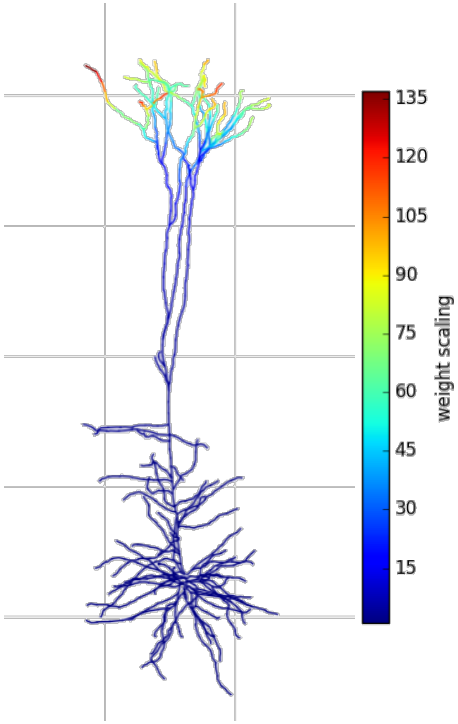
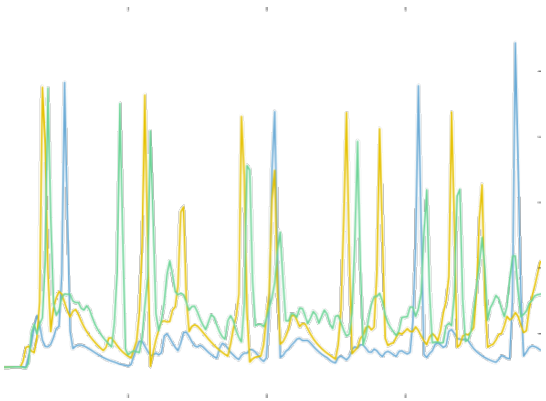
NetPyNE

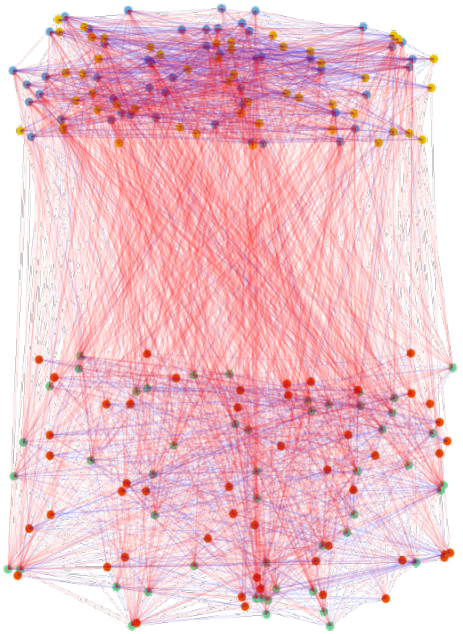
(**Net**works in **Py**thon and **NE**URON)

A Python package to facilitate the development, simulation and analysis of biological neuronal networks in NEURON



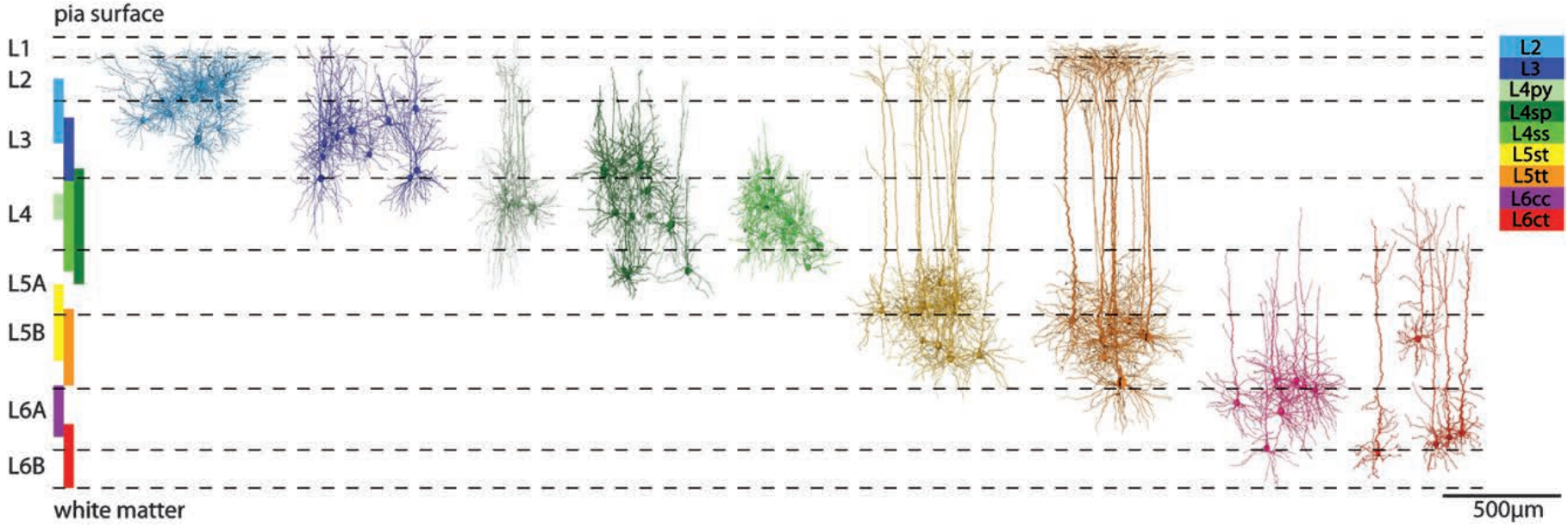
www.netpyne.org





Motivation

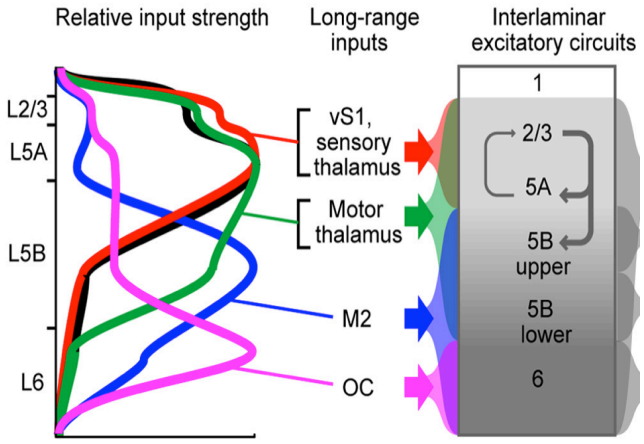
- Facilitate incorporation of experimental data at multiple scales



