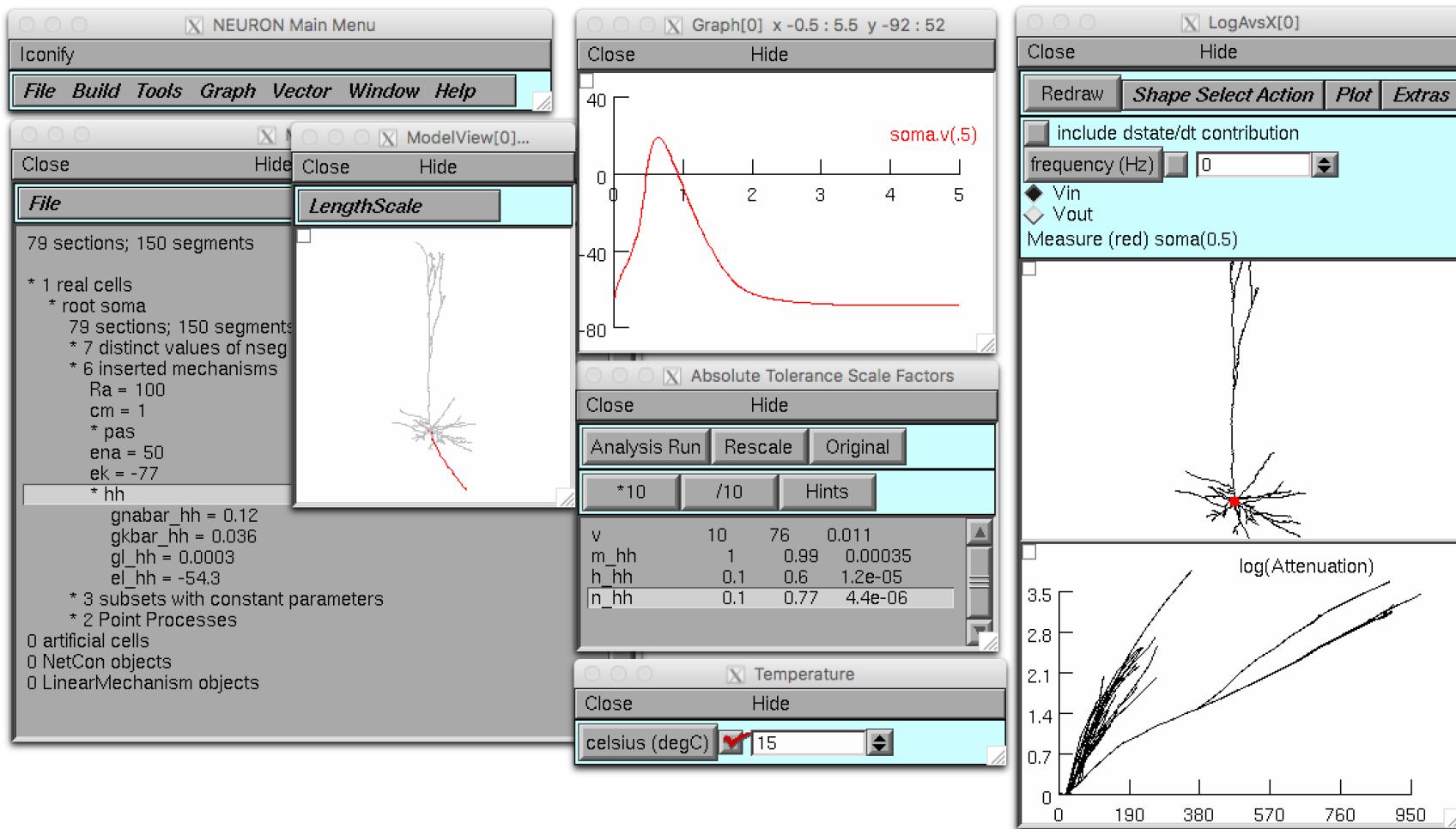


# NEURON

<http://neuron.yale.edu>

NEURON is a tool for *developing, simulating, and analysing* empirically-based models of neurons and networks of neurons. NEURON supports all classes of spiking models and runs on both desktops and supercomputers.



Powerful GUI tools • Fully Python scriptable • Large networks and single cells • Morphologically and biophysically detailed cells, integrate-and-fire cells, and anything in between • Run on a single core or on 128,000 processors.

# Plans and in development

## Features

- Standards support: NeuroML, SBML.
- Extracellular reaction-diffusion (rxn).
- Stochastic rxn simulations.
- 3D intracellular rxn simulation.

## Performance enhancements

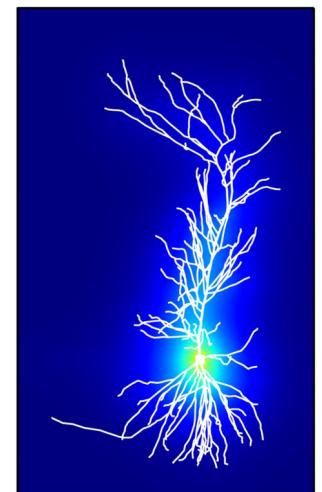
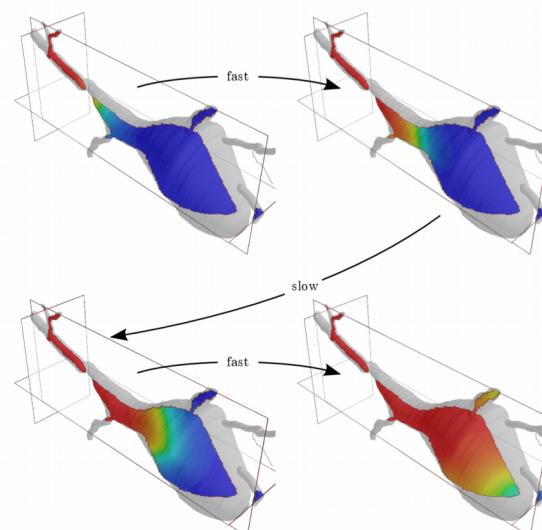
- GPU support.
- Faster reaction-diffusion.

