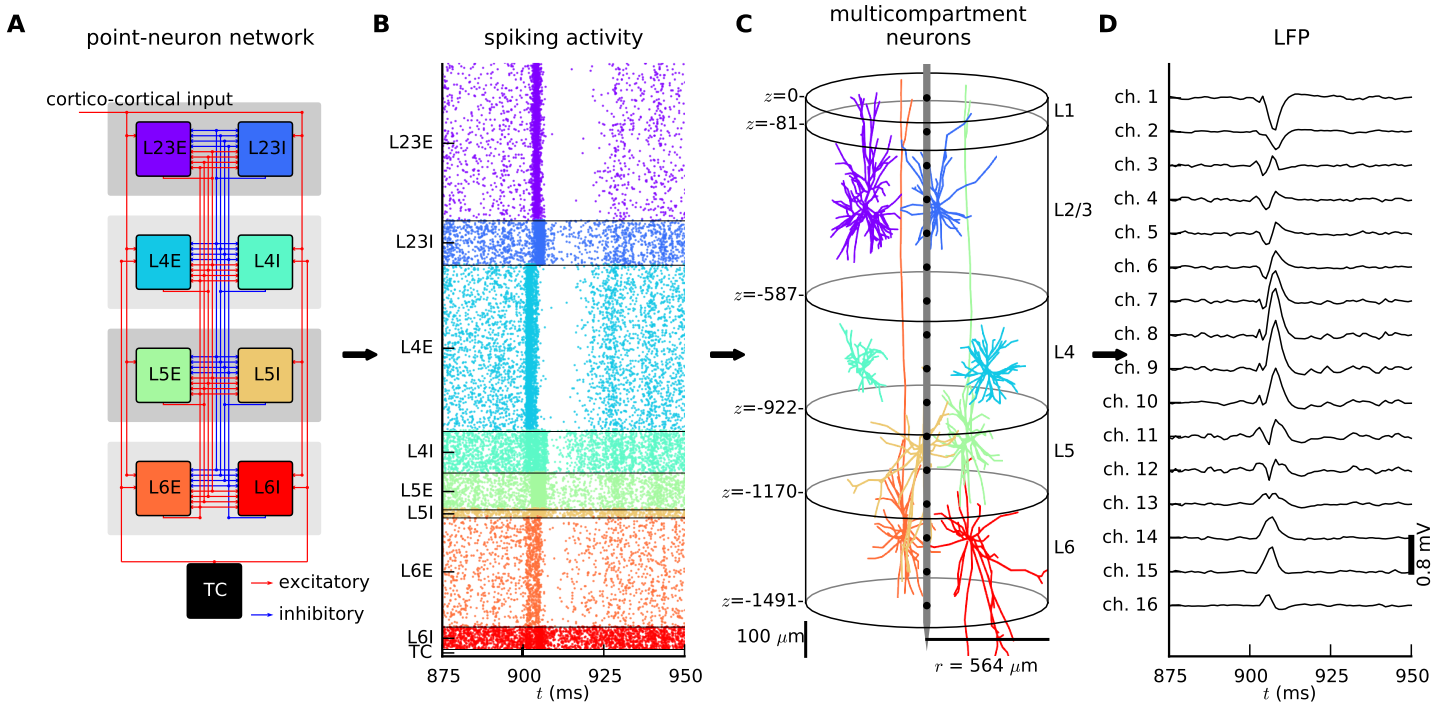
- point-neuron network, Potjans&Diesmann, *Cereb Cortex* (2014) (http://www.opensourcebrain.org)
  - network spikes $\rightarrow$ synaptic activation times

`hybridLFPy` - Application with microcircuit model



- point-neuron network, Potjans&Diesmann, *Cereb Cortex* (2014) (http://www.opensourcebrain.org)
  - network spikes → synaptic activation times
  - cell-type and layer specific connectivity

`hybridLFPy` - Application with microcircuit model



- point-neuron network, Potjans&Diesmann, *Cereb Cortex* (2014) (http://www.opensourcebrain.org)
  - network spikes → synaptic activation times
  - cell-type and layer specific connectivity
  - multi-compartment neurons: "antennas" for LFP generation

