Workshop Series Summer 2023

CAMBAM

Centre for Applied Mathematics
in Bioscience and Medicine

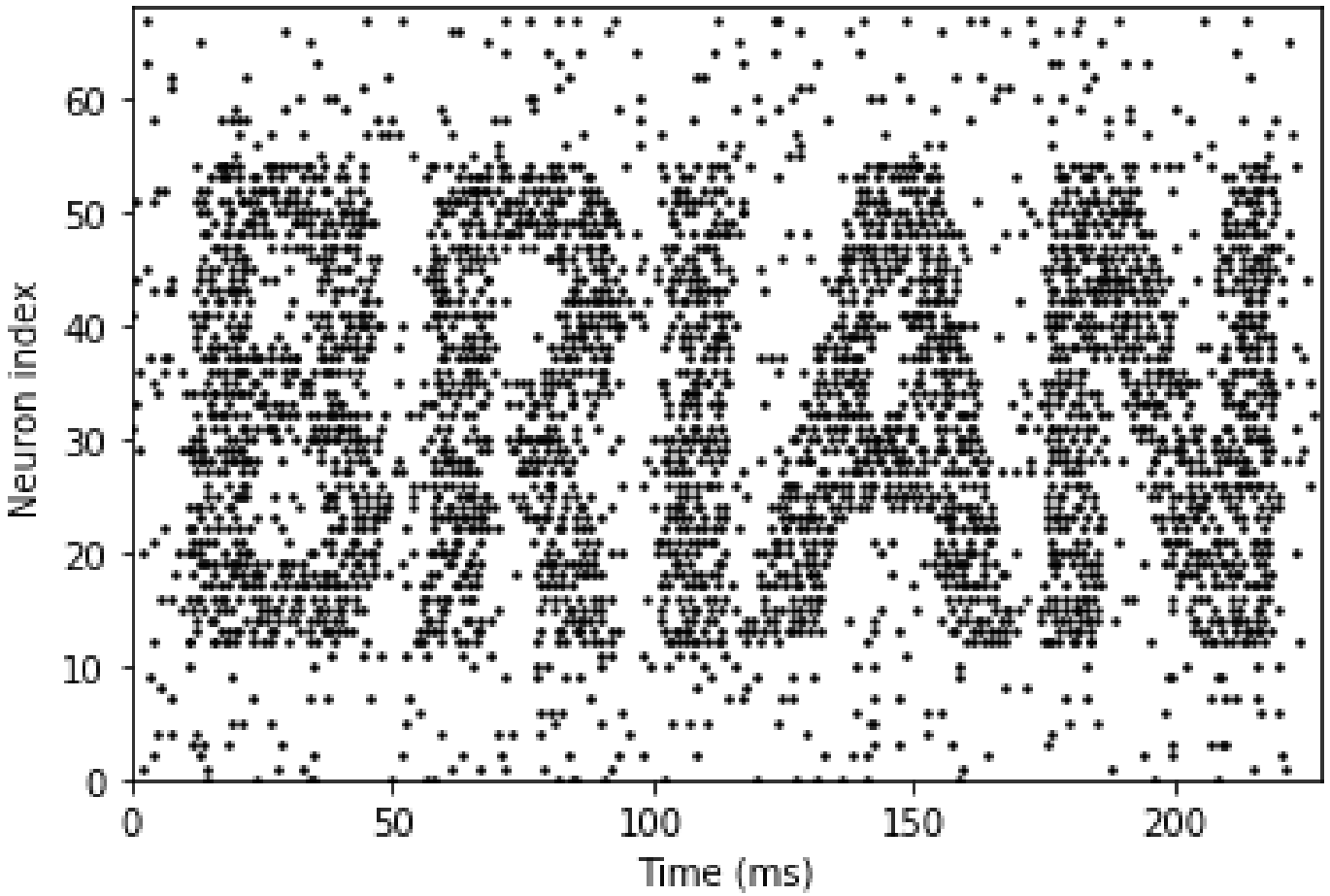
June 16th, 2023

Exploring Single Neuron Excitability with Mathematical and Computational Models

By Niklas Brake and Nils Koch

Lecture 3: Model Fitting and Brian2