## Better documentation

- Recently released Python programmer's reference.



Migliore et al 2014. Olfactory bulb network model. Up to 69,000 cells. [modeldb.yale.edu/151681](http://modeldb.yale.edu/151681)



3D intra- (left) and extracellular (right) reaction-diffusion simulations.

# More NEURON Resources

**API documentation** (both Python and HOC):

[https://neuron.yale.edu/neuron/static/py\\_doc/index.html](https://neuron.yale.edu/neuron/static/py_doc/index.html)

**ModelDB** (over 575 NEURON models):

<http://modeldb.yale.edu>

**NEURON forum** (over 14,000 posts):

<https://neuron.yale.edu/phpBB/>

**Tutorials:**

<http://neuron.yale.edu/neuron/docs>

**NEURON courses:**

Week-long NEURON course every summer.

Day-long NEURON course before each Society for Neuroscience conference.

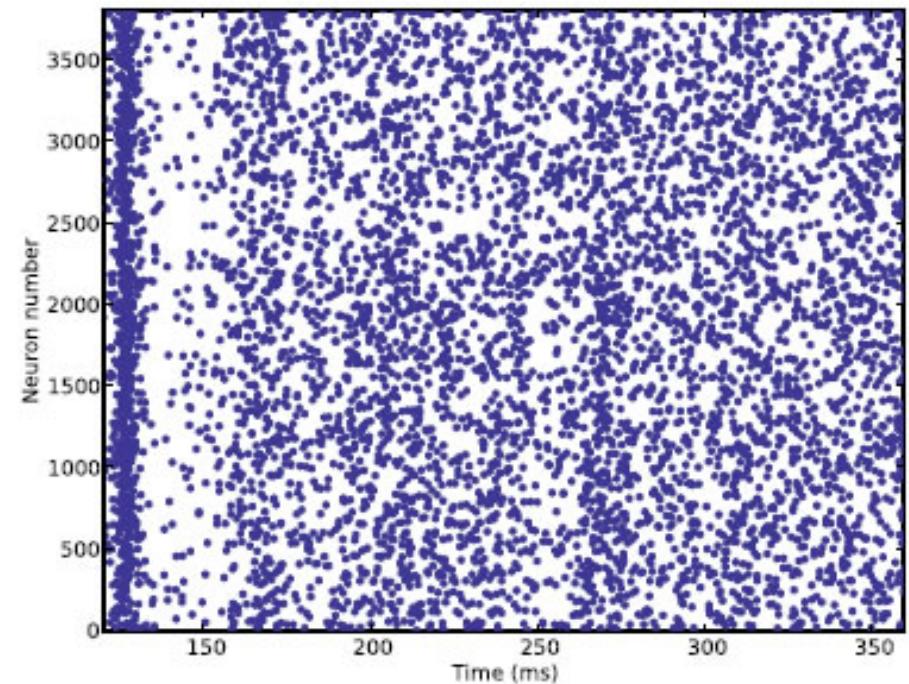
# Brian 2

<http://briansimulator.org>

Brian 2 is a free, open source simulator for spiking neural networks. It is designed to be easy to learn and use, highly flexible and easily extensible.

Brian 2 allows for concise but complete descriptions of neural and synaptic models, based on mathematical equations and physical units.

