

Simulating neural computation and information processing with



www.briansimulator.org

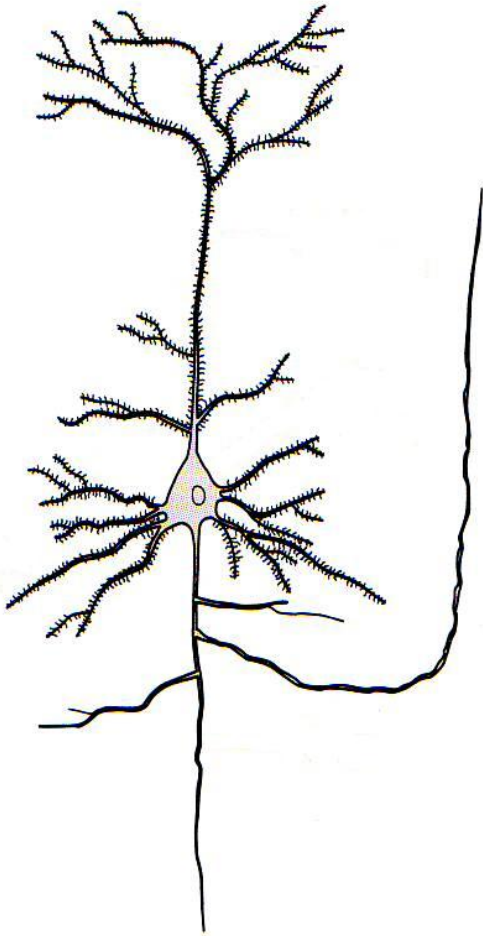
Marcel Stimberg
Computational Neuroscience of Sensory Systems Group – Institut de la Vision
marcel.stimberg@inserm.fr

Plan for today

- Introduction to modelling with Brian

Interactive tutorial (“live coding”):

- The jupyter notebook
- **Part 1:** Modelling neurons
- **Part 2:** Modelling synapses
- **Part 3:**
Case study: learning patterns with STDP



Course material

Updated material will be uploaded here:

github.com/brian-team/brian-material/tree/master/2017-TD-Brian-UPMC-Paris

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

BRIAN²

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



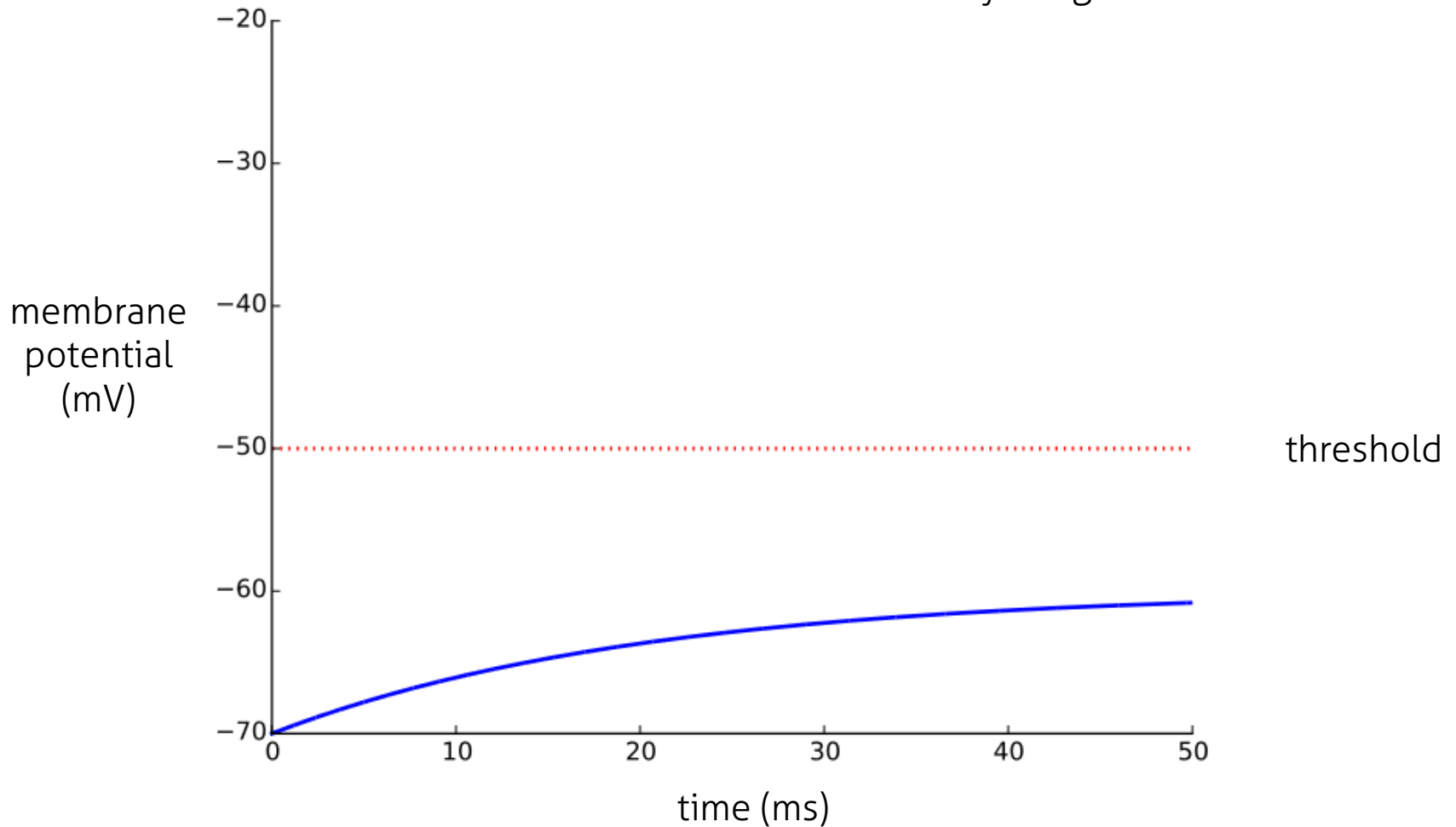
Brian's philosophy:

Mathematical model descriptions

Stimberg M, Goodman DF, Benichoux V, and Brette R (2014)
Equation-Oriented Specification of Neural Models for Simulation
Frontiers in Neuroinformatics

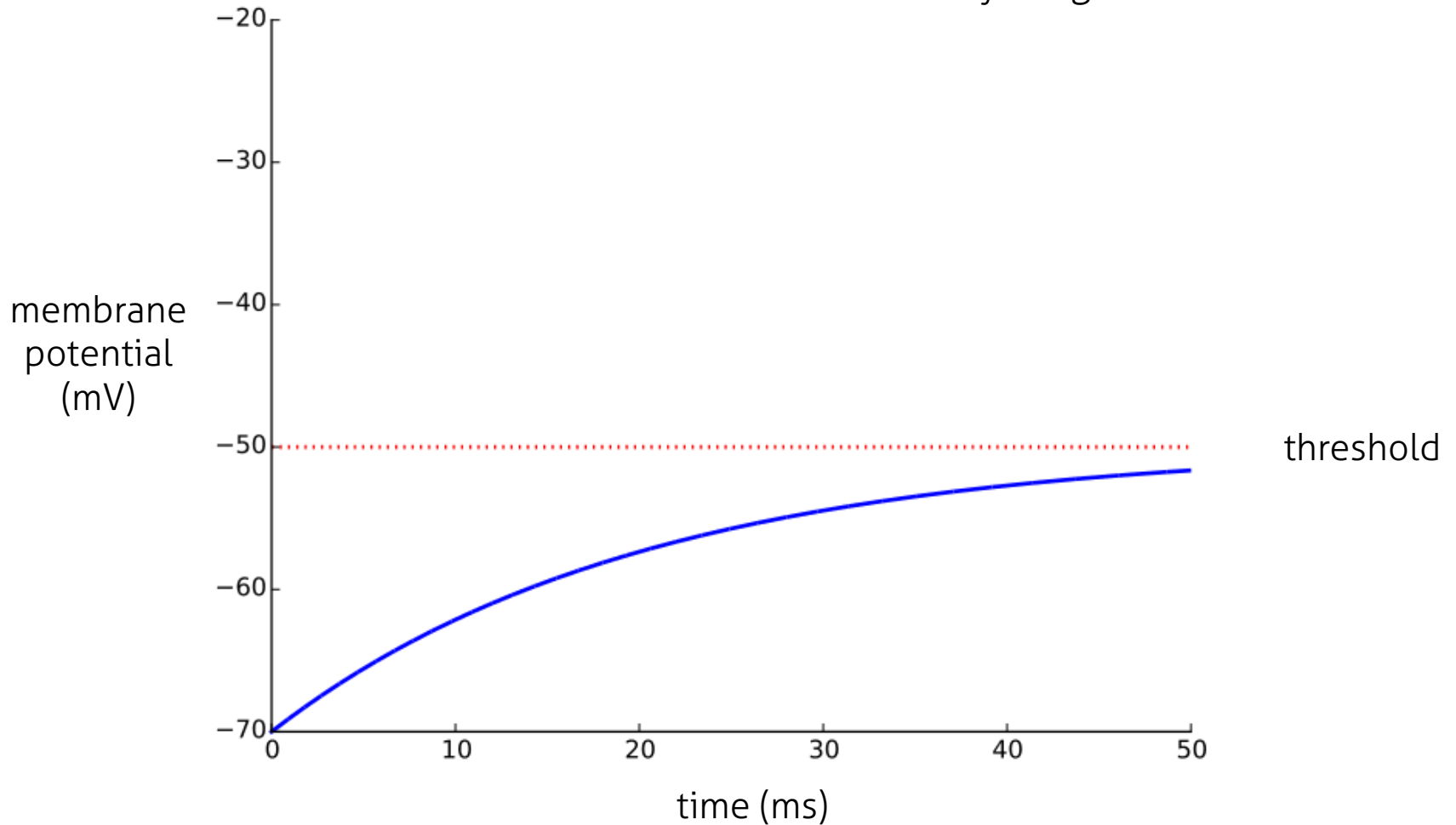
Modelling neural activity

“Leaky integrate-and-fire neuron”