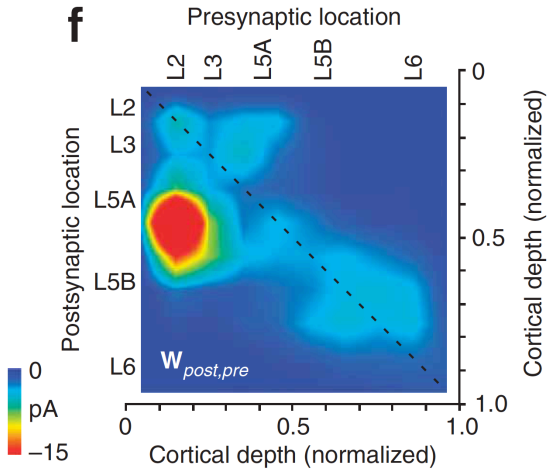
Motivation

- Facilitate incorporation of experimental data at multiple scales

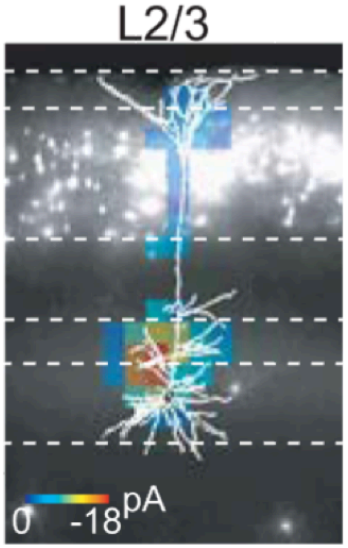
Long-range inputs



Local microcircuits



Dendritic inputs

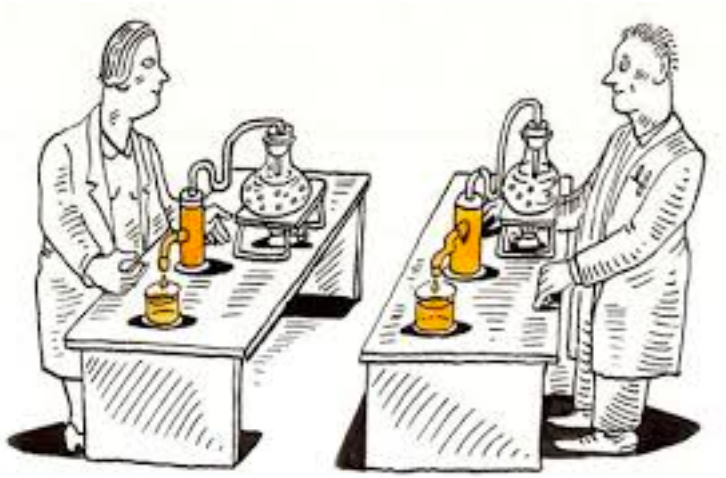


Motivation

- Separate model parameters from implementation
- Standardize format – easy to read, interpret, edit, share etc

```
popParams['EXC_L2'] = {
 'cellType': 'PYR',
 'yRange':    [100, 400],
 'numCells':  50}
```

```
for cellParams in range(pop['numCells']):
      cell = sim.Cell(cellParams)
      cell.tags['y'] = numpy.random(100,400)
      cell.tags['cellType'] = 'PYR'
```