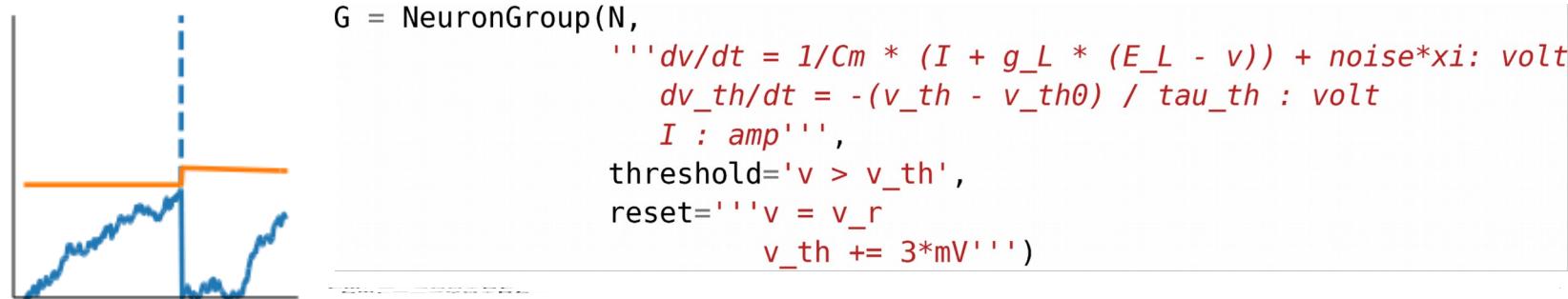
```
from brian2 import *
eqs = '''
dv/dt  = (ge+gi-(v+49*mV))/(20*ms) : volt
dge/dt = -ge/(5*ms)                  : volt
dgi/dt = -gi/(10*ms)                 : volt
'''
P = NeuronGroup(4000, eqs, threshold='v>-50*mV',
                 reset='v=-60*mV')
P.v = -60*mV
Pe = P[:3200]
Pi = P[3200:]
Ce = Synapses(Pe,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



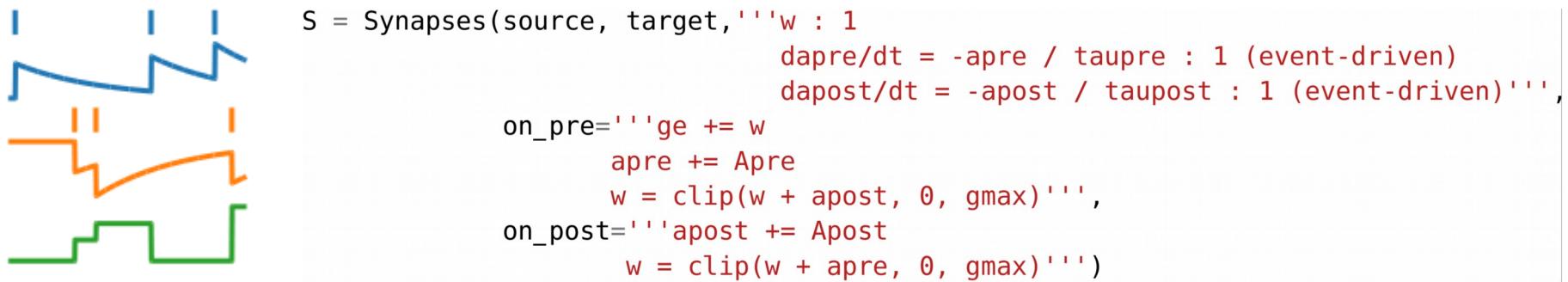
The user-provided model descriptions are transformed into executable code in a low-level language (e.g. C++). This *code generation* approach permits to benefit from the flexibility and ease-of-use of a high-level language with the execution speed of a low-level language.

General formalism allows to define a wide range of neural and synaptic models

### Noisy adaptive integrate-and-fire neuron



### Spike-timing dependent plasticity (STDP)



Simulations can be executed on a variety of targets

“**runtime mode**” – generated code is executed directly from the Python main loop  
→ generated code can be numpy or C++ code

“**standalone mode**” – generated code is written to disk and executed separately  
→ C++ code, with optional support for multiple processor cores via OpenMP  
→ with *Brian2GeNN*: C++ code for the GeNN simulator, compiling into CUDA code for the GPU

# Resources for Brian 2

Brian documentation (including examples and tutorials)

<http://brian2.readthedocs.io>