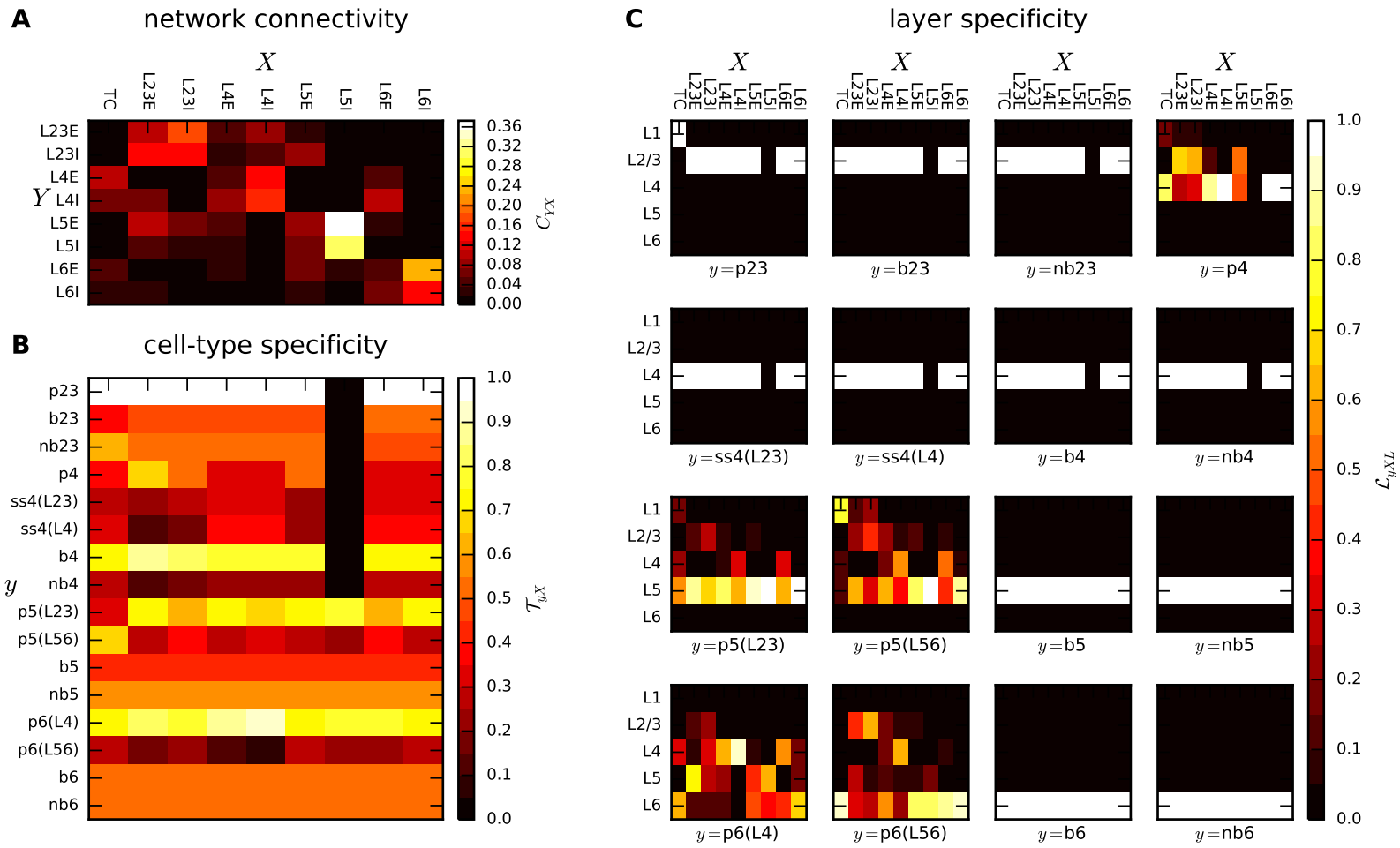
`hybridLFPy` - Application with microcircuit model