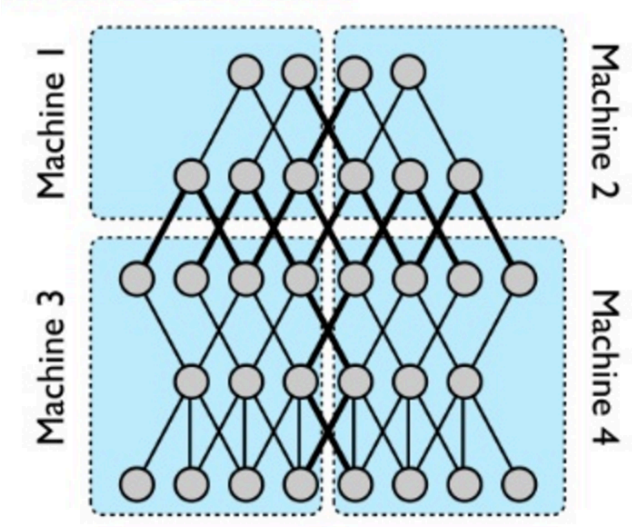


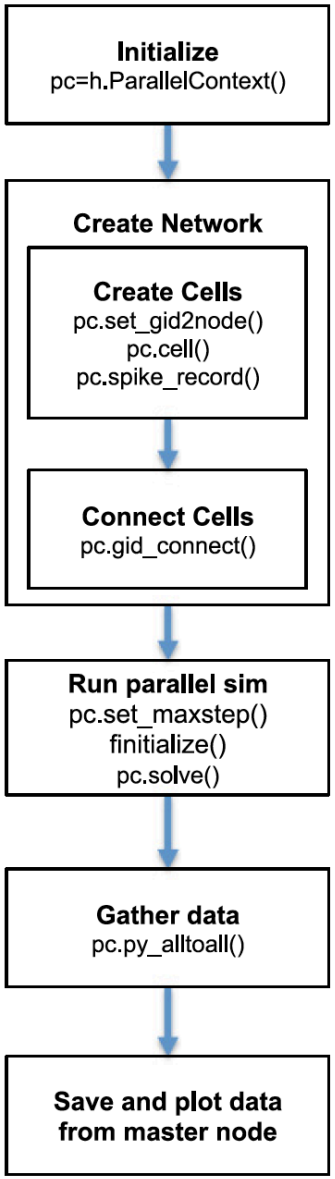
Replicate: get same thing to run again

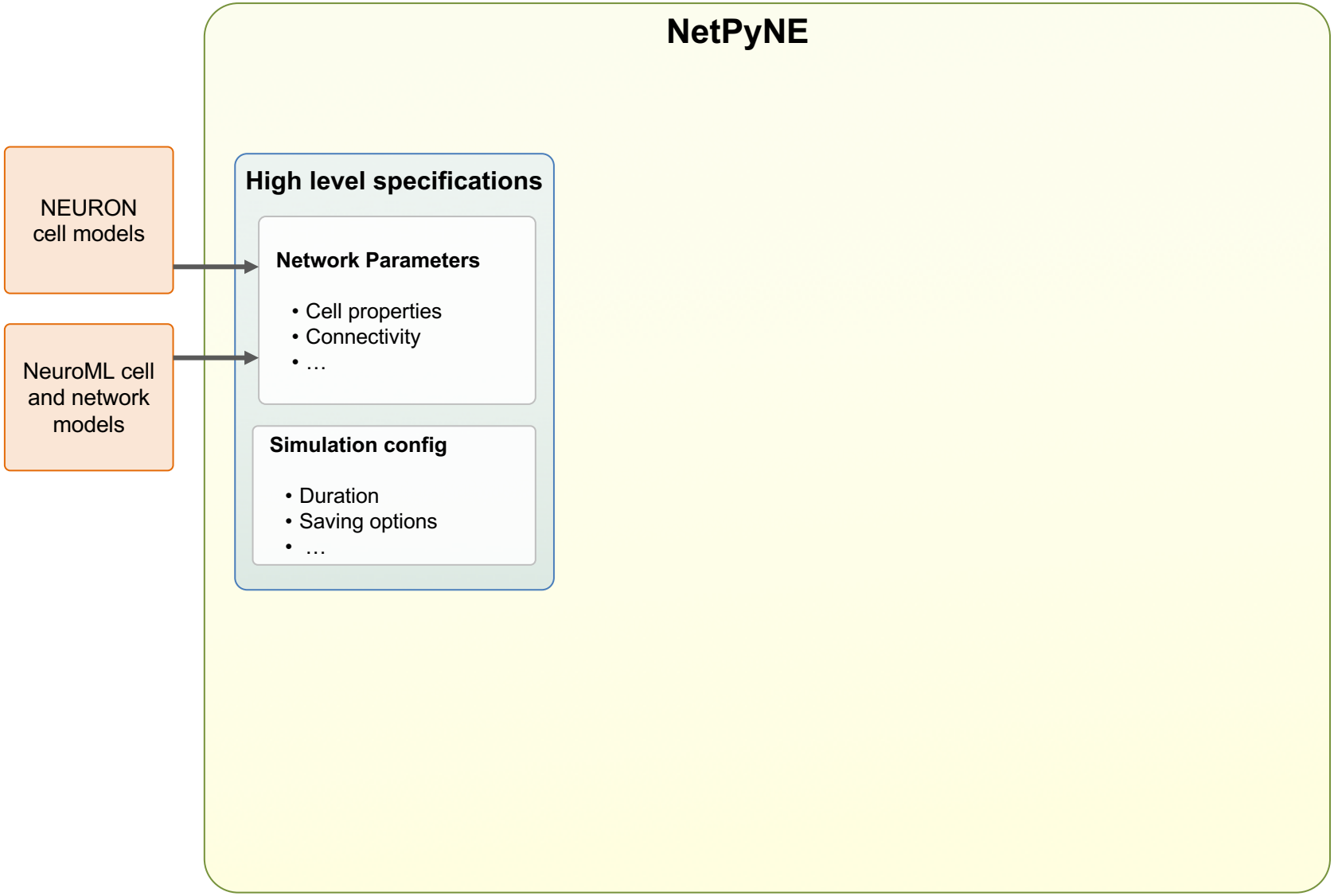
Reproduce: make it youself

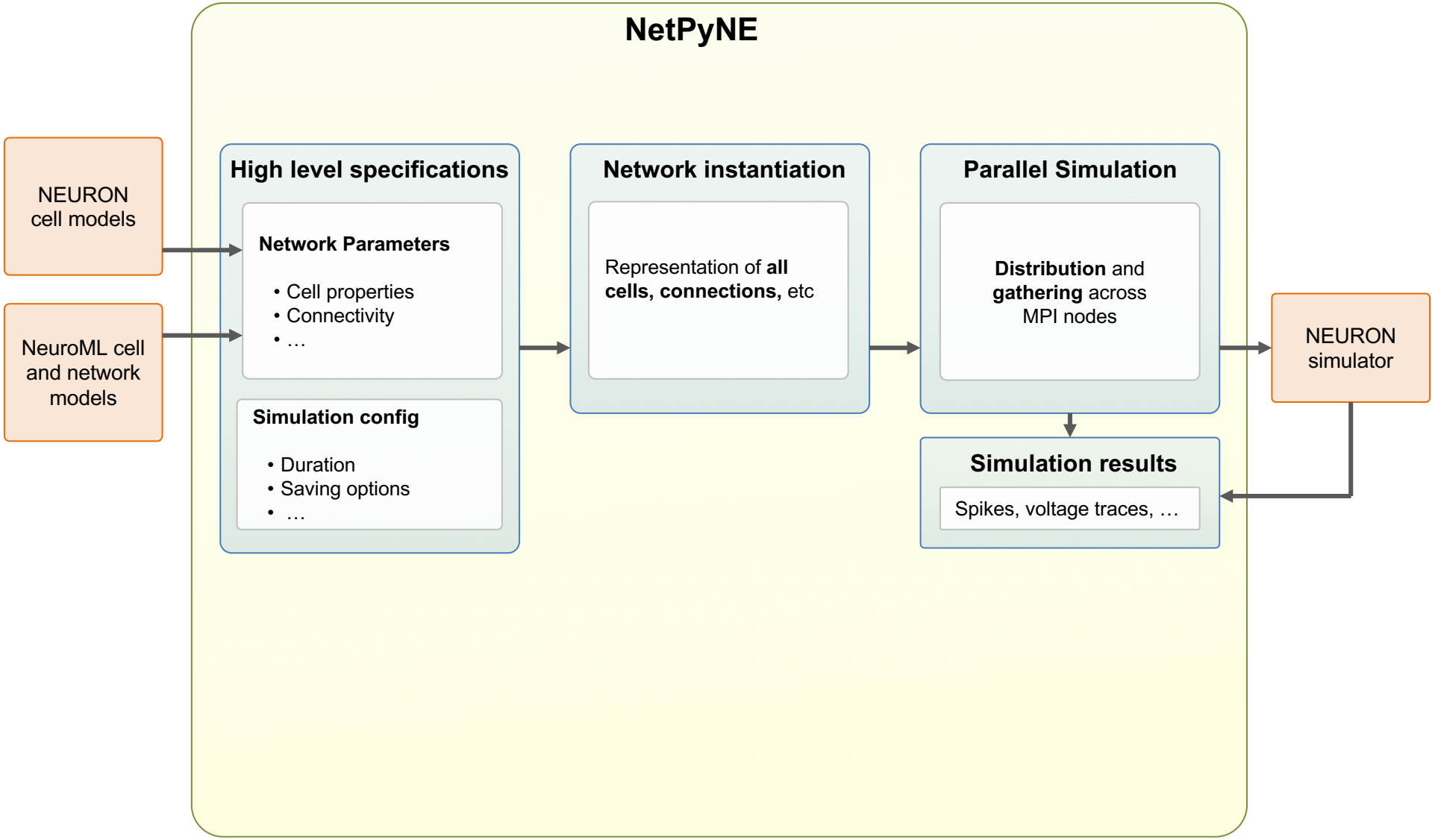
Motivation

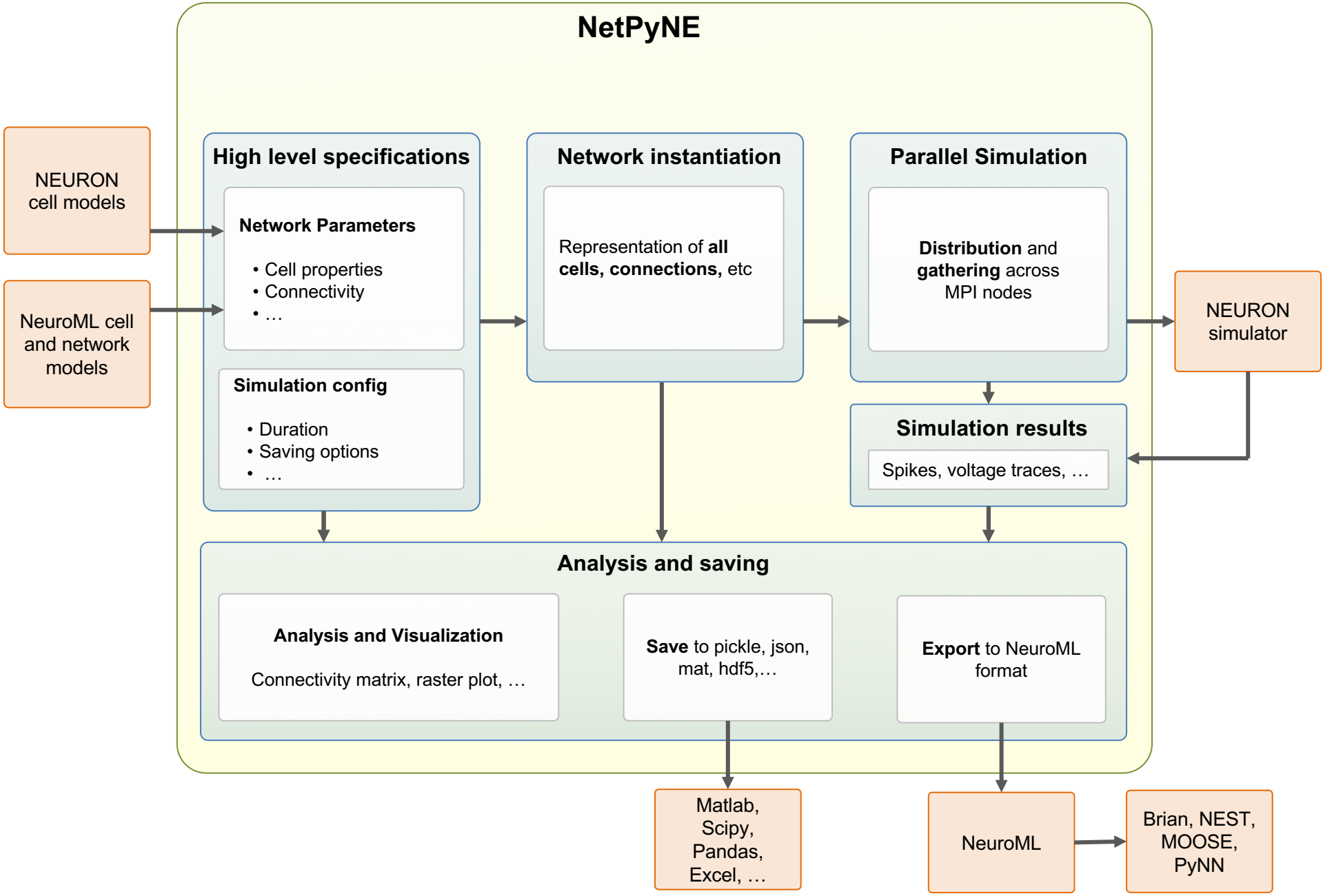
- Facilitate model parallelization (HPCs)
- Batch parameter exploration/optimization