## Mailing lists

<https://groups.google.com/group/briansupport>

<https://groups.google.com/group/brian-development>

## Publications

Goodman DF and Brette R (2009). The Brian simulator. *Frontiers Neurosci*

doi: [10.3389/neuro.01.026.2009](https://doi.org/10.3389/neuro.01.026.2009)

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](https://doi.org/10.3389/fninf.2014.00006)

## Related software packages

Brian2GeNN (code generation for GPUs via the GeNN simulator)

<http://brian2genn.readthedocs.io>

Brian2tools (tools for visualization and model export)

<http://brian2tools.readthedocs.io>

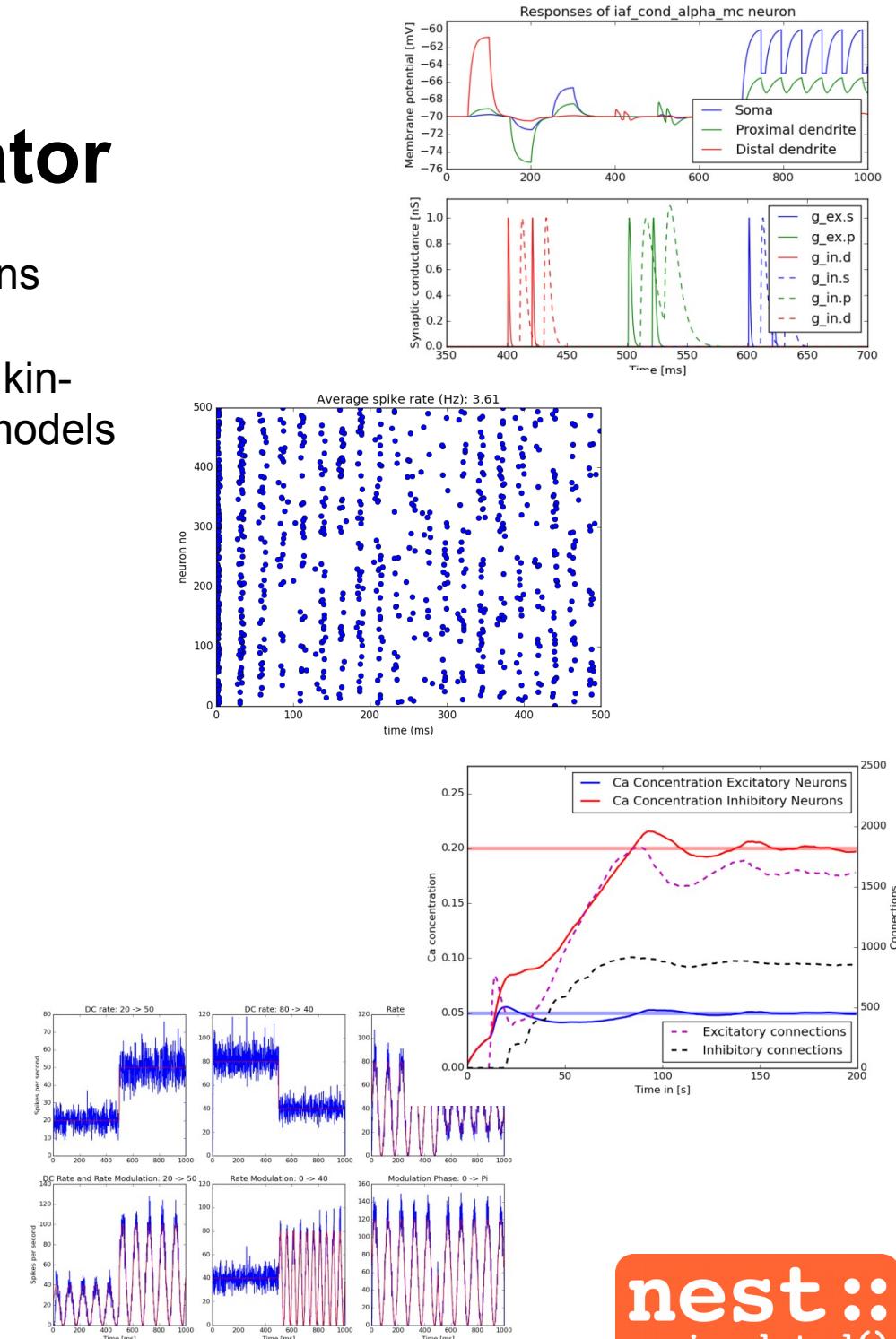
# NEST: The Neural Simulation Tool

<http://www.nest-simulator.org>

NEST is a simulator for spiking neural network models focussing on the dynamics, size and structure of neural systems rather than on the exact morphology of individual neurons. NEST is ideal for networks of spiking neurons of any size, from individual neurons to whole-brain models.

# NEST: A powerful simulator

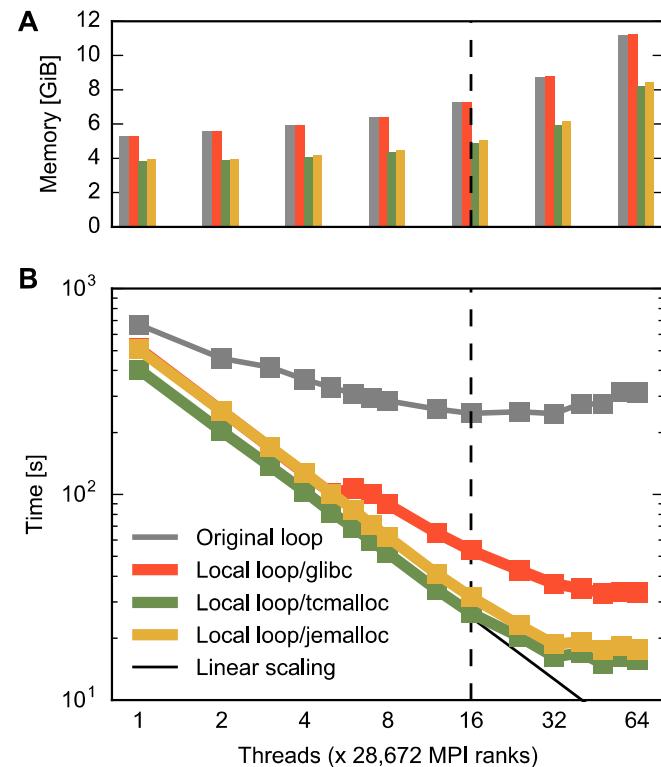
- Focused on networks of spiking point neurons
- Supports
  - Many neuron models including Hodgkin-Huxley style and few-compartment models
  - Synaptic plasticity including neuromodulatory signals
  - Structural plasticity
  - Gap junctions
  - Spatially structured networks
- Python interface
- Scales from laptops to supercomputers
- Active user and developer community
