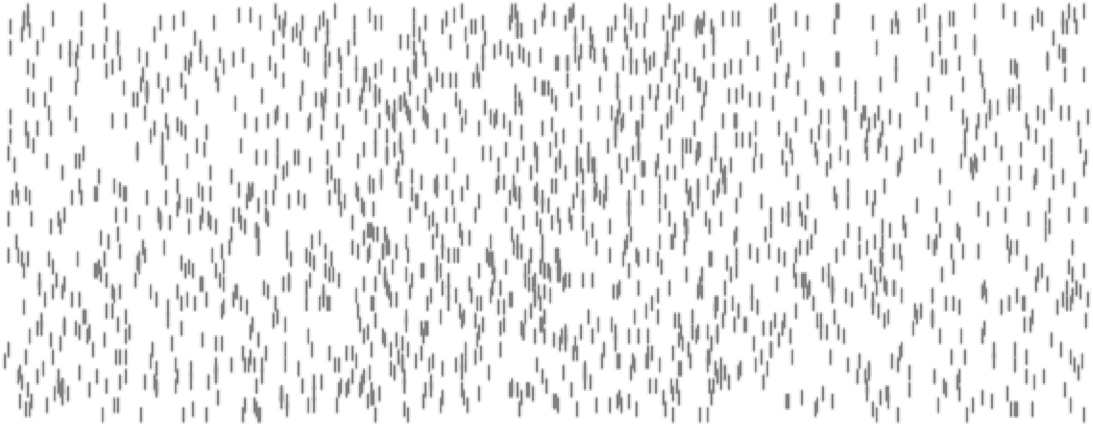


Simulating neural computation and information processing with *Brian*

Marcel Stimberg

Institut de la Vision/Sorbonne Université

marcel.stimberg@inserm.fr

Course material

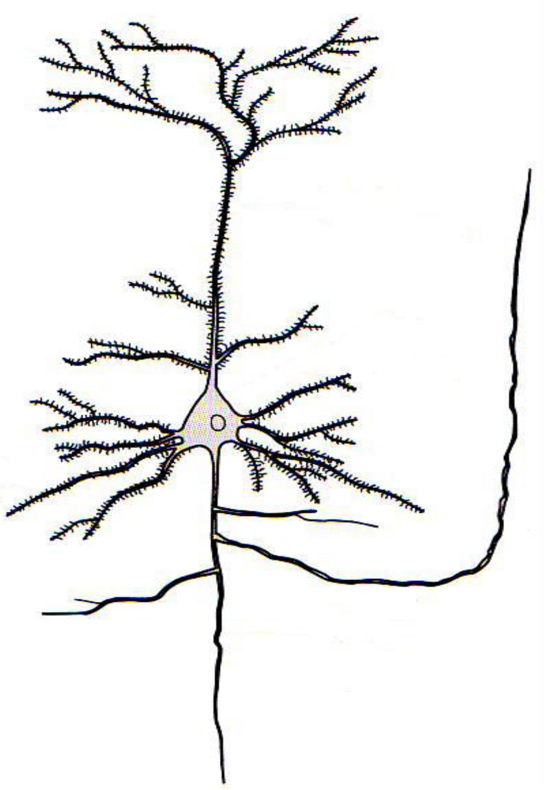
Updated material will be uploaded here:

`github.com/brian-team/brian-material/tree/master/2019-TD-Brian-Sorbonne`

To download everything in a single ZIP file (includes material from other courses as well):
`github.com/brian-team/brian-material/archive/master.zip`