Part 3: Brian2



https://brian2.readthedocs.io

Part 3: Brian2

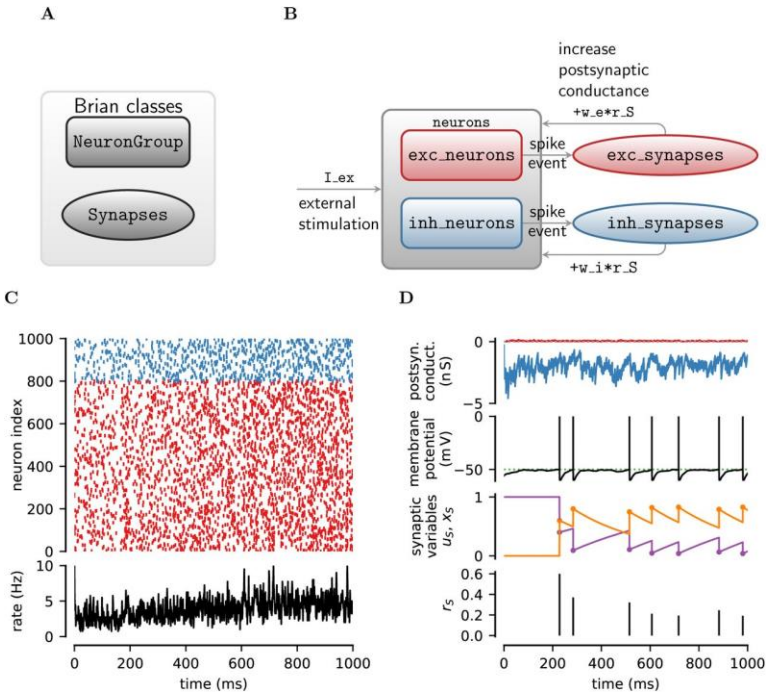
https://briansimulator.org/

https://brian2.readthedocs.io

Brian2

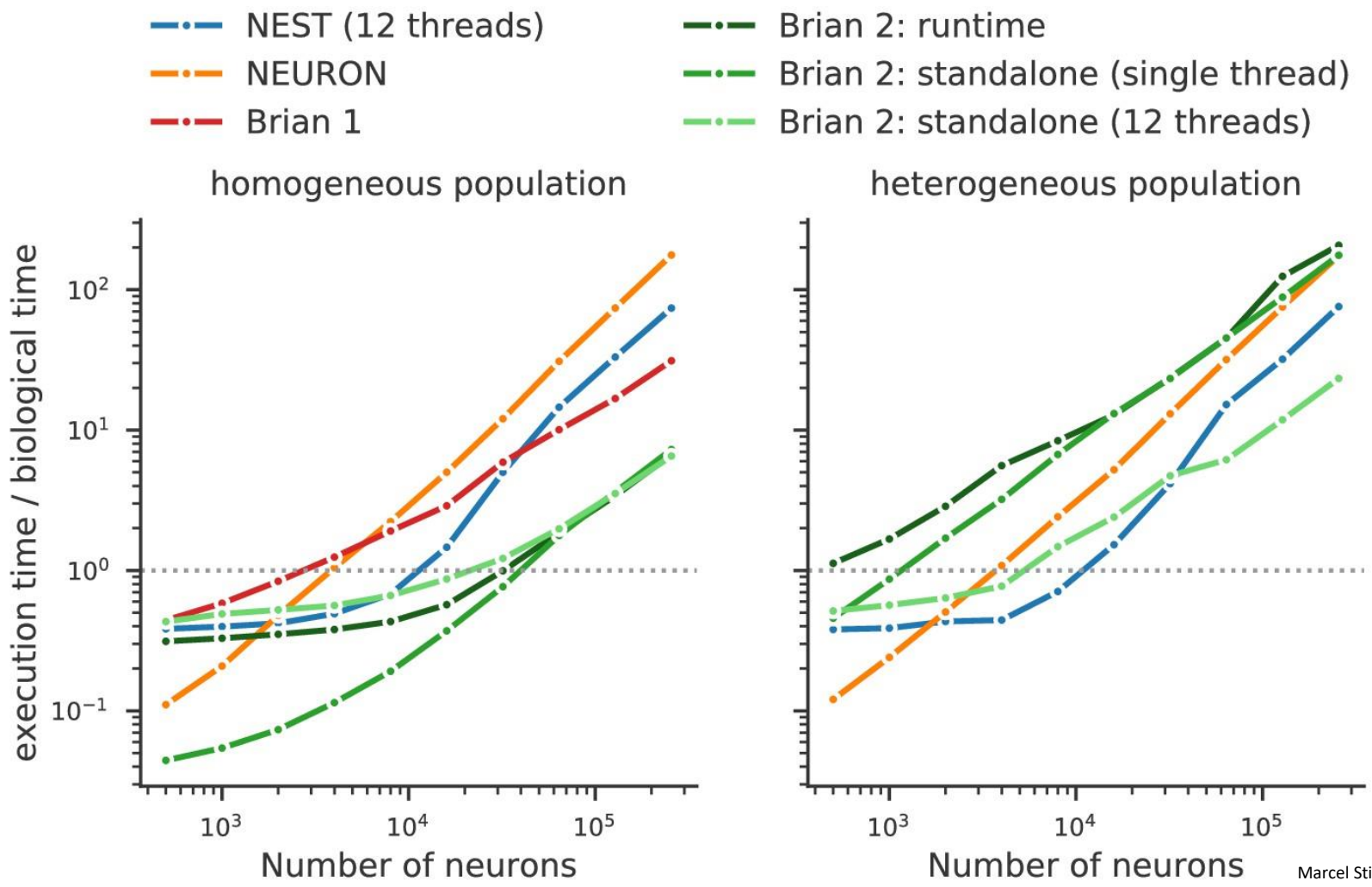
Free, open-source Python simulator for spiking neural network