- Systematic quality control by continuous integration testing and code review
- Based on over 20 years of experience



# NEST Development

- Active developer community on Github
- Regular open developer video conferences
- Focus on improving performance and usability
- Some current projects
  - Support for rate models
  - NESTML: Automatic code generation for neuron models
  - NESTIO: Efficient data recording to binary file formats
  - Dry-run mode: Efficient performance analysis on supercomputers
  - Improved network construction speeds for highly threaded simulations
- Regular publications on NEST simulation technology

The screenshot shows the GitHub repository page for 'nest / nest-simulator'. The repository has 21 stars, 105 forks, and 111 issues. There are 24 open pull requests. The search bar shows 'Is:pr Is:open'. The pull requests list includes several items related to infrastructure and kernel improvements, with labels like Installation, Kernel, Infrastructure, and Maintenance.



Ippen et al (2017)

# More NEST Resources

## **Simulator homepage**

<http://www.nest-simulator.org>

## **Github repository**

<http://github.com/nest/nest-simulator>

## **NEST User mailing list**

[http://mail.nest-initiative.org/cgi-bin/mailman/listinfo/nest\\_user](http://mail.nest-initiative.org/cgi-bin/mailman/listinfo/nest_user)

## **NEST Initiative**

<http://www.nest-initiative.org>