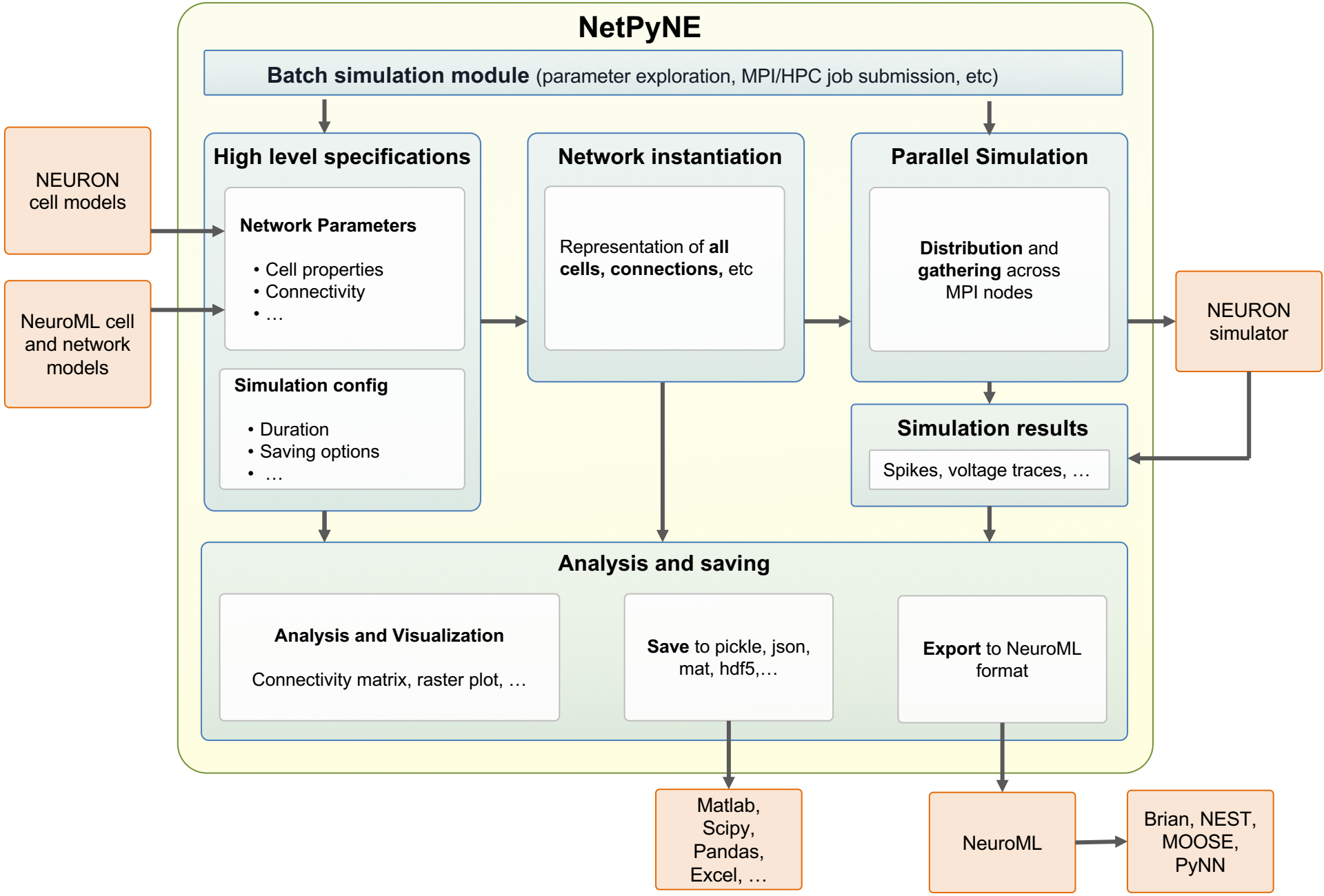





NetPyNE
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …


NetPyNE
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …
Network instantiation
Representation of all cells, connections, etc
Parallel Simulation
Distribution and gathering across MPI nodes
Simulation results
Spikes, voltage traces, …
NEURON simulator


NetPyNE
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …
Network instantiation
Representation of all cells, connections, etc
Parallel Simulation
Distribution and gathering across MPI nodes
NEURON simulator
Simulation results
Spikes, voltage traces, …
Analysis and saving
Analysis and Visualization
Connectivity matrix, raster plot, …
Save to pickle, json, mat, hdf5,…
Export to NeuroML format
Matlab, Scipy, Pandas, Excel, …
NeuroML
Brian, NEST, MOOSE, PyNN


NetPyNE
Batch simulation module (parameter exploration, MPI/HPC job submission, etc)
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …
Network instantiation
Representation of all cells, connections, etc
Parallel Simulation
Distribution and gathering across MPI nodes
NEURON simulator
Simulation results
Spikes, voltage traces, …
Analysis and saving
Analysis and Visualization
Connectivity matrix, raster plot, …
Save to pickle, json, mat, hdf5,…
Export to NeuroML format
Matlab, Scipy, Pandas, Excel, …
NeuroML
Brian, NEST, MOOSE, PyNN

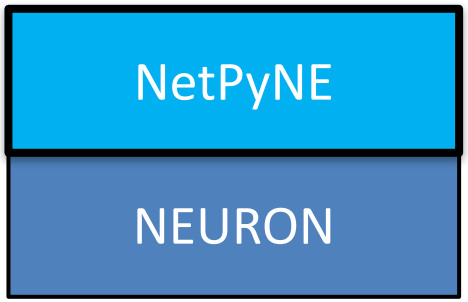
High level specifications

NEURON

```
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

pre1Con = h.NetCon(preCell1.soma(0.5)._ref_v,
                   postSyn1,
                   sec=preCell1.soma)
pre1Con.delay = 1
pre1Con.weight[0] = 0.001
pre1Con.threshold = 0
```