- Easy to use
  - Easy to understand syntax to define, run and plot neural models in just a few lines of code
- Flexible
  - Any model: write equations in standard mathematical notation and run
- Performance
  - Automatically converting your equations into low level C++ code, compiling and running
- Reliable
  - widely used: https://briansimulator.org/papers-using-brian/



Marcel Stimberg, Dan F. M. Goodman, Romain Brette, Maurizio De Pittà. Modeling neuron–glia interactions with the *Brian 2* simulator. bioRxiv 198366

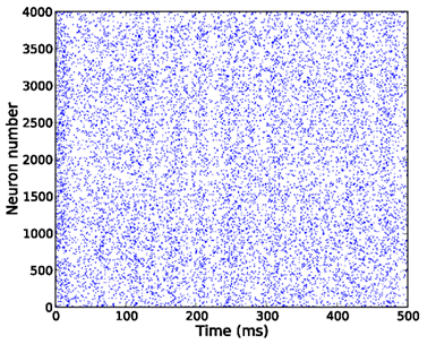
Part 3: Brian2



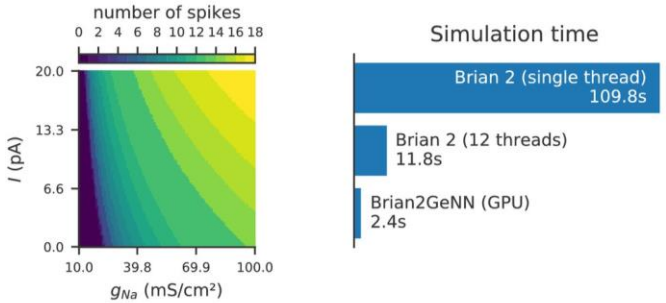
Marcel Stimberg, Romain Brette, Dan FM Goodman (2019) Brian 2, an intuitive and efficient neural simulator eLife 8:e47314