presynaptic neurons

postsynaptic neurons		percent of cells	number of synapses	nb1	p2/3	b2/3	nb2/3	ss4(L4)	ss4(L2/3)	p4	b4	nb4	p5(L2/3)	p5(L5/6)	b5	nb5	p6(L4)	p6(L5/6)	b6	nb6	corticocortical	TCs	TCn	TIs	TIn	TRN
nb1		1.5	8890	10.1	6.3	0.6	1.1	.	.	0.1	.	.	0.1	.	.	.	.	.	.	.	77.6	.	4.1	.	.	.
p2/3	L2/3	26	5800	.	59.9	9.1	4.4	0.6	6.9	7.7	.	0.8	7.4	.	.	.	2.3	.	.	0.8	.	.	.	.	.	.
	L1		1306	10.2	6.3	0.1	1.1	.	.	0.1	.	.	0.1	.	.	.	.	.	.	.	78	.	4.1	.	.	.
b2/3		3.1	3854	1.3	51.6	10.6	3.4	0.5	5.8	6.6	.	0.8	6.3	.	.	.	2.1	.	.	0.7	9.8	.	0.5	.	.	.
nb2/3		4.2	3307	1.7	48.6	11.4	3.3	0.5	5.5	6.2	.	0.8	5.9	.	.	.	1.8	.	.	0.6	13	.	0.7	.	.	.
ss4(L4)		9.2	5792	.	2.7	0.2	0.6	11.9	3.7	4.1	7.1	2	0.8	0.1	.	.	32.7	.	.	5.8	25.3	1.7	1.3	.	.	.
ss4(L2/3)		9.2	4989	.	5.6	0.4	0.8	11.3	3.8	4.3	7.2	2.1	1.1	0.1	.	.	31.1	.	.	5.5	23.9	1.7	1.3	.	.	.
p4	L4	9.2	5031	.	4.3	0.2	0.6	11.5	3.6	4.2	7.2	2.1	1.2	0.1	.	.	31.4	0.1	.	5.9	24.5	1.7	1.3	.	.	.
	L2/3		866	.	63.1	5.1	4.1	0.6	7.2	8.1	.	0.6	7.8	.	.	.	2.5	.	.	0.8	.	.	.	.	.	.
	L1		806	10.2	6.3	0.1	1.1	.	.	0.1	.	.	0.1	.	.	.	.	.	.	.	78	.	4.1	.	.	.
b4		5.4	3230	.	5.8	0.5	0.8	11	3.8	4.2	8.4	2.4	1.1	.	.	.	30.3	.	.	5.4	23.3	1.6	1.2	.	.	.
nb4		1.5	3688	.	2.7	0.2	0.6	11.7	3.6	4	8.2	2.3	0.8	0.1	.	.	32.2	.	.	5.7	24.9	1.7	1.3	.	.	.
p5(L2/3)	L5	4.8	4316	.	45.9	1.8	0.3	3.3	2	7.5	.	0.9	11.7	1	0.8	1.1	2.3	2.1	.	11.5	7.2	0.1	0.4	.	.	.
	L4		283	.	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	.	.	33.7	.	.	5.9	26	1.8	1.4	.	.	.
	L2/3		412	.	63.1	5.1	4.1	0.6	7.2	8.1	.	0.6	7.8	.	.	.	2.5	.	.	0.8	.	.	.	.	.	.
	L1		185	10.2	6.3	0.1	1.1	.	.	0.1	.	.	0.1	.	.	.	.	.	.	.	78	.	4.1	.	.	.
p5(L5/6)	L5	1.3	5101	.	44.3	1.7	0.2	3.2	2	7.3	.	0.8	11.3	1.2	0.8	1.1	2.3	2.5	0.3	11.3	9.2	0.2	0.5	.	.	.
	L4		949	.	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	.	.	33.7	.	.	5.9	26	1.8	1.4	.	.	.
	L2/3		1367	.	63.1	5.1	4.1	0.6	7.2	8.1	.	0.6	7.8	.	.	.	2.5	.	.	0.8	.	.	.	.	.	.
	L1		5658	10.2	6.3	0.1	1.1	.	.	0.1	.	.	0.1	.	.	.	.	.	.	.	78	.	4.1	.	.	.
b5		0.6	2981	.	45.5	2.3	0.2	3.3	2	7.5	.	1.1	11.6	1	0.9	1.3	2.3	2	.	11.4	7.2	0.1	0.4	.	.	.
nb5		0.8	2981	.	45.5	2.3	0.2	3.3	2	7.5	.	1.1	11.6	1	0.9	1.3	2.3	2	.	11.4	7.2	0.1	0.4	.	.	.
p6(L4)	L6	13.6	3261	.	2.5	0.1	0.1	0.7	0.9	1.3	.	0.1	0.1	4.9	.	0.3	1.2	13.2	7.7	7.7	55.7	0.6	2.9	.	.	.
	L5		1066	.	46.8	0.8	0.3	3.4	2.1	7.7	.	0.6	11.9	1	0.6	0.8	2.3	2.1	.	11.7	7.4	0.1	0.4	.	.	.
	L4		1915	.	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	.	.	33.7	.	.	5.9	26	1.8	1.4	.	.	.
	L2/3		121	.	63.1	5.1	4.1	0.6	7.2	8.1	.	0.6	7.8	.	.	.	2.5	.	.	0.8	.	.	.	.	.	.
p6(L5/6)	L6	4.5	5573	.	2.5	0.1	0.1	0.7	0.9	1.3	.	0.1	0.1	4.9	.	0.3	1.2	13.2	7.8	7.8	55.7	0.6	2.9	.	.	.
	L5		257	.	46.8	0.8	0.3	3.4	2.1	7.7	.	0.6	11.9	1	0.6	0.8	2.3	2.1	.	11.7	7.4	0.1	0.4	.	.	.
	L4		243	.	2.8	0.1	0.7	12.2	3.8	4.2	5.2	1.5	0.8	0.1	.	.	33.7	.	.	5.9	26	1.8	1.4	.	.	.
	L2/3		286	.	63.1	5.1	4.1	0.6	7.2	8.1	.	0.6	7.8	.	.	.	2.5	.	.	0.8	.	.	.	.	.	.
	L1		62	10.2	6.3	0.1	1.1	.	.	0.1	.	.	0.1	.	.	.	.	.	.	.	78	.	4.1	.	.	.
b6		2	3220	.	2.5	0.1	0.1	0.7	0.9	1.3	.	0.1	0.1	4.9	.	0.4	1.2	13.2	7.7	7.7	55.7	0.6	2.9	.	.	.
nb6		2	3220	.	2.5	0.1	0.1	0.7	0.9	1.3	.	0.1	0.1	4.9	.	0.4	1.2	13.2	7.7	7.7	55.7	0.6	2.9	.	.	.
				brainstem		sensory																				
TCs		0.5	4000	31	.	7.1	.	.	.	.	.	.	.	.	.	.	23	8	.	.	.	.	.	5	.	25.9
TCn		0.5	4000	31	.	7.1	.	.	.	.	.	.	14	3.8	.	.	.	13.2	.	.	.	.	.	.	5	25.9
TIs		0.1	3000	13.5	.	48.7	.	.	.	.	.	.	.	.	.	.	9.8	3.3	.	.	.	0.4	.	24.4	.	.
TIn		0.1	3000	13.4	.	48.7	.	.	.	.	.	.	5.8	1.6	.	.	.	5.4	.	.	.	.	0.6	.	24.4	.
TRN		0.5	4000	40	.	.	.	.	.	.	.	.	.	.	.	.	30	.	.	.	.	10	10	.	.	10