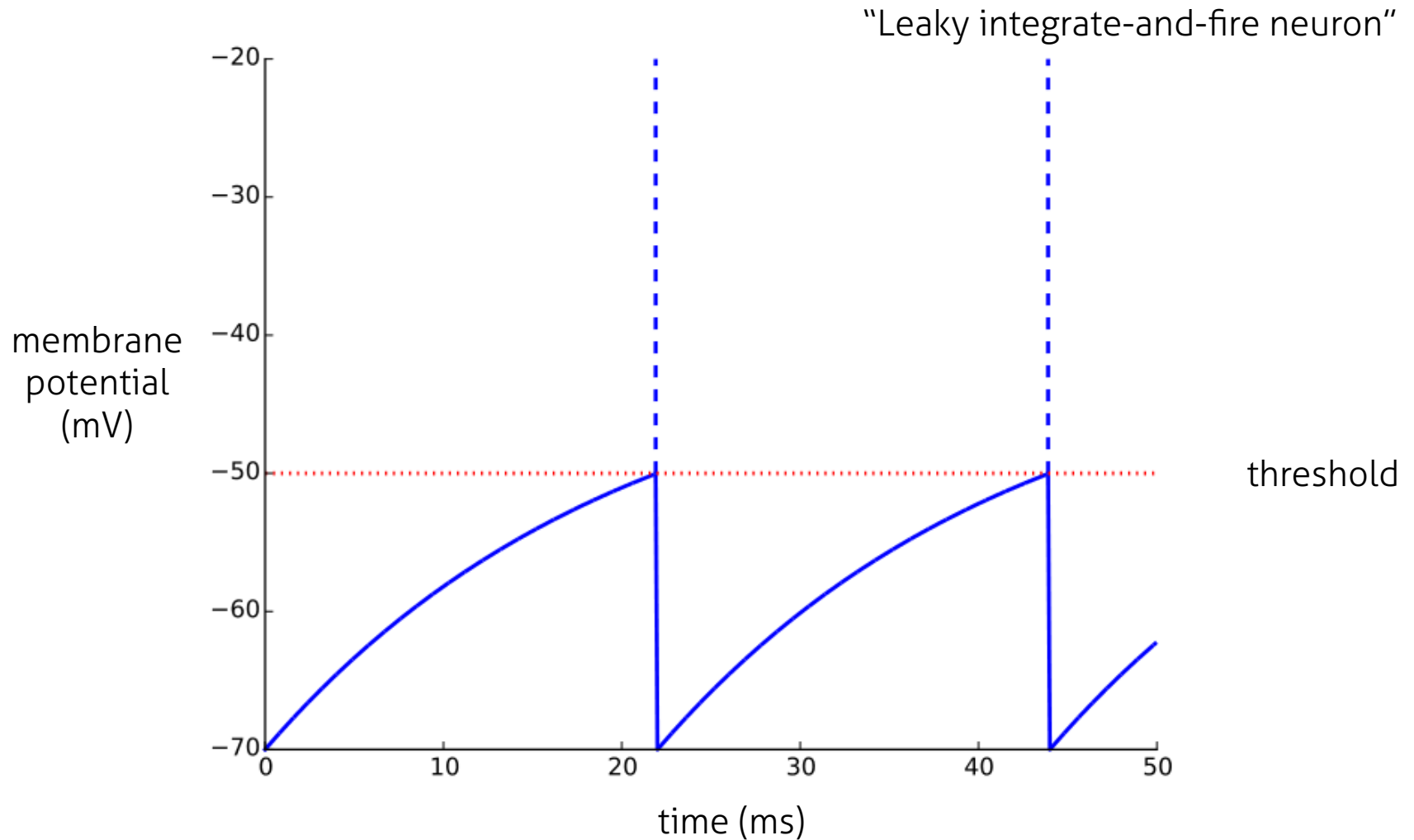


Modelling neural activity

“Leaky integrate-and-fire neuron”



Modelling neural activity



Modelling neural activity

Mathematical description:

$$C_m = 200 \text{ pF}$$

$$g_L = 10 \text{ nS}$$

$$E_L = -70 \text{ mV}$$

$$v_r = E_L$$

$$v_{th} = -50 \text{ mV}$$

$$v_r = E_L$$

$$\frac{dv}{dt} = \frac{1}{C_m} (I + g_L (E_L - v))$$

v has units of volt,
 I has units of amp(ère)

When $v > v_{th}$:

$$v \leftarrow v_r$$

Modelling neural activity

Mathematical description:

$$C_m = 200 \text{ pF}$$

$$g_L = 10 \text{ nS}$$

$$E_L = -70 \text{ mV}$$

$$v_r = E_L$$

$$v_{th} = -50 \text{ mV}$$

$$v_r = E_L$$

$$\frac{dv}{dt} = \frac{1}{C_m} (I + g_L (E_L - v))$$

Constants

v has units of volt,
 I has units of amp(ère)

When $v > v_{th}$:

$$v \leftarrow v_r$$

Modelling neural activity

Mathematical description:

$$C_m = 200 \text{ pF}$$

$$g_L = 10 \text{ nS}$$

$$E_L = -70 \text{ mV}$$

$$v_r = E_L$$

$$v_{th} = -50 \text{ mV}$$

$$v_r = E_L$$

Constants

Continuous dynamics

$$\frac{dv}{dt} = \frac{1}{C_m} (I + g_L (E_L - v))$$

v has units of volt,
 I has units of amp(ère)

When $v > v_{th}$:

$$v \leftarrow v_r$$

Modelling neural activity

Mathematical description:

$$C_m = 200 \text{ pF}$$

$$g_L = 10 \text{ nS}$$

$$E_L = -70 \text{ mV}$$

$$v_r = E_L$$

$$v_{th} = -50 \text{ mV}$$

$$v_r = E_L$$

Constants

Continuous dynamics

$$\frac{dv}{dt} = \frac{1}{C_m} (I + g_L (E_L - v))$$

v has units of volt,
 I has units of amp(ère)

Physical units

When $v > v_{th}$:

$$v \leftarrow v_r$$

Modelling neural activity

Mathematical description:

$$C_m = 200 \text{ pF}$$

$$g_L = 10 \text{ nS}$$

$$E_L = -70 \text{ mV}$$

$$v_r = E_L$$

$$v_{th} = -50 \text{ mV}$$

$$v_r = E_L$$

Constants

Continuous dynamics

$$\frac{dv}{dt} = \frac{1}{C_m} (I + g_L (E_L - v))$$

v has units of volt,
 I has units of amp(ère)

Physical units

When $v > v_{th}$:

$$v \leftarrow v_r$$

Discontinuous state
change

Modelling neural activity

Brian description:

```
Cm = 200*pF
g_L = 10*nS
E_L = -70*mV
v_r = E_L
v_th = -50*mV
```

```
G = NeuronGroup(N,
    '''dv/dt = 1/Cm * (I + g_L * (E_L - v)) : volt
    I : amp''',
    threshold='v > v_th',
    reset='v = v_r')
```