High level specifications

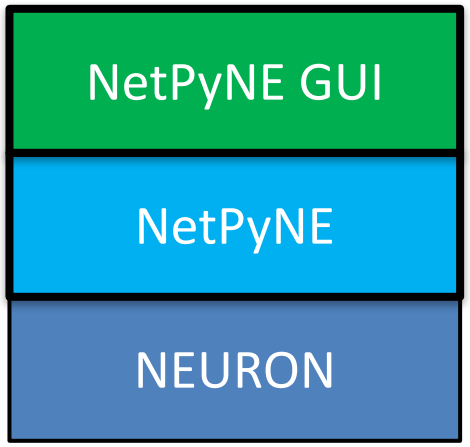


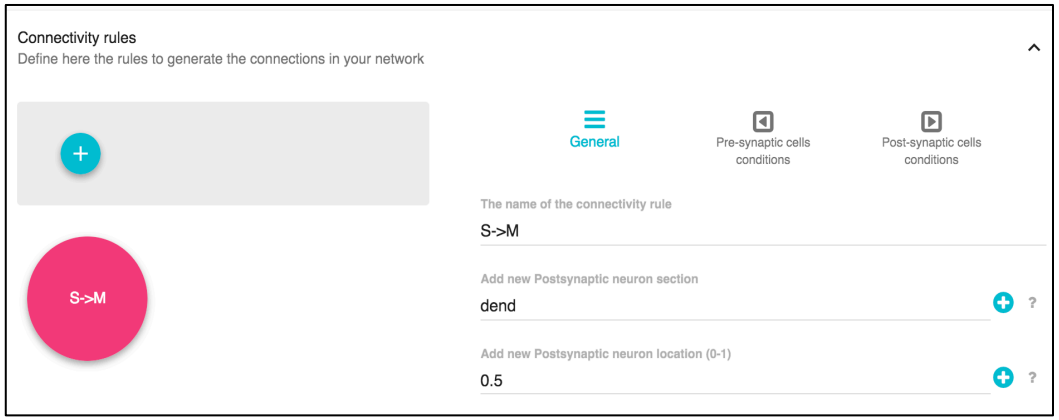

```
## Cell connectivity rules
netParams.connParams['S->M'] = {
        'preConds': {'pop': 'S'},
        'postConds': {'pop': 'M'},
        'probability': 0.5,
        'weight': 0.01,
        'delay': 5,
        'synMech': 'exc'}
```

```
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

pre1Con = h.NetCon(preCell1.soma(0.5)._ref_v,
                   postSyn1,
                   sec=preCell1.soma)
pre1Con.delay = 1
pre1Con.weight[0] = 0.001
pre1Con.threshold = 0
```

High level specifications





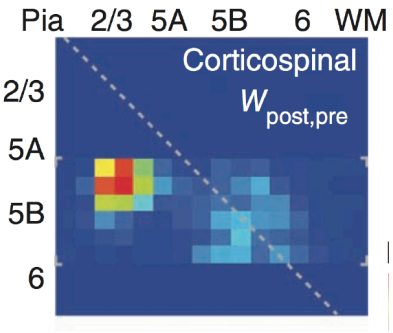

```
## Cell connectivity rules
netParams.connParams['S->M'] = {
        'preConds': {'pop': 'S'},
        'postConds': {'pop': 'M'},
        'probability': 0.5,
        'weight': 0.01,
        'delay': 5,
        'synMech': 'exc'}
```

```
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

pre1Con = h.NetCon(preCell1.soma(0.5)._ref_v,
                   postSyn1,
                   sec=preCell1.soma)
pre1Con.delay = 1
pre1Con.weight[0] = 0.001
pre1Con.threshold = 0
```

High level specifications

- Specifications are provided in **a standardized, declarative** Python format (JSON-like, lists and dicts).
- Clear **separation** of parameters from implementation code.
- Error **checking** and **suggestions** to facilitate model definition.

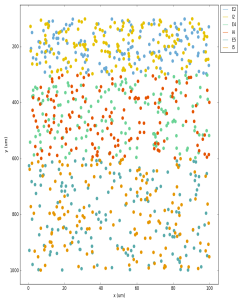


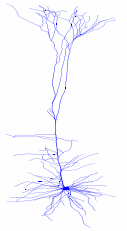

```
connParams['bin3->CSP'] = {
 'preConds':     {'y': [100, 150]},
 'postConds':    {'pop': 'CSP'},
 'probability': 0.15,
 'weight':       0.4,
 'delay':        5,
 'synMech':      'AMPA'}
```

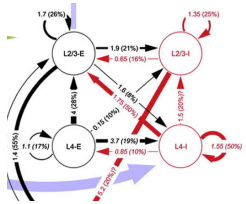
NetPyNE facilitates building models based on experimental data

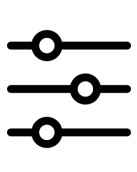
High level specifications

- User can define:
  - ***Populations**:* cell type, number of neurons or density, spatial extent, ...
  - ***Cell properties:*** Morphology, biophysics, implementation, …
  - ***Synaptic mechanisms:*** Time constants, reversal potential, implementation, …
  - ***Stimulation:*** Spike generators, current clamps, spatiotemporal properties, …
  - ***Connectivity rules:*** conditions of pre- and post-synaptic cells, different functions, …
  - ***Simulation configuration:*** duration, saving and analysis, graphical output, ...