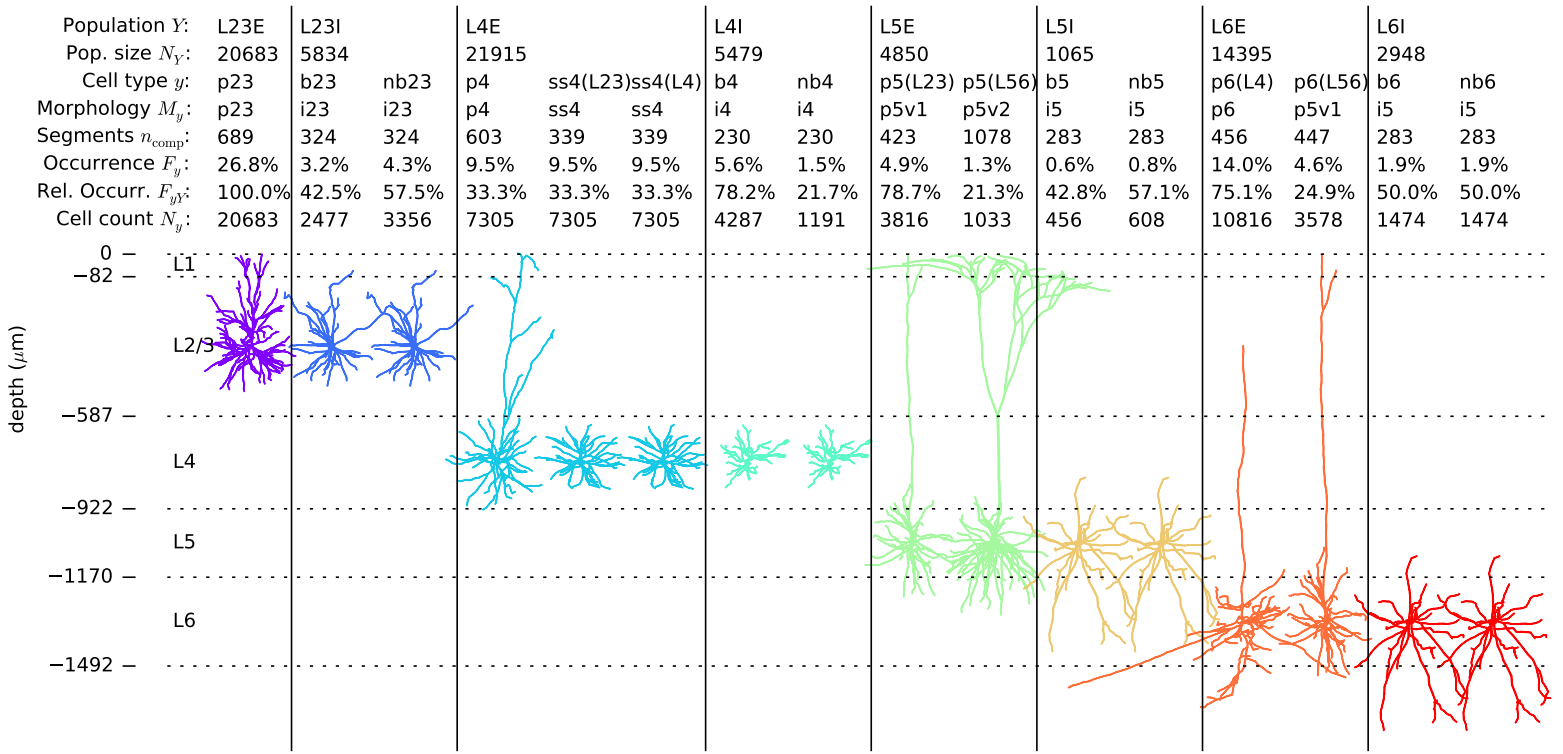
Cat VC connectivity of Binzegger et al. *J Neurosci* (2004)

hybridLFPy - Application with microcircuit model



Network connectivity and layer specificity of connections

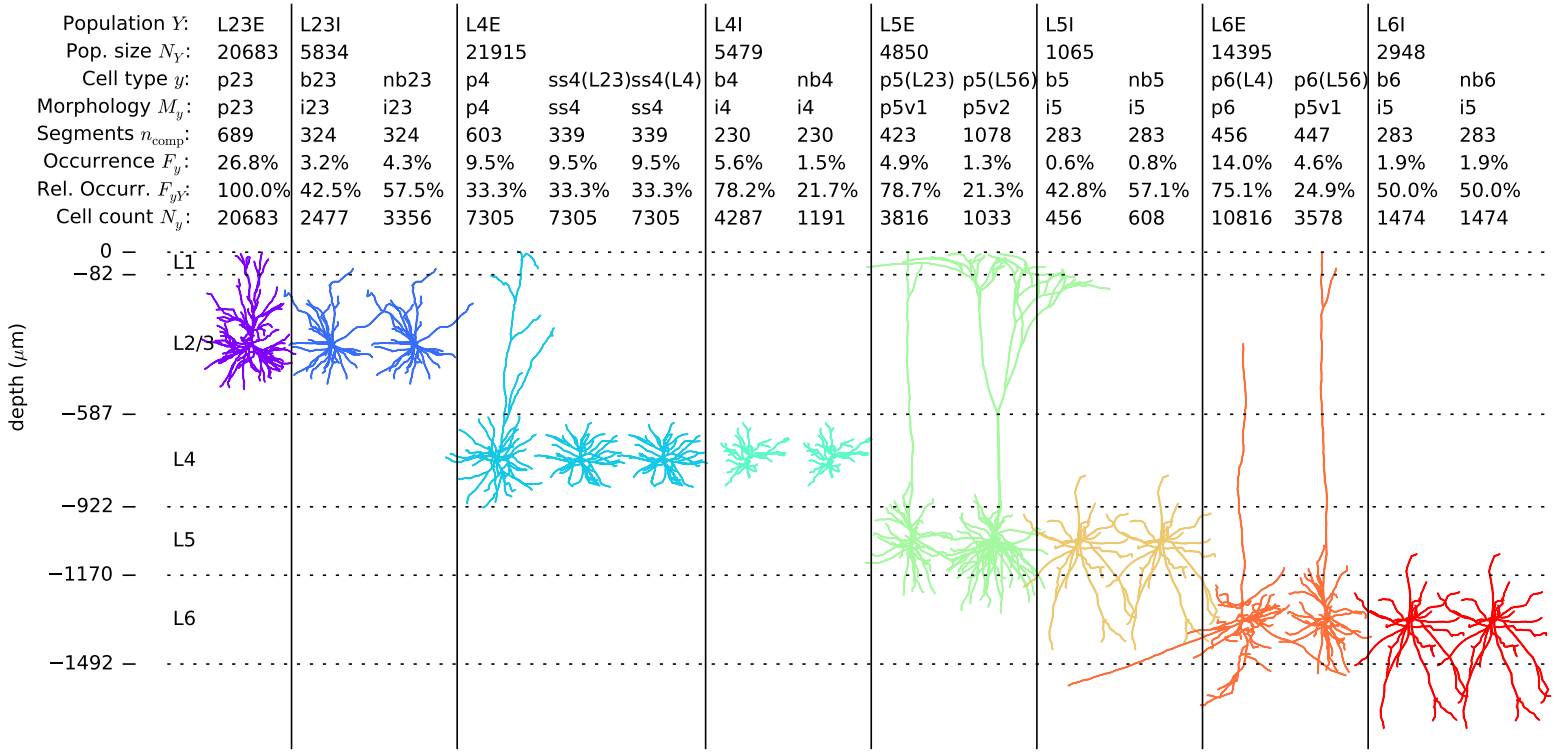
hybridLFPy - Application with microcircuit model