Modelling neural activity

Brian description:

Cm = 200*pF
g_L = 10*nS
E_L = -70*mV
v_r = E_L
v_th = -50*mV

Constants

```
G = NeuronGroup(N,  
    '''dv/dt = 1/Cm * (I + g_L * (E_L - v)) : volt  
    I : amp''',  
    threshold='v > v_th',  
    reset='v = v_r')
```


Modelling neural activity

Brian description:

Cm = 200*pF
g_L = 10*nS
E_L = -70*mV
v_r = E_L
v_th = -50*mV

Constants

G = NeuronGroup(N,

Continuous
dynamics

```
'''dv/dt = 1/Cm * (I + g_L * (E_L - v)) : volt  
I : amp''' ,  
threshold='v > v_th',  
reset='v = v_r')
```

Modelling neural activity

Brian description:

Cm = 200*pF
g_L = 10*nS
E_L = -70*mV
v_r = E_L
v_th = -50*mV

Constants

G = NeuronGroup(N,

Continuous
dynamics

```
'''dv/dt = 1/Cm * (I + g_L * (E_L - v)) : volt  
I : amp''',  
threshold='v > v_th',  
reset='v = v_r')
```

Physical units

Modelling neural activity

Brian description:

Cm = 200*pF
g_L = 10*nS
E_L = -70*mV
v_r = E_L
v_th = -50*mV

Constants

G = NeuronGroup(N,

*'''dv/dt = 1/Cm * (I + g_L * (E_L - v)) : volt
I : amp'''*,
threshold='v > v_th',
reset='v = v_r')

Continuous
dynamics

Discontinuous state
change

Physical units

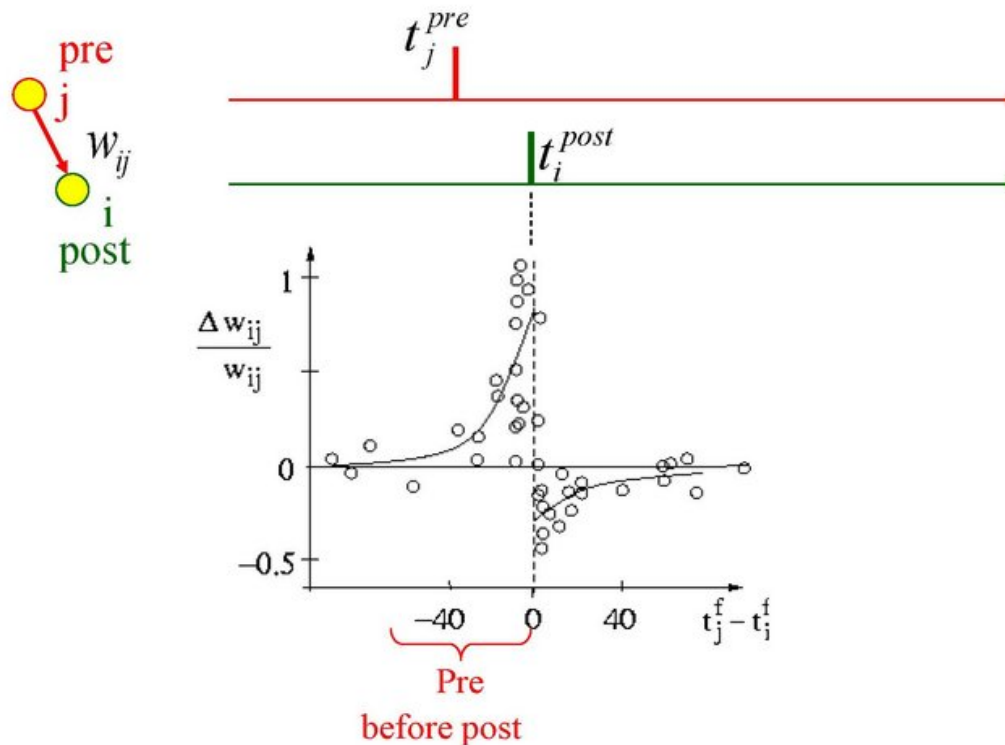
Part 1: Modelling neurons

[see jupyter notebook]