To download individual jupyter notebook files, make sure to switch to "raw" view

Plan for today



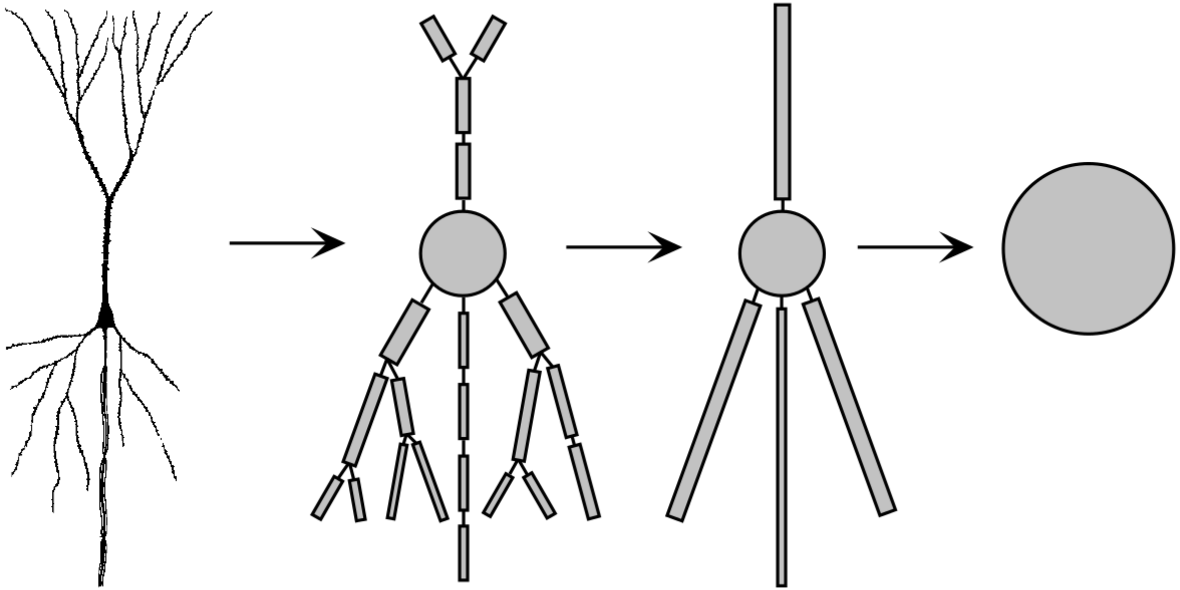
- Introduction to modelling with Brian

Interactive tutorial (“live coding”):

- The jupyter notebook
- **Part 1:** Modelling neurons
- **Part 2:** Modelling synapses

Modelling networks of neurons

Individual elements

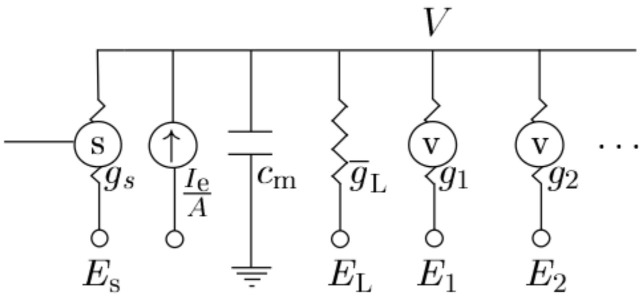


Detailed neuronal morphologies → point-neuron models

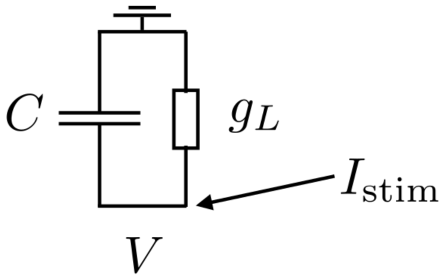
Modelling networks of neurons

Individual elements

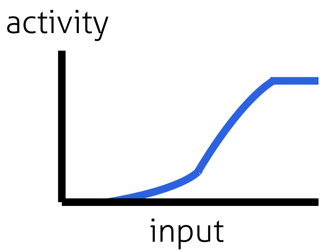
Point-neuron models



Hodgkin-Huxley formalism



integrate-and-fire model

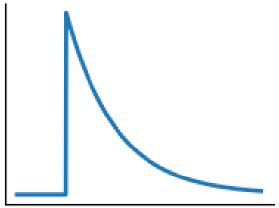


firing rate models

Modelling networks of neurons

Synapses

membrane potential

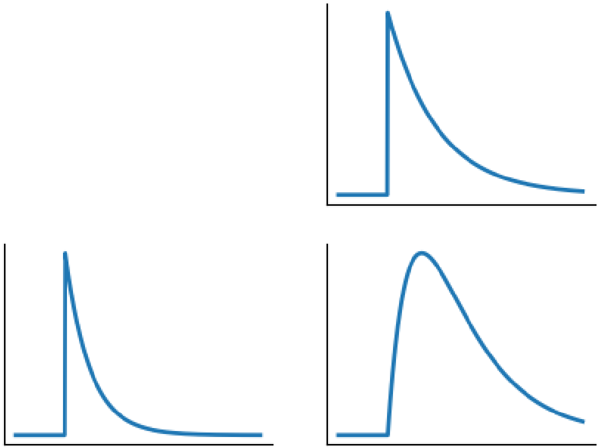


“delta synapse”