Part 3: Brian2

```
from brian import *
eqs = '''
dV/dt  = (ge+gi-(V+49*mV))/(20*ms) : volt
dge/dt = -ge/(5*ms)                : volt
dgi/dt = -gi/(10*ms)               : volt
'''
P = NeuronGroup(4000, model=eqs,
        threshold=-50*mV, reset=-60*mV)
Pe = P.subgroup(3200)
Pi = P.subgroup(800)
Ce = Connection(Pe, P, 'ge')
Ci = Connection(Pi, P, 'gi')
Ce.connect_random(Pe, P, p=0.02,
                  weight=1.62*mV)
Ci.connect_random(Pi, P, p=0.02,
                  weight=-9*mV)
M = SpikeMonitor(P)
P.V = -60*mV+10*mV*rand(len(P))
run(.5*second)
raster_plot(M)
show()
```



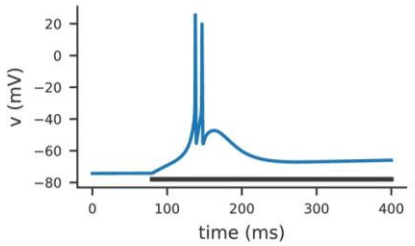
Goodman, Dan & Brette, Romain. (2008). Brian: A Simulator for Spiking Neural Networks in Python. Frontiers in neuroinformatics. 2. 5. 10.3389/neuro.11.005.2008.

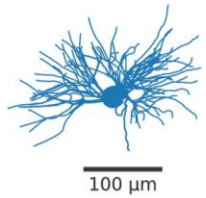


(b)

```
from brian2 import *
import brian2genn
set_device('genn')
# HH model with injected current
area = 20000*umetre**2; Cm = (1*ufarad*cm**-2) * area;
gl = (5e-5*siemens*cm**-2) * area; El = -60*mV; EK = -90*mV; ENa = 50*mV;
g_kd = (30*msiemens*cm**-2) * area; VT = -63*mV
eqs = Equations('''
dv/dt = (gl*(El-v)- g_na*(m*m*m)*h*(v-ENa)- g_kd*(n*n*n*n)*(v-EK) + I)/Cm : volt
dm/dt = alpha_m*(1-m)-beta_m*m : 1
dn/dt = alpha_n*(1-n)-beta_n*n : 1
dh/dt = alpha_h*(1-h)-beta_h*h : 1
alpha_m = 0.32*(mV**-1)*(13*mV-v+VT)/ (exp((13*mV-v+VT)/(4*mV))-1.)/ms : Hz
beta_m = 0.28*(mV**-1)*(v-VT-40*mV)/ (exp((v-VT-40*mV)/(5*mV))-1)/ms : Hz
alpha_h = 0.128*exp((17*mV-v+VT)/(18*mV))/ms : Hz
beta_h = 4./(1+exp((40*mV-v+VT)/(5*mV)))/ms : Hz
alpha_n = 0.032*(mV**-1)*(15*mV-v+VT)/ (exp((15*mV-v+VT)/(5*mV))-1.)/ms : Hz
beta_n = .5*exp((10*mV-v+VT)/(40*mV))/ms : Hz
I : amp (constant)
g_na : siemens (constant)
''')
# Explore 300 * 300 values
g_na_values, I_values = np.linspace(10, 100, num=300), np.linspace(0, 20, num=300)
neuron = NeuronGroup(len(g_na_values)*len(I_values), eqs, method='exponential_euler',
                     threshold='v>-20*mV', refractory='v>-20*mV')
neuron.v = El
spike_mon = SpikeMonitor(neuron)
all_g_na_values, all_I_values = np.meshgrid(g_na_values, I_values)
all_g_na_values, all_I_values = all_g_na_values.flat[:], all_I_values.flat[:]
neuron.g_na, neuron.I = all_g_na_values*msiemens*cm**-2 * area, all_I_values*pA

run(1*second)
```





(b)