Part 2: Modeling synapses

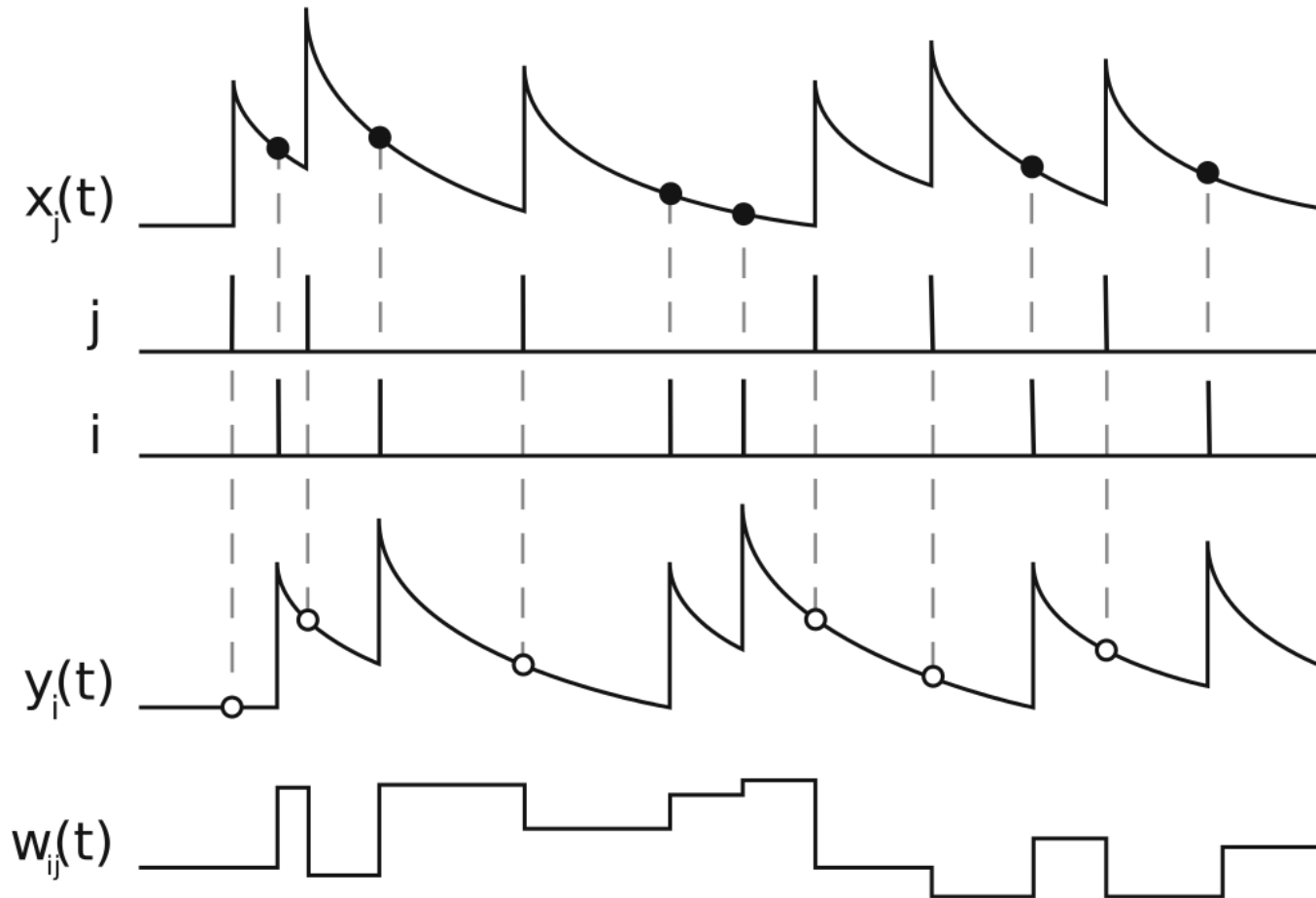
Spike-timing dependent plasticity

STDP



Sjöström & Gerstner, Scholarpedia (2010)
Redrawn after Bi & Po, J Neurosci (1998)

STDP: online implementation



[see jupyter notebook]