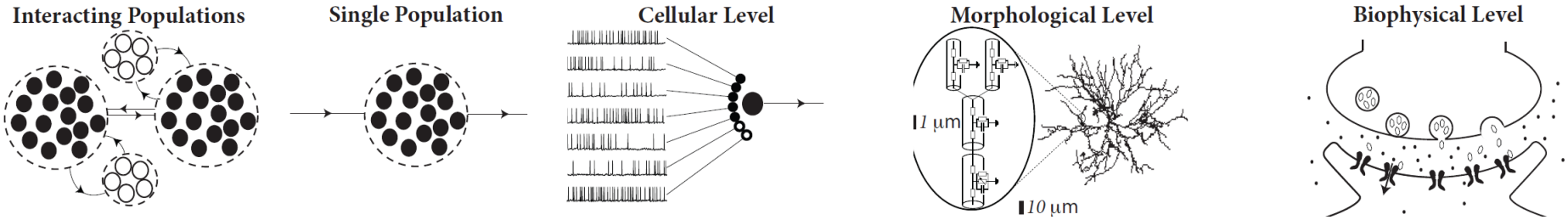


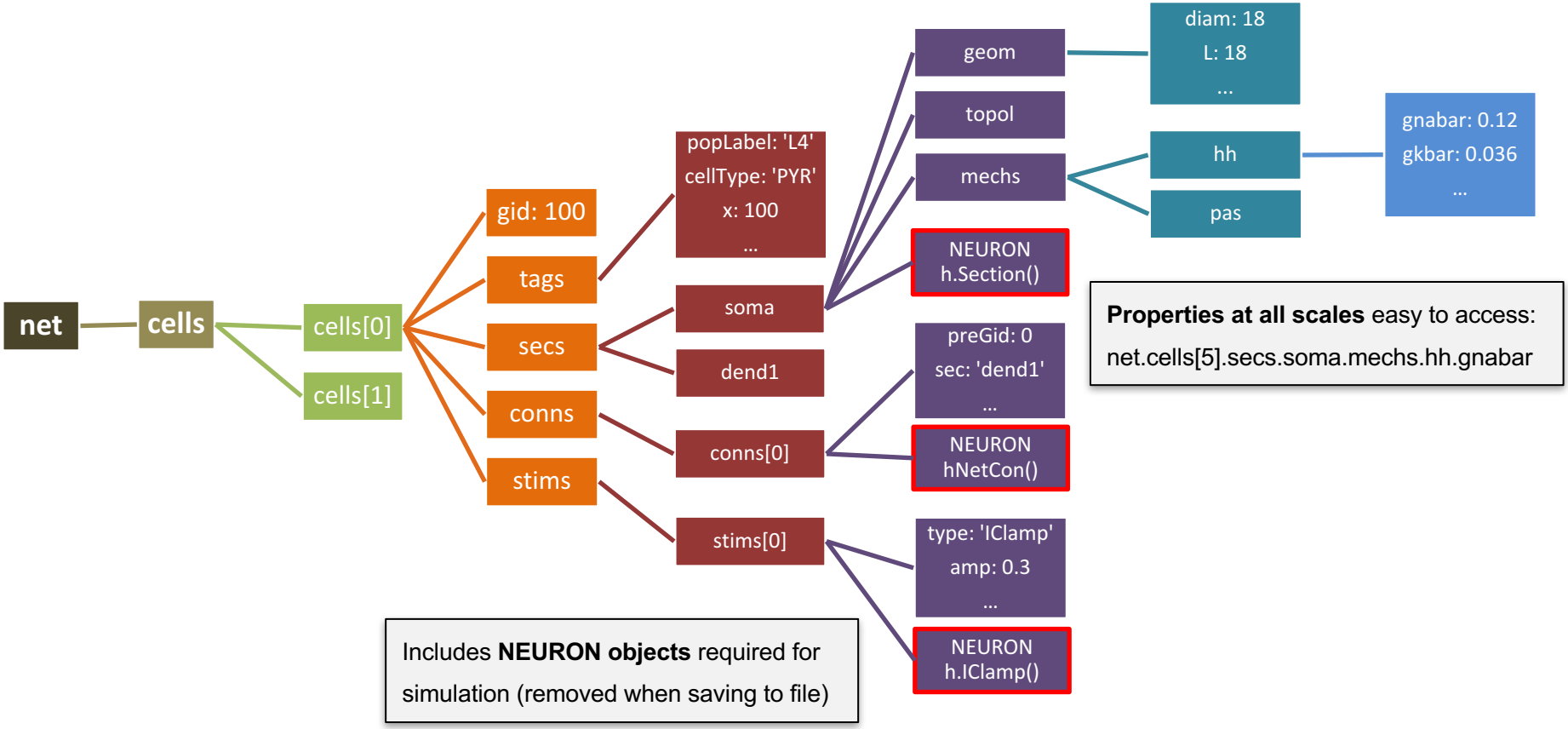




Network instantiation

- Network instance as **standardized hierarchical** Python strucutre (JSON-like, lists and dicts)

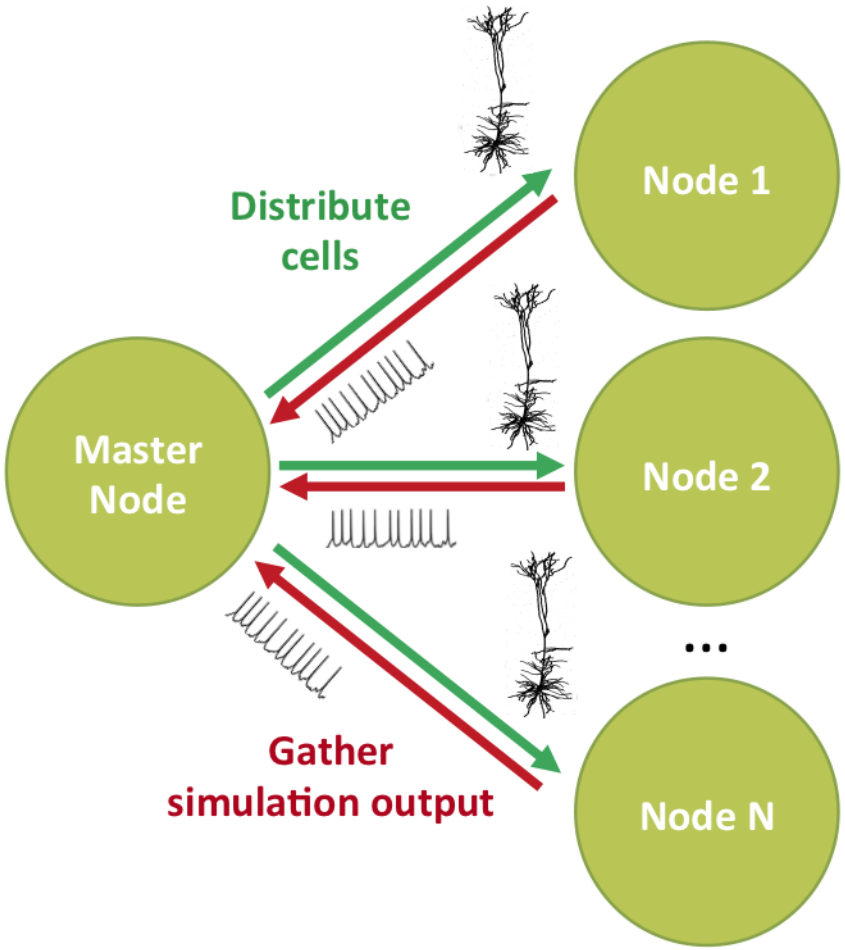




Parallel Simulation

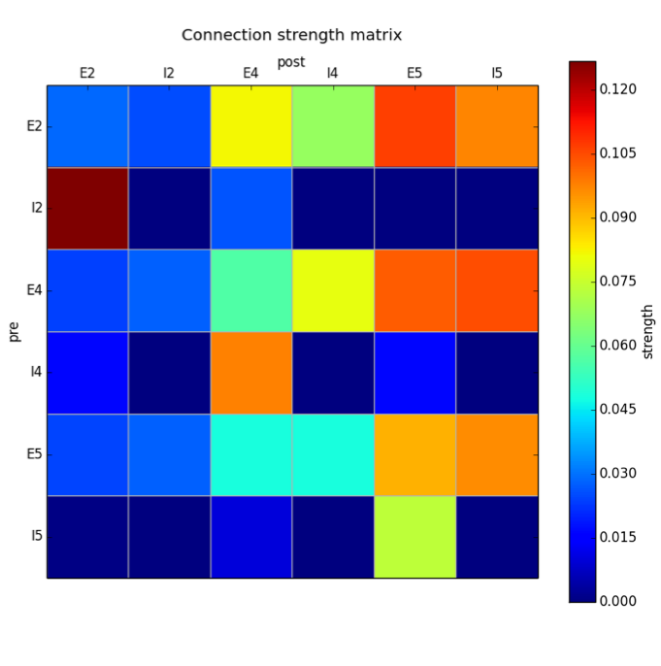
- Set up for MPI **parallel simulation** across multiple nodes (via NEURON simulator).

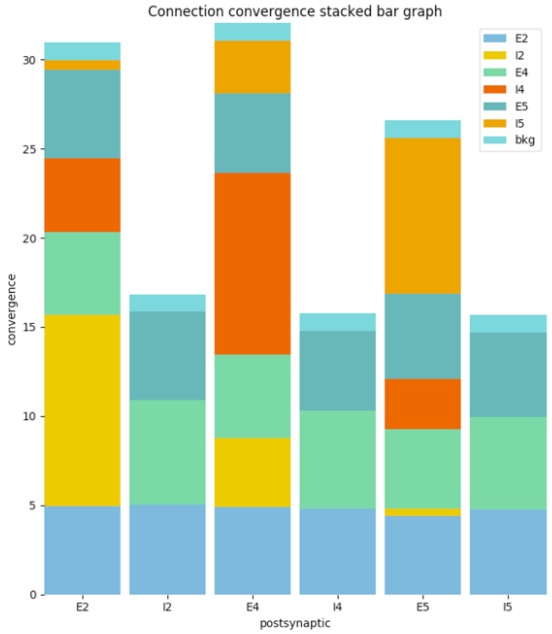
- Takes care of balanced **distribution** of cells and **gathering** of simulation output from nodes.