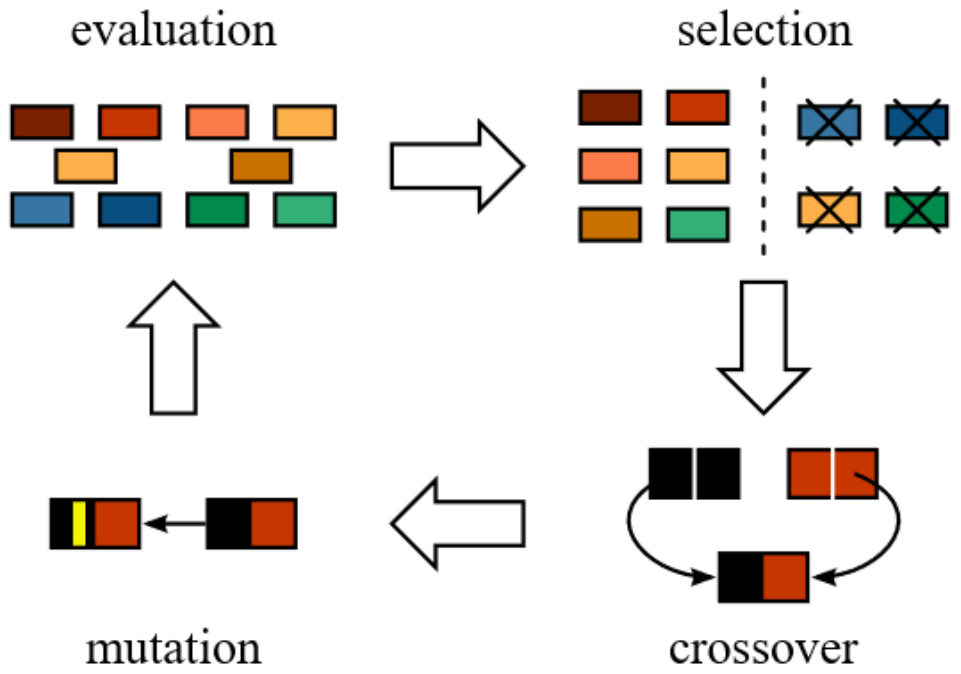
```
# Constants
El = -76.5*mV; E_Na = 50*mV; E_K = -100*mV
# ...
eqs = Equations('''
Im = gl*(El-v) - I_Na - I_K - I_T: amp/meter**2
I_inj : amp (point current)

# HH-type currents for spike initiation
g_Na : siemens/meter**2
I_Na = g_Na * m**3 * h * (v-E_Na) : amp/meter**2
v2 = v - VT : volt  # shifted membrane potential (Traub convention)
dm/dt = (0.32*(mV**-1)*(13.*mV-v2)/
        (exp((13.*mV-v2)/(4.*mV))-1.)*(1-m)-0.28*(mV**-1)*(v2-40.*mV)/
        (exp((v2-40.*mV)/(5.*mV))-1.)*m) / ms * tadj_HH: 1
# ...
''')
# Load morphology from SWC file
morpho = Morphology.from_file('tc200.CNG.swc')
neuron = SpatialNeuron(morpho, eqs, Cm=0.88*uF/cm**2, Ri=173*ohm*cm,
                       method='exponential_euler')
# Only the soma has Na/K channels
neuron.main.g_Na = 100*msiemens/cm**2
neuron.main.g_K = 100*msiemens/cm**2
neuron.P_Ca = 1.7e-5*cm/second
# Distal dendrites
neuron.P_Ca['(distance + length/2) > 11*um'] = 8.5e-5*cm/second
neuron.v = -74*mV
neuron.m_T = 'm_T_inf'
neuron.h_T = 'h_T_inf'
mon = StateMonitor(neuron, ['v'], record=morpho[0])  # Record at soma
```

Marcel Stimberg, Romain Brette, Dan FM Goodman (2019) Brian 2, an intuitive and efficient neural simulator eLife 8:e47314

Part 3: Brian2

- Many different optimization algorithms and packages
  - PyGAD



https://www.electricalelibrary.com/en/2018/04/13/what-is-genetic-algorithm/

Break: 15:00 – 15:15

Workshop Series Summer 2023

CAMB AM

Centre for Applied Mathematics
in Bioscience and Medicine

June 16th, 2023

Exploring Single Neuron Excitability with Mathematical and Computational Models