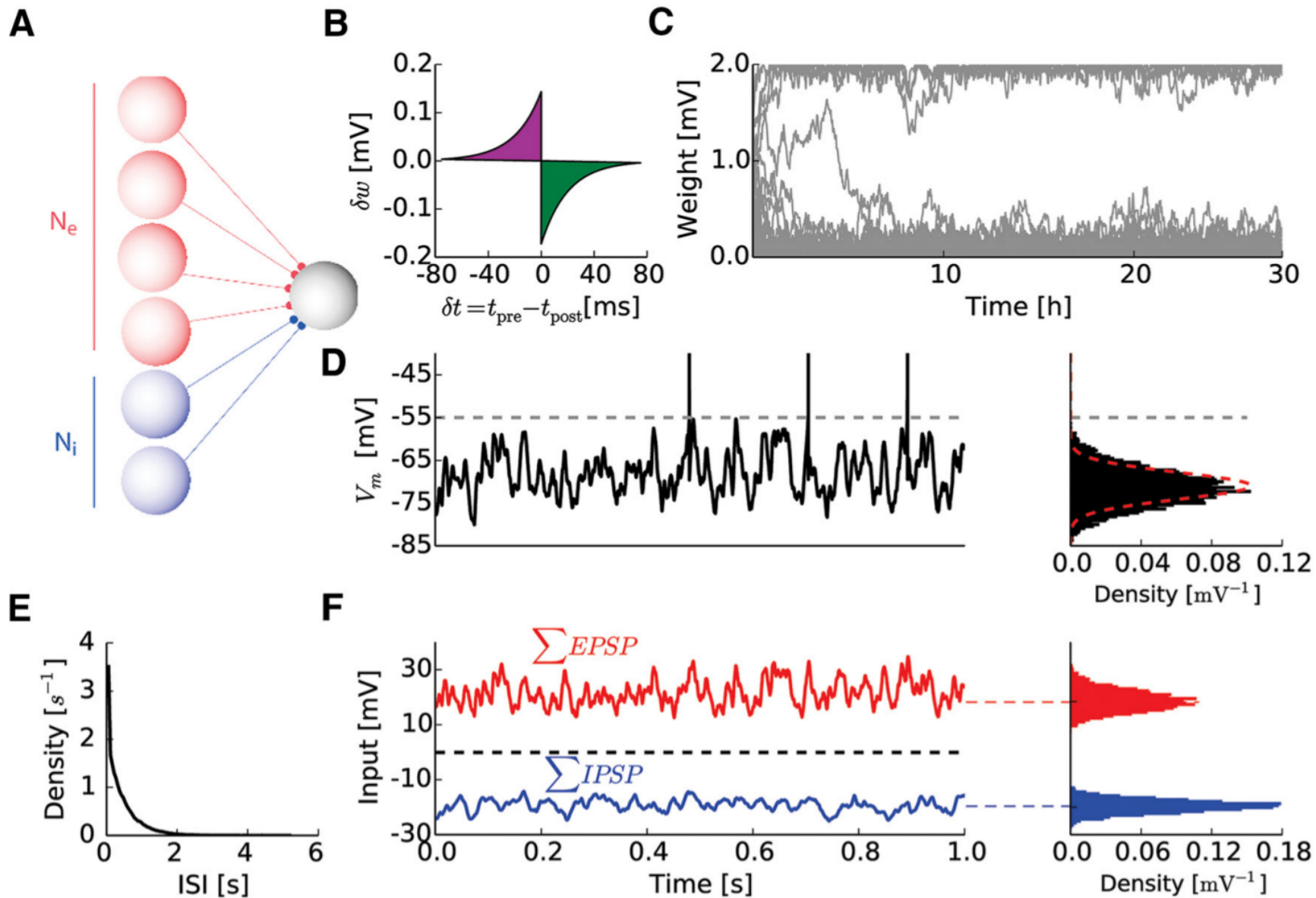
Part 3: Case Study

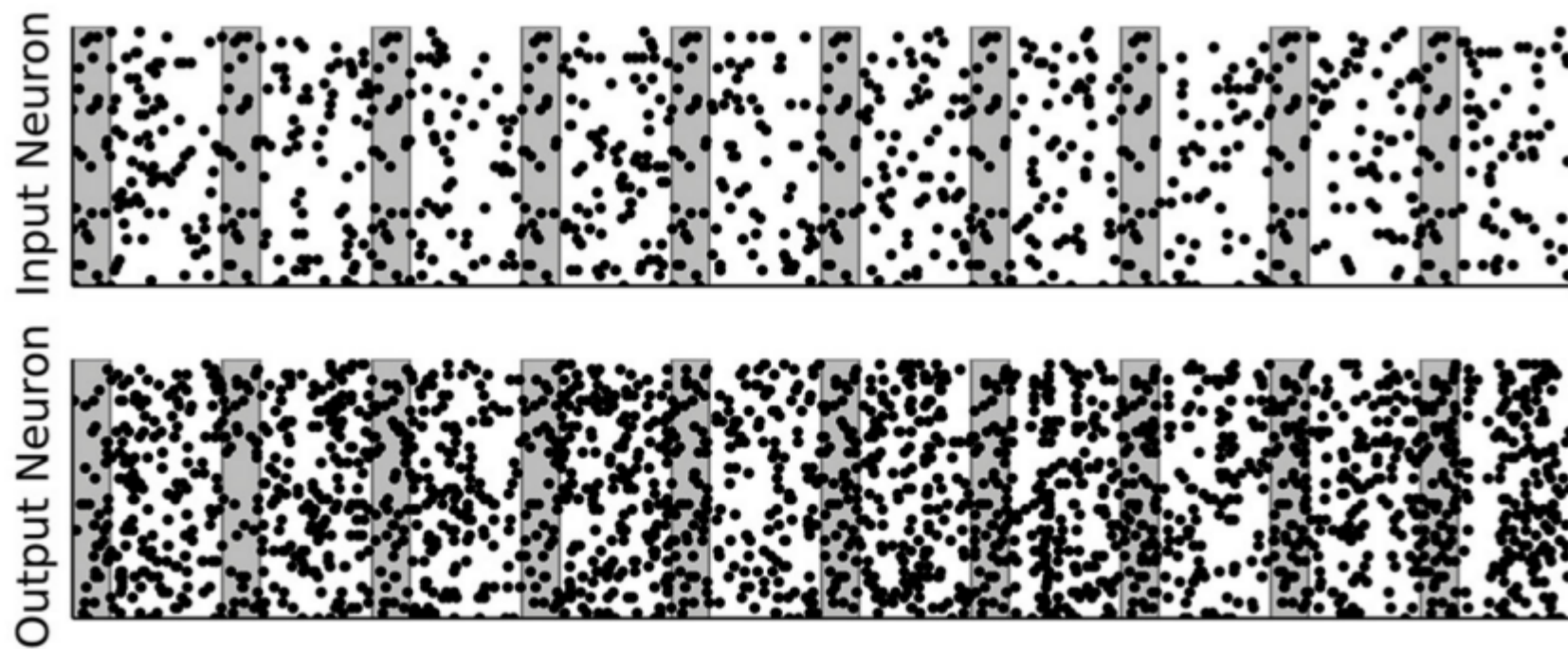
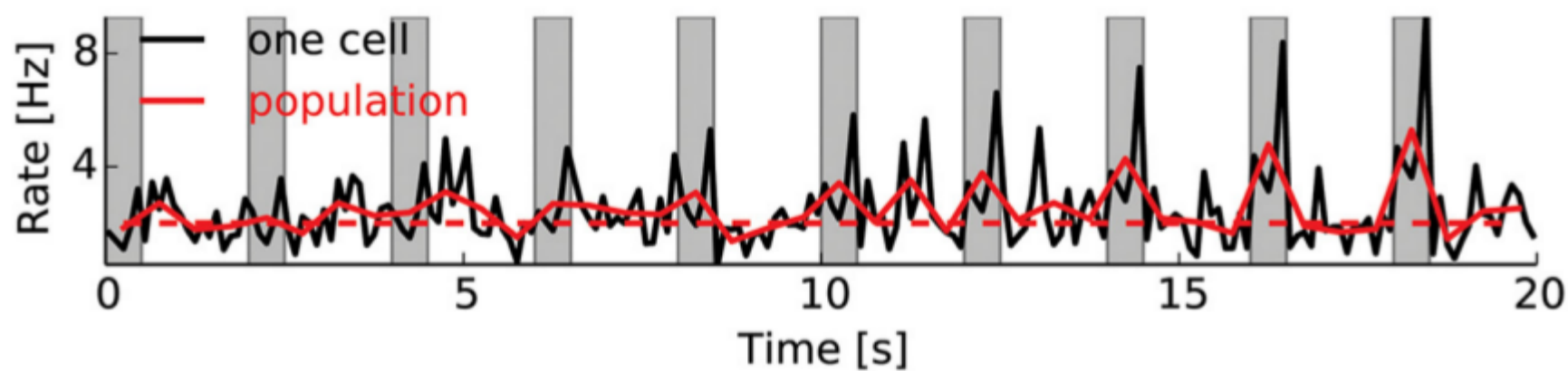
Systems/Circuits