Morphologies and layer boundaries: [Stepanyants et al. *Cereb Cortex* (2008)]

- reconstructions from cat (visual/somatosensory cortices)
- limited data availability: reuse files across cell types

hybridLFPy - Application with microcircuit model



Extrapolation from anatomical connectivity data:

- cell-type specific connectivity: 16 cell types
- layer specificity of connections

[Binzegger et al. (2004)]

hybridLFPy - Application with microcircuit model

Microcircuit model example

- Main example scripts:

```
example_microcircuit.py
example_microcircuit_params.py
binzegger_connectivity_table.json
morphologies/ballnsticks/*.hoc
expsyni.mod
```

hybridLFPy - Application with microcircuit model

Microcircuit model example

- Main example scripts:

```
example_microcircuit.py
example_microcircuit_params.py
binzegger_connectivity_table.json
morphologies/ballnsticks/*.hoc
expsyni.mod
```

- Execute model

```
cd /PATH/TO/hybridLFPy/examples/
nrnivmodl
mpirun -np 128 python example_microcircuit.py
```

hybridLFPy - Application with microcircuit model

Microcircuit model example

- example_microcircuit_params.py
  - Defines and derives parameter values
  - Process anatomical connectivity
  - Map network connectivity onto LFP model
  - Defined through parameter class objects

```
class general_params(object):
    '''class defining general model parameters'''

class point_neuron_network_params(general_params):
    '''class defining point-neuron network parameters'''

class multicompartment_params(point_neuron_network_params):
    '''class defining additional attributes needed by
    hybridLFPy.Population and hybridLFPy.DummyNetwork'''
```

hybridLFPy - Application with microcircuit model

Microcircuit model example

- example_microcircuit.py
  - Load parameter objects

```
networkParams = point_neuron_network_params()
params = multicompartment_params() #all
```

  - Executes network simulation

```
sli_run(parameters=networkParams, fname='microcircuit.sli'
merge_gdf(networkParams, ...)
networkSim = CachedNetwork(**params.networkSimParams)
```

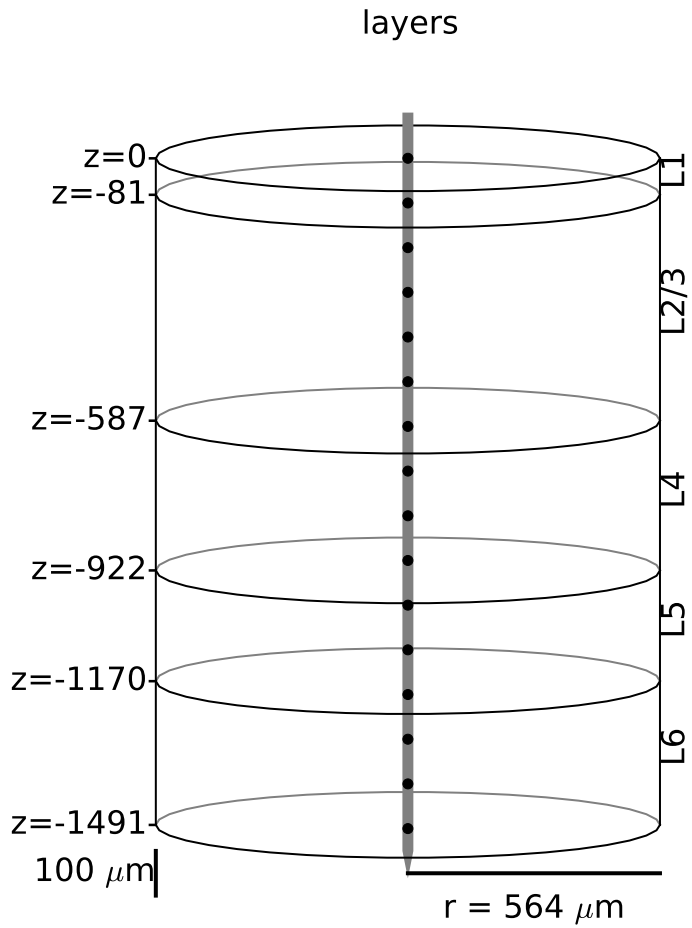
  - Calculate extracellular potentials

```
for i, y in enumerate(params.y):
    pop = Population(cellParams=params.yCellParams[y], ...)
    pop.run()
    pop.collect_data()
```