Modelling networks of neurons

Synapses

synaptic current membrane potential



"delta synapse"

exponential
current-based

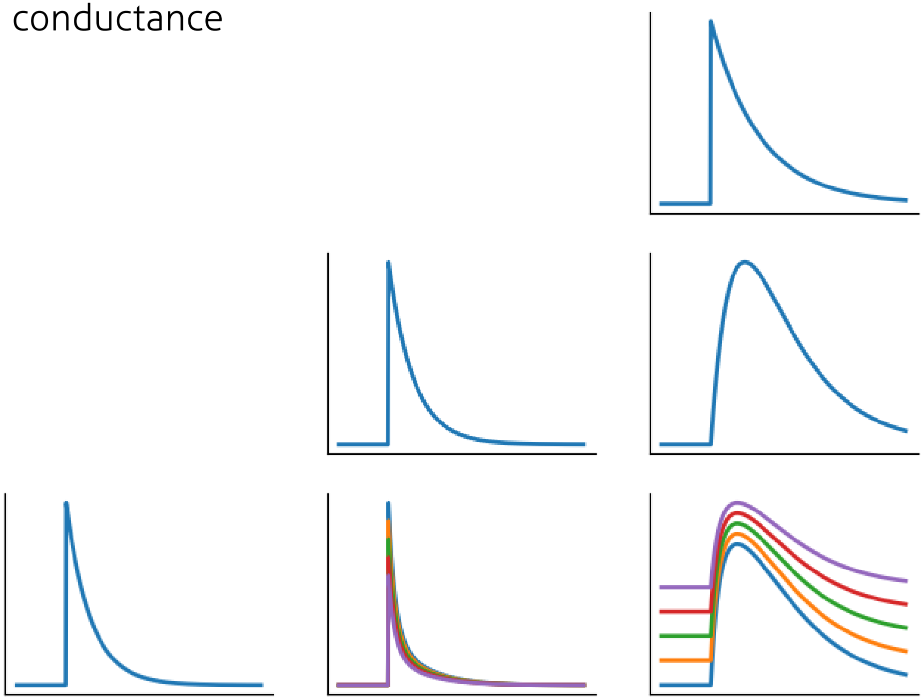
Modelling networks of neurons

Synapses

synaptic conductance

synaptic current

membrane potential



“delta synapse”

exponential current-based

exponential conductance-based

The BRIAN simulator

Who is Brian?

- Simulator for spiking neuronal networks, written in Python
- Started by Dan Goodman and Romain Brette at ENS Paris in 2007
- "A simulator should not only save the time of processors, but also the time of scientists"
- Does not provide a library of fixed models but allows for a flexible definition of (almost) arbitrary models
- Focusses on "medium-sized" neuronal networks
  ("a few" to ~100000 neurons), simulations on standard PCs, not supercomputers
- Tool for research and teaching
- Free-and-open-source

Brian's approach

- *Philosophy*: Mathematical model descriptions
  - Flexible system to define models instead of library of prepared models
  - Explicit about model details
  - Mathematical notation, physical units
- *Technology*: Code generation
  - High-level descriptions transformed into low-level code
  - Modular architecture allows for extensions (e.g. to run code on GPU)

Example: neuron model

$$C\frac{\mathrm{d}V}{\mathrm{d}t} = g_L(V_{\mathrm{rest}} - V) + I_{\mathrm{stim}}$$

$$V(t) > V_{\mathrm{threshold}} \quad \rightarrow \quad \mathrm{spike} + V(t) = V_{\mathrm{reset}}$$

```
N_neurons = 100
C = 200*pF
g_L = 10*nS
V_rest = -70*mV
V_threshold = -50*mV
V_reset = V_rest
I_stim = 1*nA

eqs = 'dV/dt = (g_L*(V_rest - V) + I_stim)/C : volt'
neurons = NeuronGroup(N_neurons, eqs,
                      threshold='V>V_threshold', reset='V=V_reset')
```

Example synapse model

exponential, current-based synapse:

- when a spike arrives, increase I_{syn} by 0.1nA
- between spikes, decay exponentially with τ_{syn}

$$\frac{dI_{syn}}{dt} = \frac{-I_{syn}}{\tau_{syn}}$$

```
eqs = '''dV/dt = (g_L*(V_rest - V) + I_syn)/C : volt
         dI_syn/dt = -I_syn/tau_syn : amp'''
neurons = NeuronGroup(N_neurons, eqs,
                      threshold='V>V_threshold', reset='V=V_reset')
synapses = Synapses(..., neurons, on_pre='I_syn += 0.1*nA')
synapses.connect(...)
```

More info

Documentation: https://brian2.readthedocs.io

Mailing list: briansupport@googlegroups.com

Articles:

Stimberg, Marcel, Romain Brette, and Dan FM Goodman. "Brian 2, an Intuitive and Efficient Neural Simulator." ELife 8 (2019): e47314. https://doi.org/10.7554/eLife.47314.

Stimberg, Marcel, Dan F. M. Goodman, Victor Benichoux, and Romain Brette. "Equation-Oriented Specification of Neural Models for Simulations." Frontiers in Neuroinformatics 8 (2014). https://doi.org/10.3389/fninf.2014.00006