Fast Learning with Weak Synaptic Plasticity

Pierre Yger,^{1,2,3}  Marcel Stimberg,^{2,3} and Romain Brette^{2,3}

¹Institut d'Etudes de la Cognition, Ecole Normale Supérieure, 75005 Paris, France, ²Sorbonne Université, UPMC Université Paris 06 UMRS968, 75006 Paris, France, and ³Institut de la Vision, INSERM U968, CNRS UMR7210, 75012 Paris, France



A**B**

[see jupyter notebook]

Further material about Brian 2

Website

briansimulator.org



The Brian spiking neural network simulator

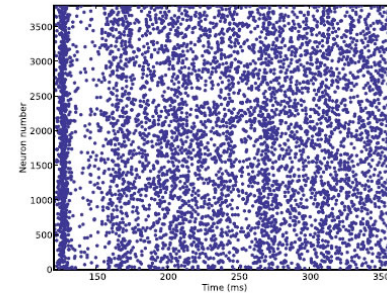
Posts Comments

About
Install
Documentation
Tutorials
Examples
Interactive demo
Support
Showcase
Publications
Contribute
Team
Twitter
Facebook

Brian is a free, open source simulator for spiking neural networks. It is written in the Python programming language and is available on almost all platforms. We believe that a simulator should not only save the time of processors, but also the time of scientists. Brian is therefore designed to be easy to learn and use, highly flexible and easily extensible.

To get an idea of what writing a simulation in Brian looks like, the following code defines a randomly connected network of integrate and fire neurons with exponential inhibitory and excitatory currents, runs the simulation and makes the raster plot on the right.

```
1 from brian2 import *
2 eqs = '''
3 dv/dt = (ge+gi-(v+49*mV))/(20*ms) : volt
4 dge/dt = -ge/(5*ms) : volt
5 dgi/dt = -gi/(10*ms) : volt
6 '''
7 P = NeuronGroup(4000, eqs, threshold='v>-50*mV', reset='v=-60*mV')
8 P.v = -60*mV
9 Pe = P[:3200]
10 Pi = P[3200:]
11 Ce = Synapses(Pe, P, on_pre='ge+=1.62*mV')
12 Ce.connect(p=0.02)
13 Ci = Synapses(Pi, P, on_pre='gi-=9*mV')
14 Ci.connect(p=0.02)
15 M = SpikeMonitor(P)
16 run(1*second)
17 plot(M.t/ms, M.i, '.')
18 show()
```



Interactive demo

We also have an interactive demo (running on mybinder.org) that lets you modify parameters and even change the simulation code, running from the web browser without installing anything.

Getting started

Once you've decided you want to use Brian, you can click the links on the left hand side to install Brian, go through the tutorials, look at example code and read the full documentation. If you have problems, we have an email support list.

Other software and tools

We have produced the following tools in addition to the main Brian simulator:

- brian2tools: for simple plotting and analysis with Brian

You can also download the older version of Brian (1.4) and the toolboxes we designed for it:

- Brian Hears (compatible with Brian 2): for modelling the auditory system
- Model fitting toolbox: for automatically fitting models to electrophysiological data

How to cite Brian

If you use Brian for your published research, we suggest that you cite one of our introductory articles:

1. Goodman DF and Brette R (2009). The Brian simulator. *Front Neurosci* doi:10.3389/neuro.01.026.2009
2. Stimberg M, Goodman DFM, Benichoux V, Brette R (2014). Equation-oriented specification of neural models for simulations. *Frontiers Neuroinf*, doi: 10.3389/fninf.2014.00006.

You can also download our logo for posters and presentations.

The Brian team

Brian is being developed by:

- Romain Brette (l'Institut de la Vision, Paris)

News

Brian 1.4.4 (maintenance release)

11 Dec 2017

Brian 2.1.2 and Brian2GeNN 1.1.5

8 Nov 2017

Brian 2.1 and Brian2GeNN 1.1

31 Oct 2017

Brian2GeNN 1.0

29 Jun 2017

Development

Brian 1.4.4 (maintenance release)

11 Dec 2017

Brian 2.1.2 and Brian2GeNN 1.1.5

8 Nov 2017

Brian 2.1 and Brian2GeNN 1.1


31 Oct 2017

Brian2GeNN 1.0

29 Jun 2017


Documentation

brian2.readthedocs.org



stable


[Introduction](#)
[Tutorials](#)
[User's guide](#)
[Advanced guide](#)
[Examples](#)
[Reference documentation](#)
[Developer's guide](#)



SMS API for Python applications

Make and receive SMS messages in your applications with just a few lines of Python code.

Ads served ethically

 Read the Docs v: stable

[Docs](#) » Brian 2 documentation

[Edit on GitHub](#)

Brian 2 documentation

Brian is a simulator for spiking neural networks. It is written in the Python programming language and is available on almost all platforms. We believe that a simulator should not only save the time of processors, but also the time of scientists. Brian is therefore designed to be easy to learn and use, highly flexible and easily extensible.

To get an idea of what writing a simulation in Brian looks like, take a look at [a simple example](#), or run our [interactive demo](#).

You can actually edit and run the examples in the browser without having to install Brian, using the Binder service (note: sometimes this service is down or running slowly):

[launch binder](#)

Once you have a feel for what is involved in using Brian, we recommend you start by following the [installation instructions](#), then going through the [tutorials](#), and finally reading the [User Guide](#).