`hybridLFPy` - Application with microcircuit model

Microcircuit model example

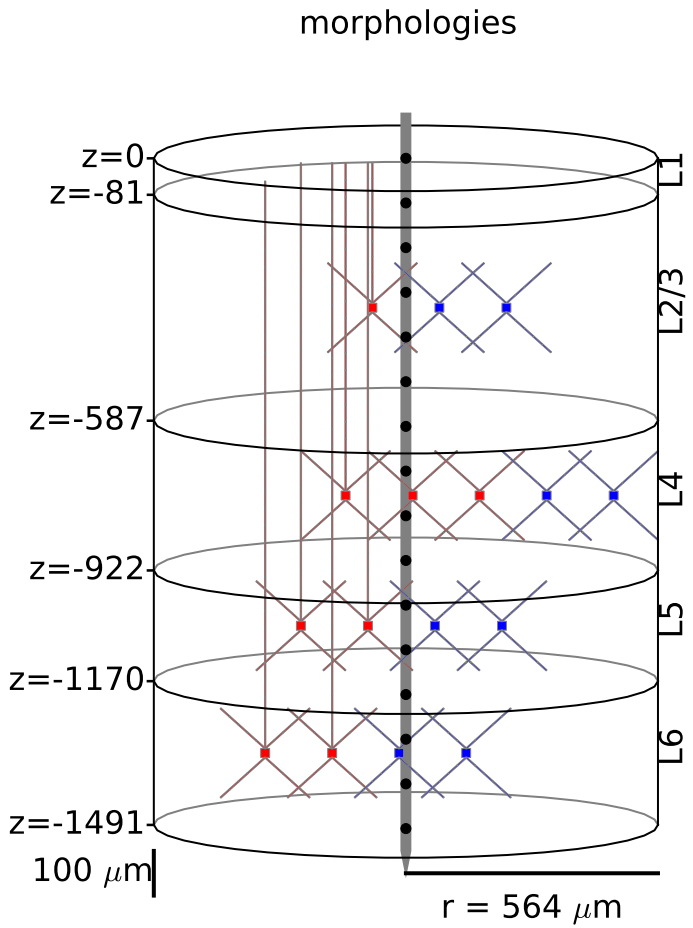
- 1mm^2 cortical surface area
- Layer boundaries from Stepanyants et al. (2008)



`hybridLFPy` - Application with microcircuit model

Microcircuit model example

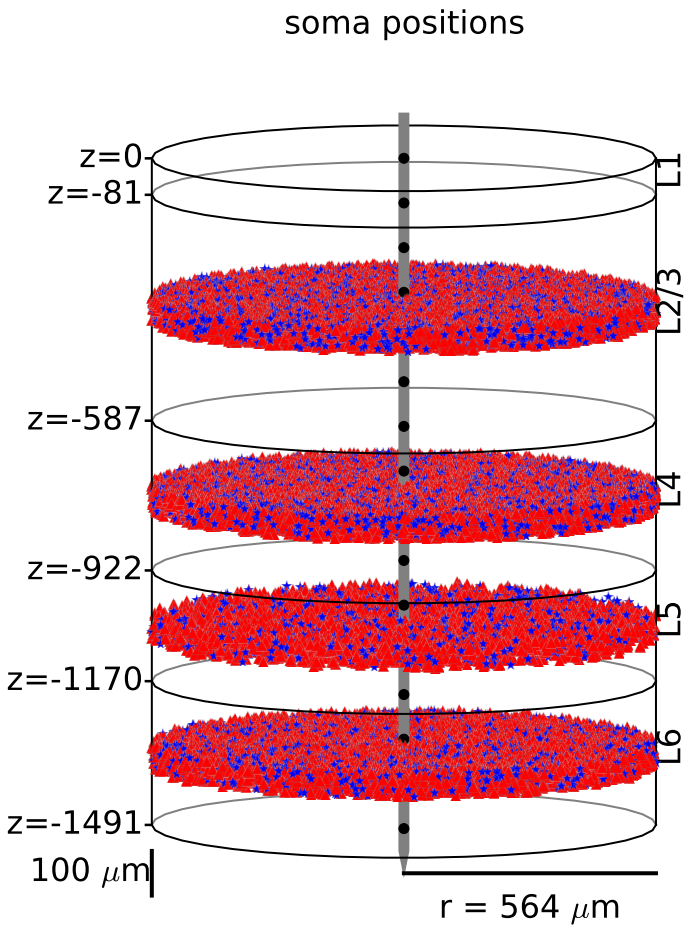
- 1mm^2 cortical surface area
- Layer boundaries from Stepanyants et al. (2008)
- Simplified morphologies for each cell type