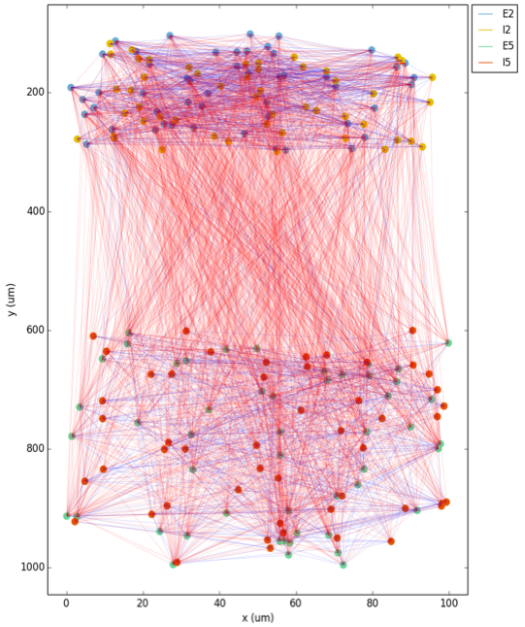


Analysis

- **Connectivity plots** at cell or population level (weights, num connections, probability,...)



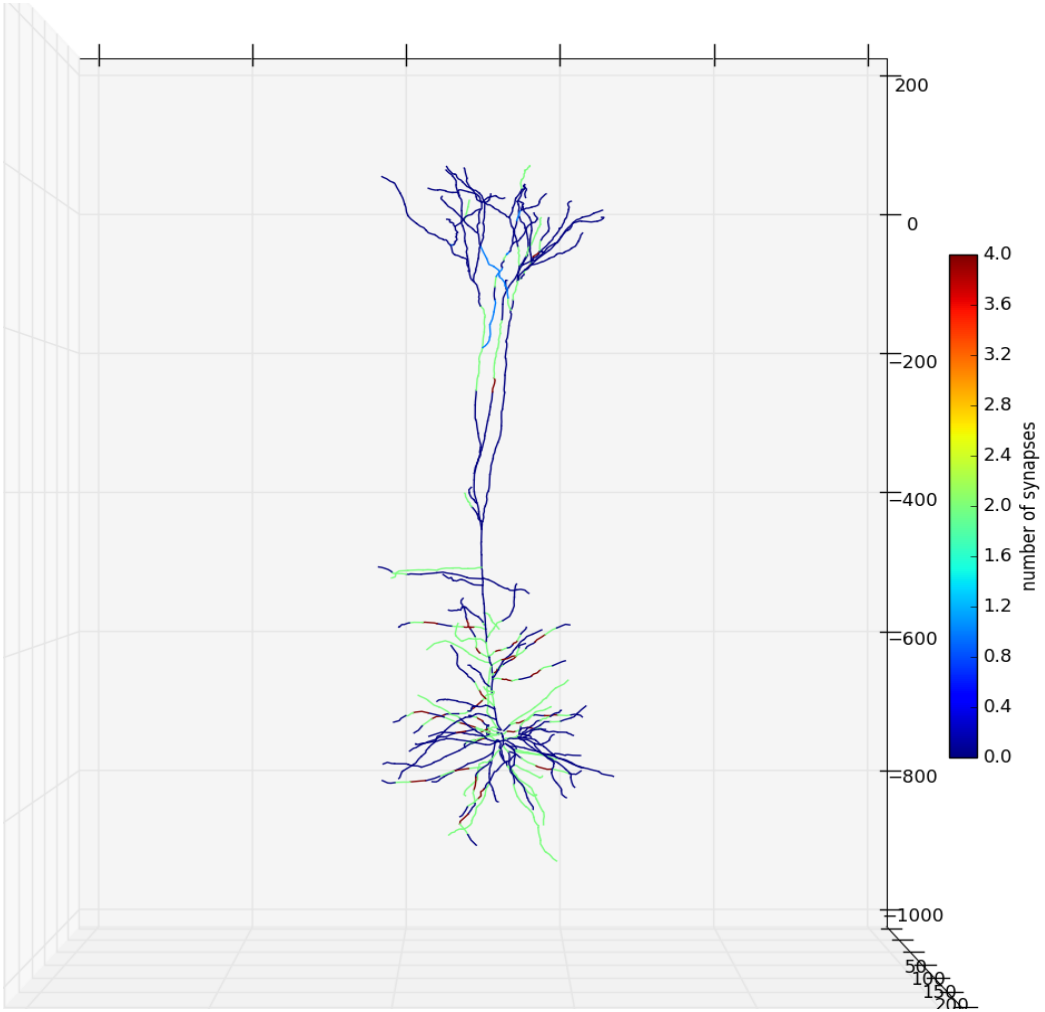





```
plotConn(include = ['allCells'], feature='strength',
groupBy='pop', figSize=(9,9), showFig=True)
```

Analysis

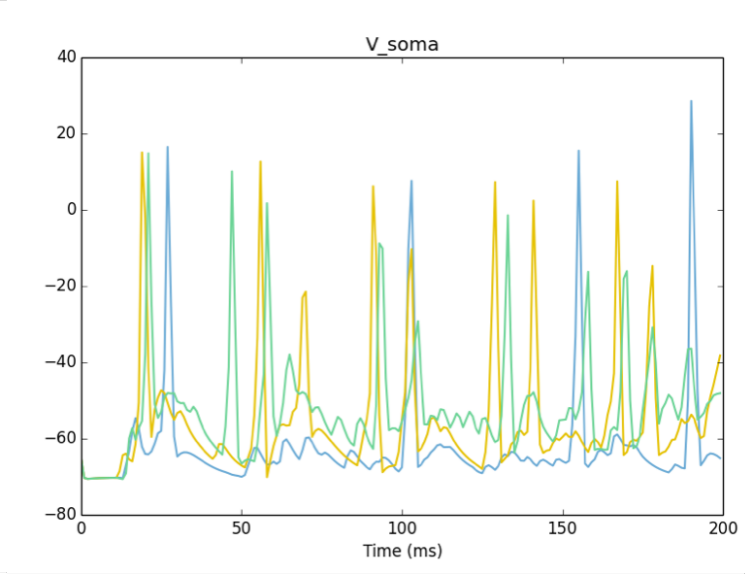
- 3D **cell shape plot**
- Option to include **color-coded variables** (eg, num of synapses)

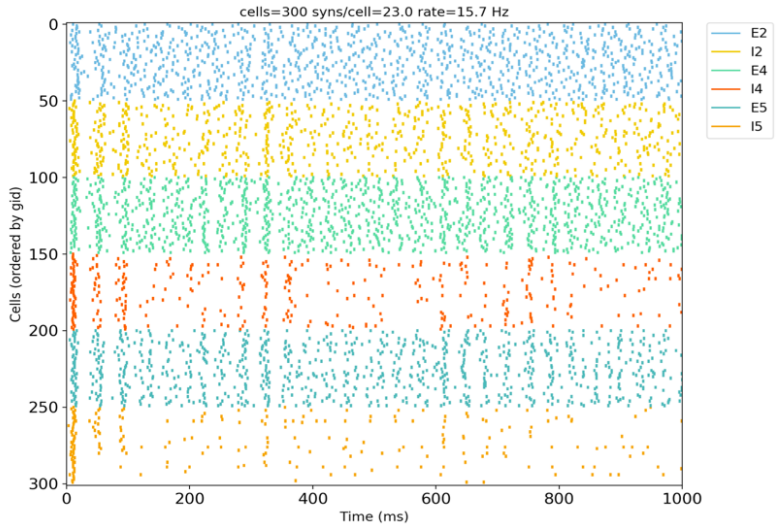


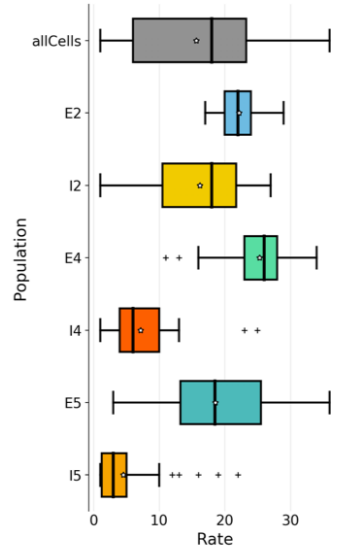
`plotShape(…)`

Analysis

- Simulation output
  - Intrinsic cell variables (voltages, currents, conductance) **trace plots**
  - **Raster plot** of any subset of cells
  - **Spike histogram** of populations or subsets of cells
  - Population **statistics**