While reading the documentation, you will see the names of certain functions and classes are highlighted links (e.g. `PoissonGroup`). Clicking on these will take you to the "reference documentation". This section is automatically generated from the code, and includes complete and very detailed information, so for new users we recommend sticking to the [User's guide](#). However, there is one feature that may be useful for all users. If you click on, for example, `PoissonGroup`, and scroll down to the bottom, you'll get a list of all the example code that uses `PoissonGroup`. This is available for each class or method, and can be helpful in understanding how a feature works.



Finally, if you're having problems, please do let us know at our [support page](#).

Contents:

- [Introduction](#)
 - [Installation](#)
 - [Release notes](#)
 - [Changes for Brian 1 users](#)
 - [Known issues](#)
 - [Support](#)
- [Tutorials](#)
 - [Introduction to Brian part 1: Neurons](#)
 - [Introduction to Brian part 2: Synapses](#)


Mailing list



briansupport@googlegroups.com
groups.google.com/forum/#!forum/briansupport

Groups

NEW TOPIC



Home

Click on a group's star icon to add it to your favorites


Recently viewed


Brian



















[Privacy](#) - [Terms of Service](#)

Brian

Shared publicly


30 of many topics 


About 

 NaNs with simple model By Dylan Muir - 2 posts - 1 view	5:16 PM
 [Ann] Release: Brian 1.4.4 By Marcel Stimberg - 1 post - 2 views	Dec 11
 Synapse connection By hon...@gmail.com - 5 posts - 8 views	Dec 7
 Mixed event/clock-driven synapses By austin...@gmail.com - 2 posts - 4 views	Dec 6
 Unable to install Jupyter Notebook from Anaconda By prupl...@gmail.com - 2 posts - 4 views	Dec 5
 warnings when testing brian2.test() By just...@gmail.com - 3 posts - 6 views	Nov 24
 Some puzzles about the example on the Brian2 official site By huguy...@gmail.com - 2 posts - 15 views	Nov 20
 Network topology By Henok Solomon - 5 posts - 12 views	Nov 19
 Pausing Plasticity By Maria Kesa - 2 posts - 14 views	Nov 13
 Question about training the network the have different SubGroups? By Qianhui Liu - 7 posts - 39 views	Nov 13
 A neuron definition question By Qianhui Liu - 7 posts - 41 views	Nov 10
 synapses with homeostatic placticity By Mohammad Reza Soltanipour - 6 posts - 29 views	Nov 9
 [Ann] Release: Brian 2.1.2 and Brian2GeNN 1.1.5 By Marcel Stimberg - 1 post - 12 views	Nov 8
 Izhikevich neuron model in Brian2 By set...@gmail.com - 2 posts - 16 views	Nov 7
 Clear function By flora.bo...@gmail.com - 4 posts - 6 views	Nov 2
 [Ann] Release: Brian 2.1 and Brian2GeNN 1.1 By Marcel Stimberg - 1 post - 2 views	Oct 31
 Problem with timed arrays in Brian2 By danielc...@gmail.com - 2 posts - 11 views	Oct 17
 Synapse problem By Franz - 20 posts - 212 views	Oct 16

Code repository

github.com/brian-team/brian2

 This repository Search Pull requests Issues Marketplace Explore

 **brian-team / brian2** Unwatch 27 Unstar 151 Fork 51

[Code](#) [Issues 125](#) [Pull requests 4](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)


Brian is a free, open source simulator for spiking neural networks. <http://briansimulator.org> [Edit](#)

[python](#) [neuroscience](#) [science](#) [differential-equations](#) [spiking-neural-networks](#) [biological-simulations](#) [code-generation](#) [simulation-framework](#)

[brian](#) [Manage topics](#)

4,409 commits 14 branches 23 releases 18 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

 **mstimmerberg** Merge pull request #901 from brian-team/fix_numpy_logical_operators Latest commit c02d40c a day ago

brian2	Use '&' and ' ' instead of '*' and '+' for logical operations in numpy	5 days ago
dev	Set the version to 2.1.2+git for the next development cycle	14 days ago
docs_sphinx	Merge branch 'fix_doc_issues'	14 days ago
examples	Merge branch 'master' into GSL_final	2 months ago
tutorials	Update tutorials ('linear' --> 'exact')	2 months ago
.coveragerc	Exclude Sphinx extension from test coverage measurements	4 years ago
.gitattributes	Add a .gitattributes file	3 years ago
.gitignore	Added randomkit to C++ standalone	2 years ago
.gitmodules	Use the https instead of ssh url for the submodule	3 years ago
.travis.yml	Update the anaconda authentication token (old one was expired)	2 months ago
.travis_long.yml	Add a travis file for long testing	3 years ago
AUTHORS	Update release notes and authors file for upcoming release	2 months ago
LICENSE	Add randomkit to the LICENSE file	2 years ago
MANIFEST.in	Add the Brian logo to the documentation	a year ago
README.rst	works?	4 months ago
appveyor.yml	Update the anaconda authentication token (old one was expired)	2 months ago
readthedocs.yml	Try configuring readthedocs documentation via a YAML file and use a f...	3 months ago
rtd_environment.yml	Simplify the dependencies for the rtd_environment, and let conda figu...	3 months ago
setup.cfg	Switch setup.py to setuptools, enable nosetests and sphinx	4 years ago
setup.py	Set the version to 2.1.2+git for the next development cycle	14 days ago

Installation

Recommended way:

- Use Anaconda distribution
- Add the "brian-team" channel

```
conda config --add channels brian-team
```

- Install brian2

```
conda install brian2
```

More infos and alternative installation:

brian2.readthedocs.org