## **Annual NEST User Workshop**

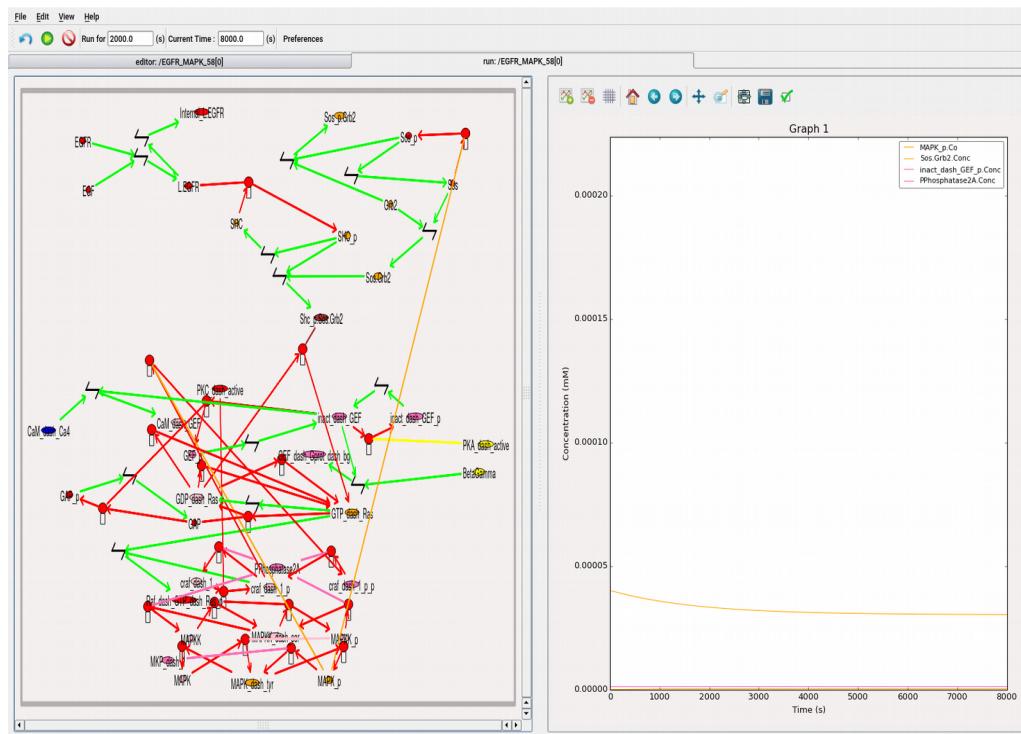
# MOOSE

## Multiscale Object Oriented Simulation Environment

<https://github.com/BhallaLab/moose>

MOOSE is designed to simulate neural systems ranging from biochemical signaling to complex models of single neurons, circuits, and large networks. MOOSE can operate at many levels of detail, from stochastic chemical computations, to multicompartment single-neuron models, to spiking neuron network models.

# Example: multiscale model

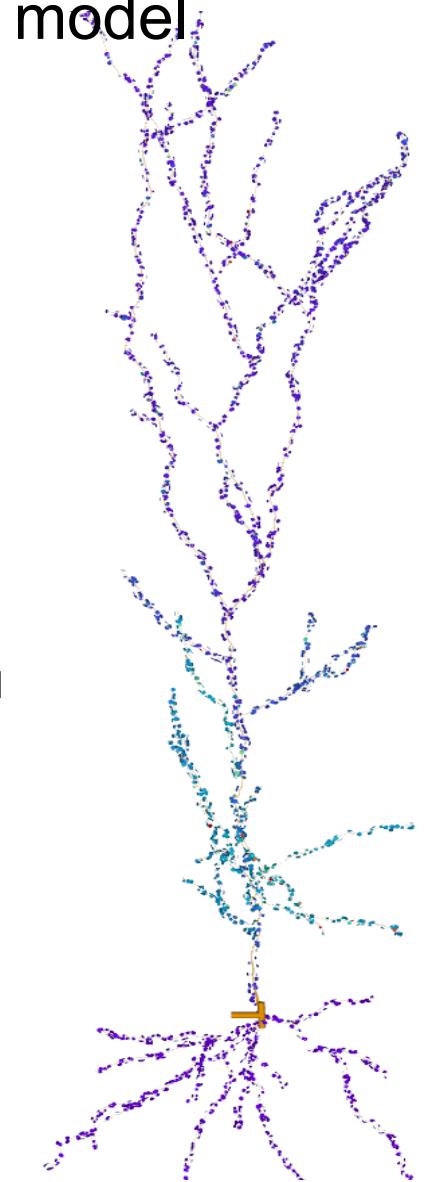


GUI for building and running chemical models.

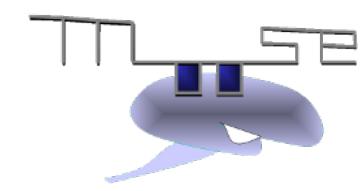
```
1 import moose
2 import rdesigneur as rd
3 rdes = rd.rdesigneur()
4     stimList = [['soma', '1', '.', 'inject', '(t>0.1 && t<0.2) * 2e-8']],
5     plotList = [['soma', '1', '.', 'Vm', 'Soma membrane potential']]
6 )
7 rdes.buildModel()
8 moose.reinit()
9 moose.start( 0.3 )
10
11 rdes.display()
```

Fully **Python scriptable**. **rdesigneur** interface to integrate chemical and electrical model.

- Morphology from Hippocampal CA1 pyramidal neuron
  - NeuroMorpho.org
- More than 3000 compartments, ion channels in each.
- Roughly 5000 spines with user defined distribution.
- Full reaction set in each spine and morpho compartment
- Background synaptic input at 0.1 Hz
- Patterned synaptic input on groups of spines → Ca influx
- Ca → signaling, signaling → channel modulation