`hybridLFPy` - Application with microcircuit model

Microcircuit model example

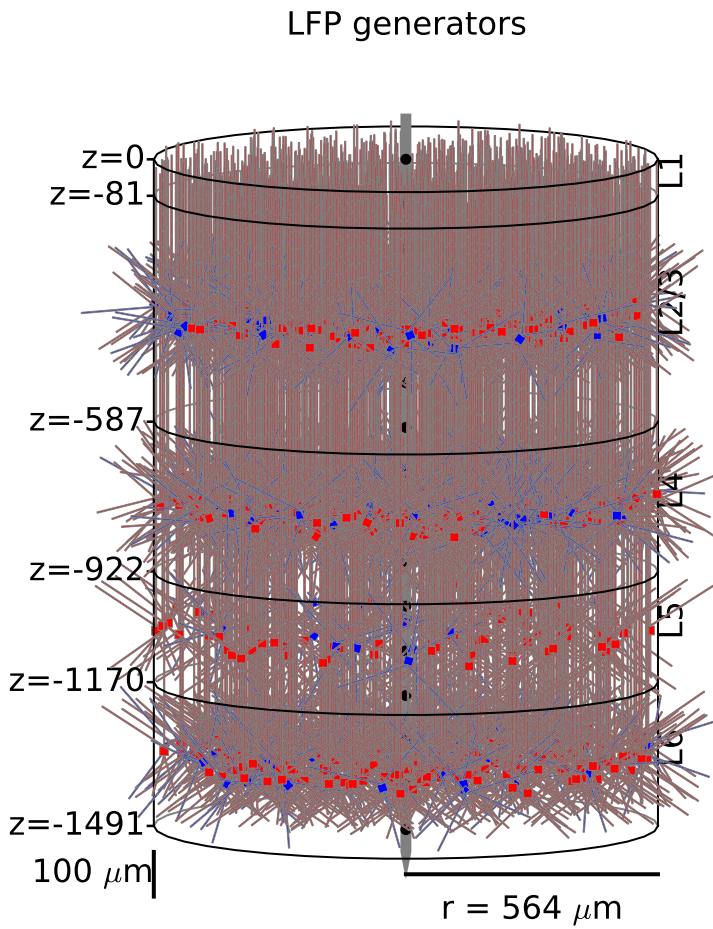
- 1mm^2 cortical surface area
- Layer boundaries from Stepanyants et al. (2008)
- Simplified morphologies for each cell type
- Random soma positions within layers



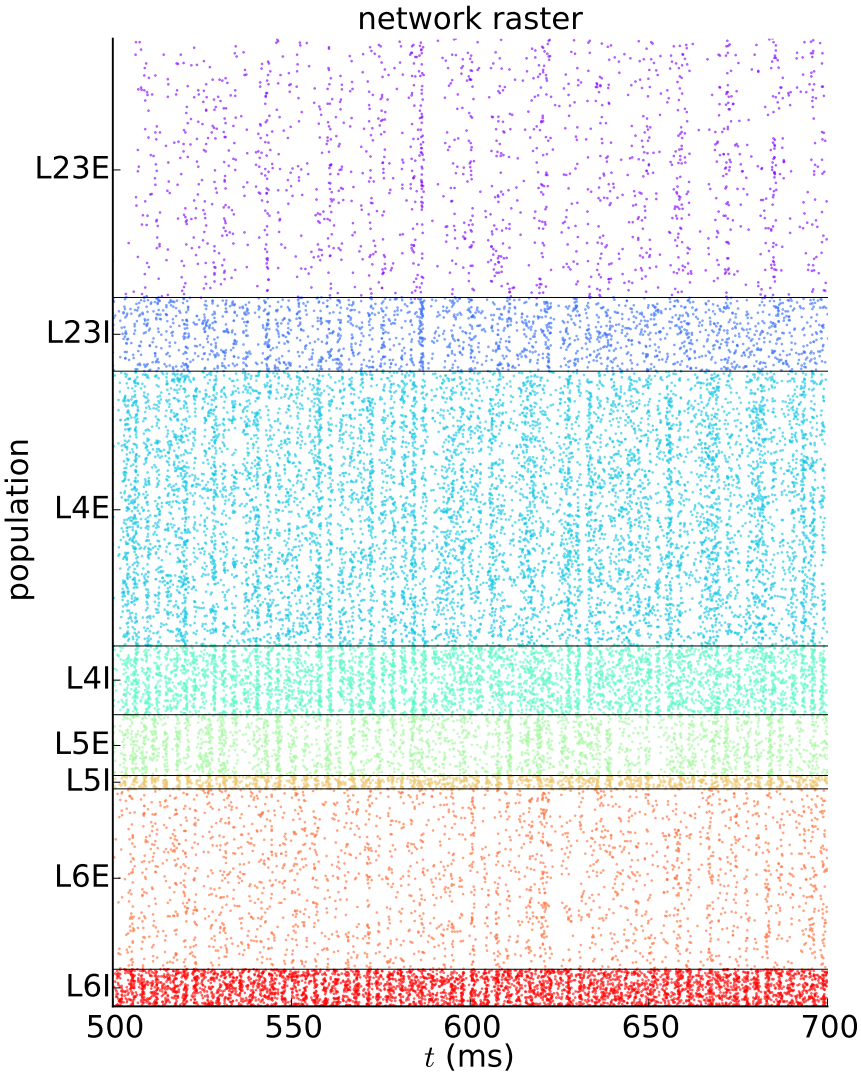
`hybridLFPy` - Application with microcircuit model