**3d visualizer** for neural activity (Linux Only).



# Features and Plans

## Features

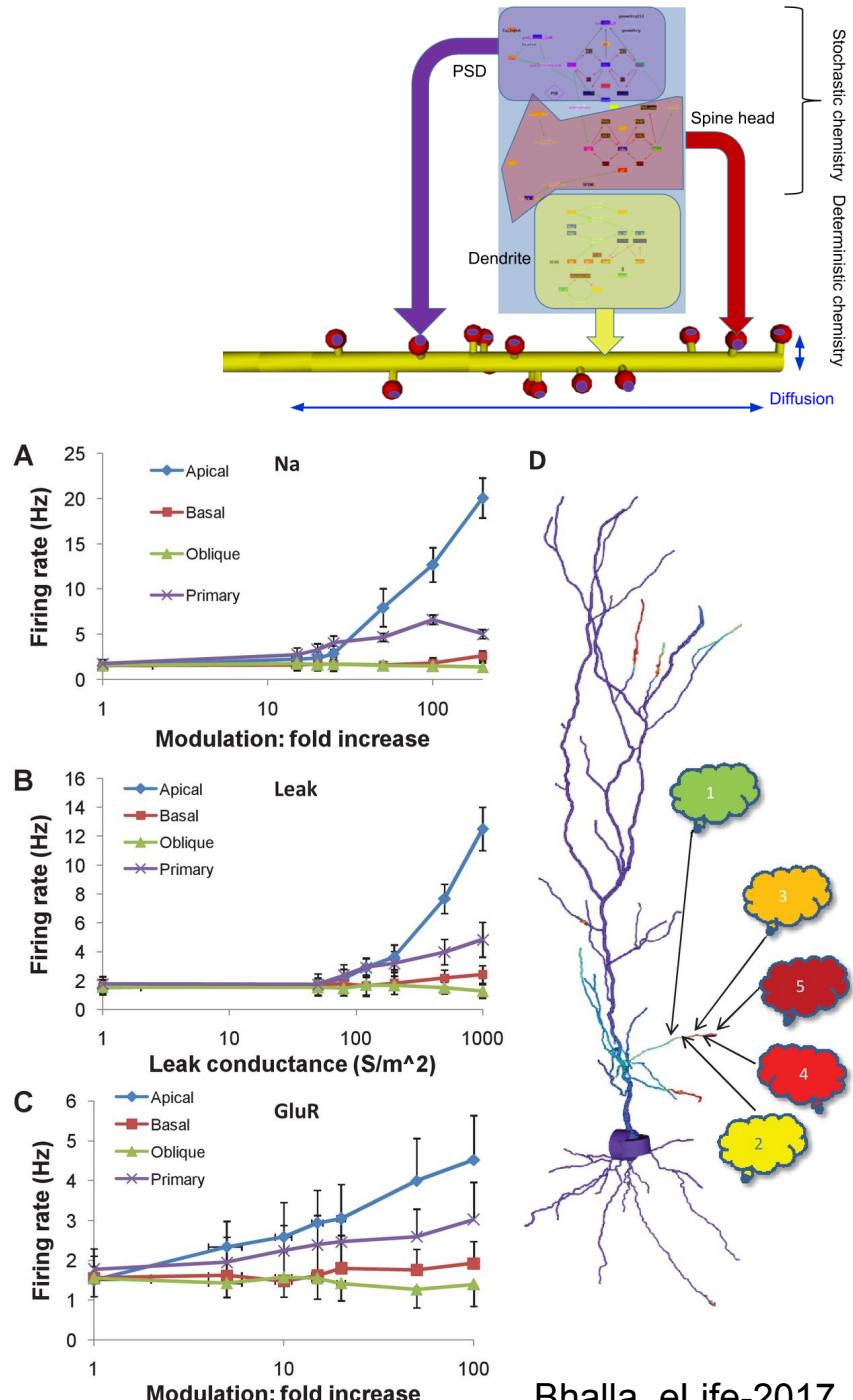
- Fast (written in C++) and easy to use (Python scriptable). **Rdesigneur** library to integrate chemical and electrical models easily.
- Chemical model formats support: SBML, NeuroML(v-1.8) and genesis format. Neuron morphology: SWC format.
- GUI for building chemical models.
- User [documentation](#) and many [examples](#)
- Linux ([via OBS](#)) and MacOSX (via Homebrew).

## In development

- Hsolve being ported to GPU (alpha). Credit: GSOC-2015,2017).
- Multithreaded chemical solvers Gsolve/Ksolve/Dsolve (beta). Speed improves roughly linearly with number of cores.
- Major upgrade of documentation.
- Activity-driven morphology change

## Future Plans

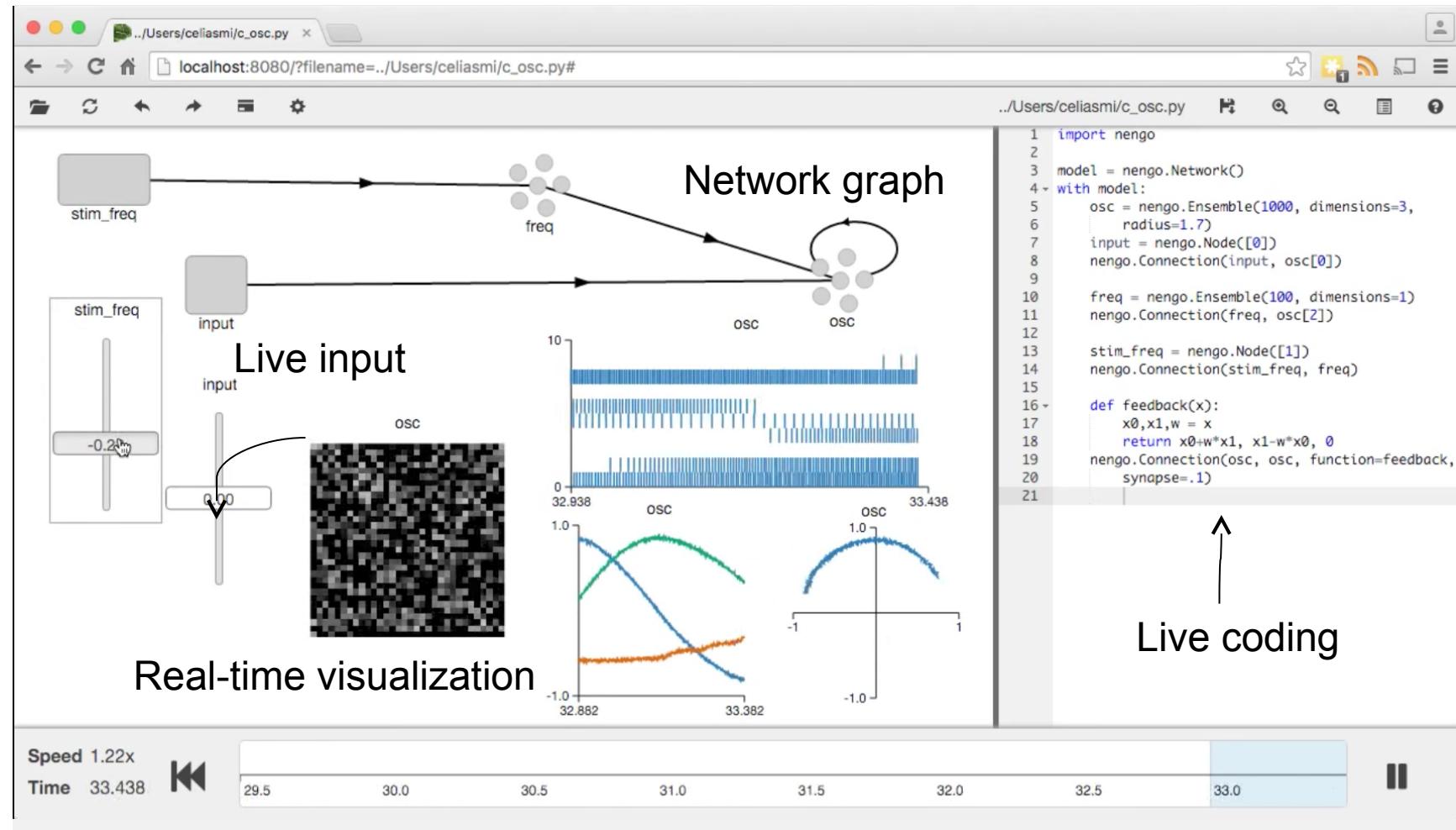
- Enhanced GPU and openMP support for various solvers.



# Nengo 2.0

<http://nengo.github.io>

Nengo is a graphical and scripting based software package for simulating large-scale spiking and non-spiking neural systems. It supports CPUs, GPUs (single and multi), MPI, and neuromorphic chips.



**Documentation** -- Usage and API documentation is at:

<https://pythonhosted.org/nengo/>

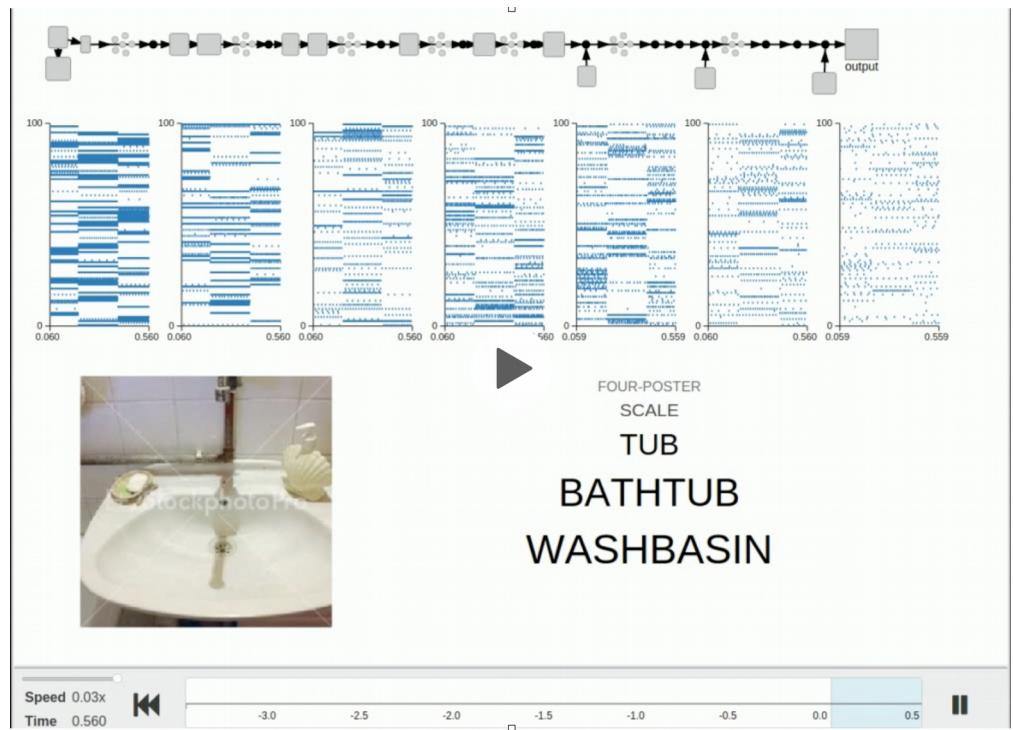
**Getting Help** -- Nengo forum at:

<https://forum.nengo.ai>

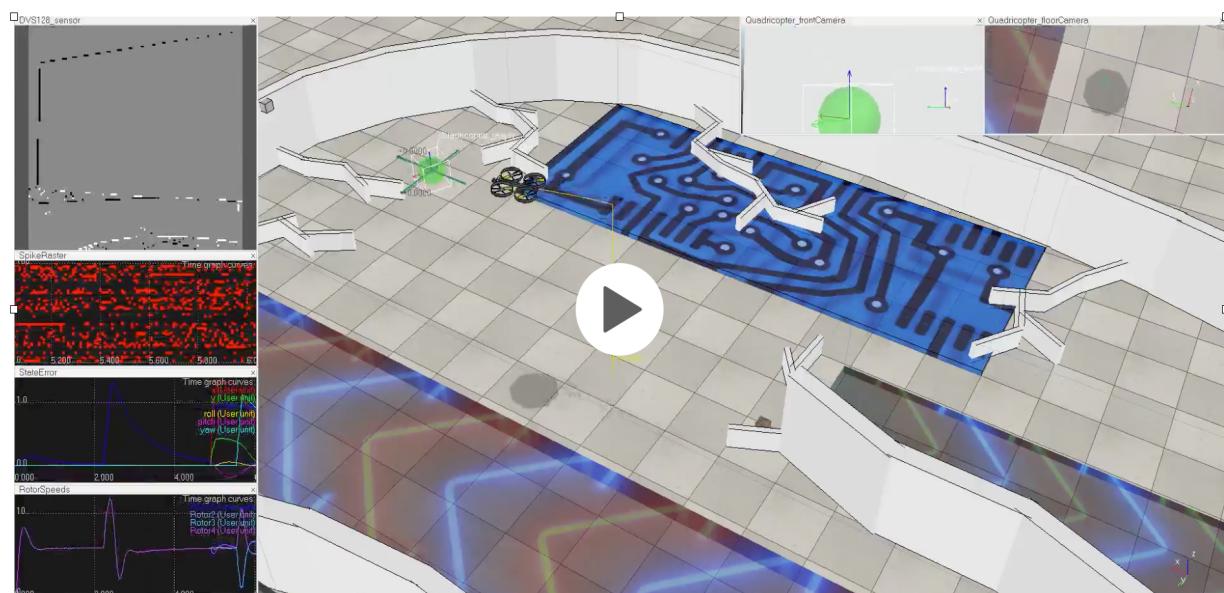
Screen cap from <https://youtu.be/UVeIPKnQAL4>

## **Application Highlights**

- Used to develop large-scale spiking deep networks