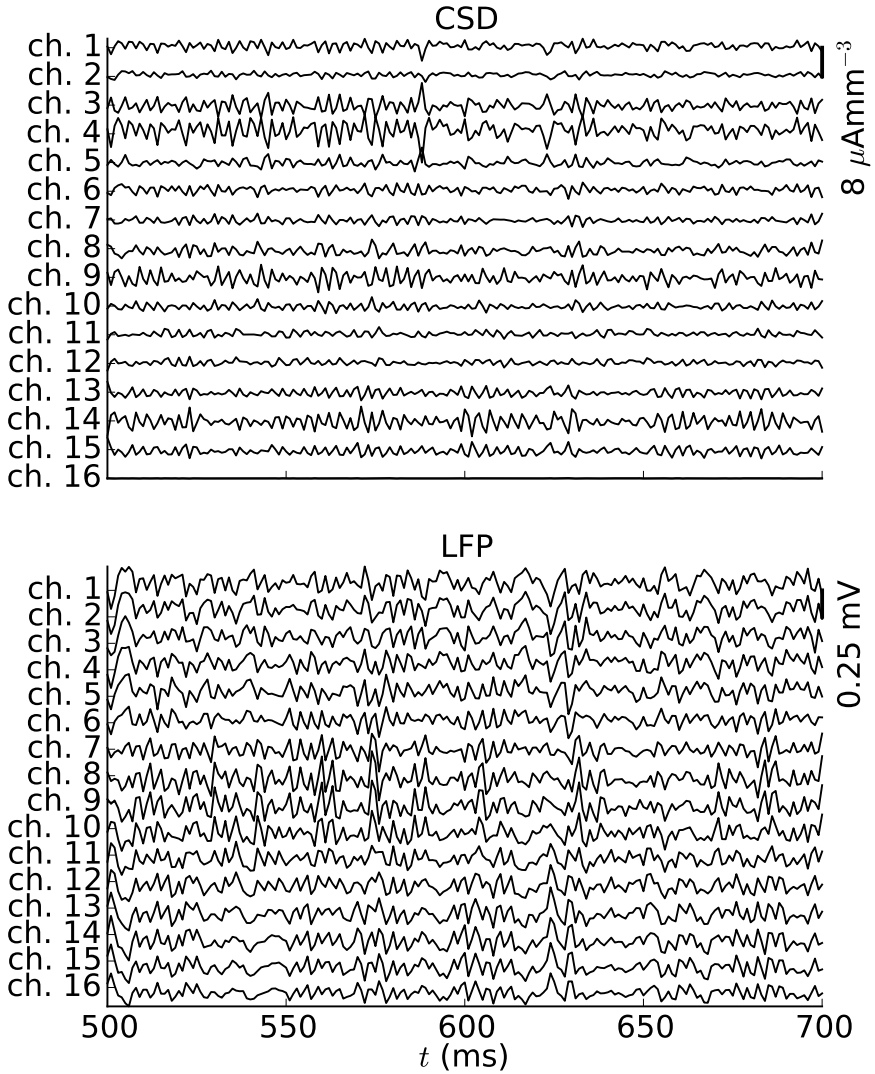
Microcircuit model example

- 1mm^2 cortical surface area
- Layer boundaries from Stepanyants et al. (2008)
- Simplified morphologies for each cell type
- Random soma positions within layers
- 78000 multicompartment neurons generate LFP



`hybridLFPy` - Application with microcircuit model





Summary & Outlook

- Forward modeling scheme:
  - Derived using standard electrostatic theory
  - Multicompartment neuron models
- **`LFPy`**; http://LFPy.github.io:
  - Extracellular potentials of single-neuron models
  - Anisotripic extracellular medium
  - Method of Images (MoI) for inhomogeneous media
  - Support parallel network implementations
- **`hybridLFPy`**; http://inm-6.github.com/hybridLFPy:
  - LFP predictions of point neuron network models
  - Main application: Potjans&Diesmann, *Cereb Cortex* (2014)
  - Multi-area model predictions
  - Network models with distance dependent connections
  - Verify simplified LFP prediction schemes

Talks & Posters

- 010: Zachariou et al. Contrast-dependent Modulation of Gamma Rhythm in V1: a Network Model.
- P60: Telenczuk et al. How neuronal correlations affect the LFP signal?
- **P67:** Hagen et al. Hybrid scheme for modeling local field potentials from point-neuron networks
- P68: Halnes et al. Can ionic diffusion have an effect on extracellular potentials?
- P93: Cui et al. Is it right to estimate inter-modular connectivity from local field potentials?
- P168: Constantinou et al. Information transfer by local field potentials in the hippocampal formation
- P194: Yates et al. Canonical correlations reveal co-variability between spike trains and local field potentials in area MT
- P203: Ness et al. Contributions from active dendritic conductances to the Local Field Potential.

Acknowledgements

- Helmholtz Association: HASB and portfolio theme SMHB
- Jülich Aachen Research Alliance (JARA)
- EU grant 269921 (BrainScaleS)
- EU Grant 604102 (Human Brain Project, HBP)
- Research Council of Norway (NFR) through NevroNor, eNEURO, Notur, NN4661K.

Questions?

Questions?

If not - feel free to try out

- **`iCSD`** and **`kCSD`** tools
  - https://github.com/espenhgn/iCSD
  - https://github.com/INCF/pykCSD
  - (ElePhAnT ElectroPhysiology Analysis Toolkit https://github.com/NeuralEnsemble/elephant, https://github.com/ccluri/elephant)
- **`LFPy`**
  - https://github.com/LFPy/LFPy
  - http://LFPy.github.io
- **`hybridLFPy`**
  - https://github.com/INM-6/hybridLFPy
  - http://inm-6.github.io/hybridLFPy