- Integration with robot simulators & platforms
- 6 DOF nonlinear adaptive control on neuromorphic hardware for 30x power savings
- Spun large-scale neuro-cognitive model published in *Science*



**Spiking ImageNet in Nengo**  
<https://youtu.be/7R5F4mNURGc>



**Nengo adaptive quadcopter control in VREP**  
<https://youtu.be/KBwBX7bzohA>

# More Nengo Resources

## In Development

- More complete NEURON integration
- Fully featured Semantic Pointer Architecture (SPA) library for cognitive modeling
- Additional tools for TensorFlow integration (Nengo DL)
- Additional backends for FPGAs, neuromorphic ASICs, etc.

## Online Tutorials & Examples (in addition to documentation)

- Covering NEF and SPA methods in Nengo GUI

[https://github.com/nengo/nengo\\_gui/tree/master/nengo\\_gui/examples/tutorial](https://github.com/nengo/nengo_gui/tree/master/nengo_gui/examples/tutorial)

- >40 Jupyter notebook examples covering core Nengo usage

<https://pythonhosted.org/nengo/examples.html>

- To accompany the book How to Build a Brain (2013)

[https://github.com/nengo/nengo\\_gui/tree/master/nengo\\_gui/examples/hbb\\_tutorials](https://github.com/nengo/nengo_gui/tree/master/nengo_gui/examples/hbb_tutorials)

## Additional resources

- Annual Nengo Summer School

<http://nengo.ca/summerschool>

- Information for current or prospective developers can be found at

<https://nengo.github.io/contributing.html>