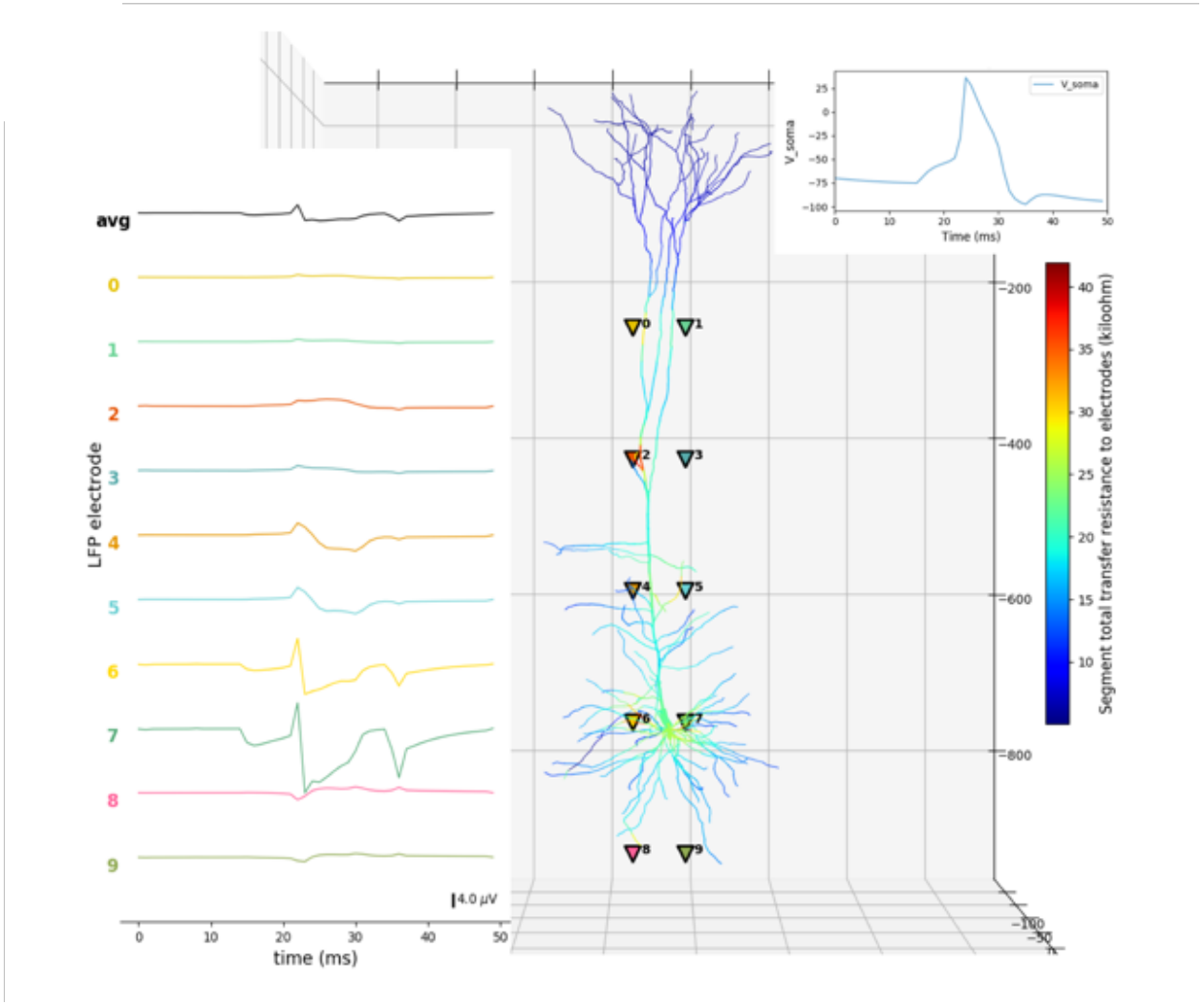






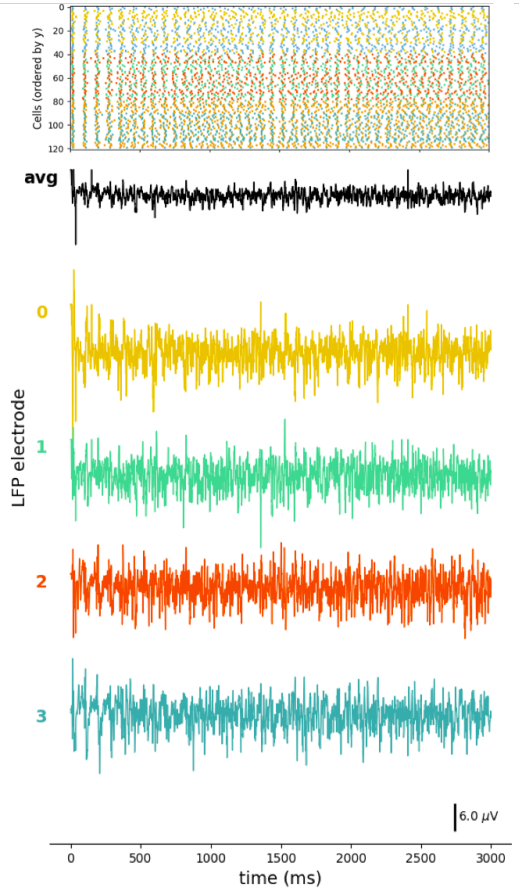
LFP

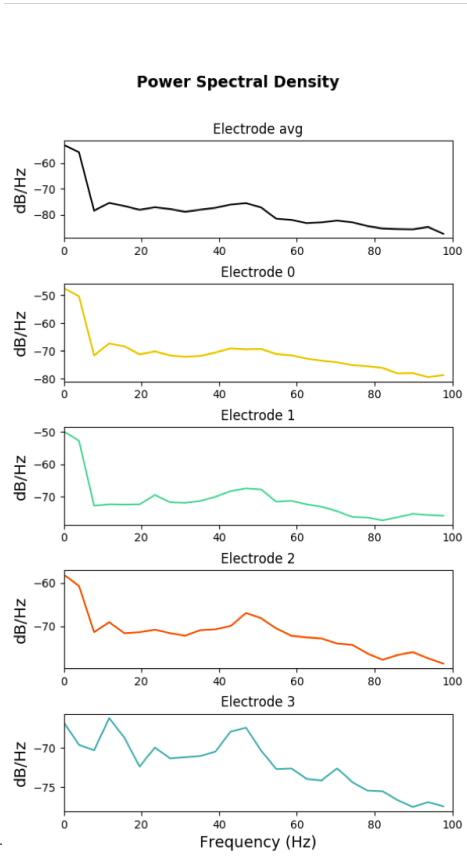
- **LFP** time-series, PSD, spectrogram and electrode locations (single cell)

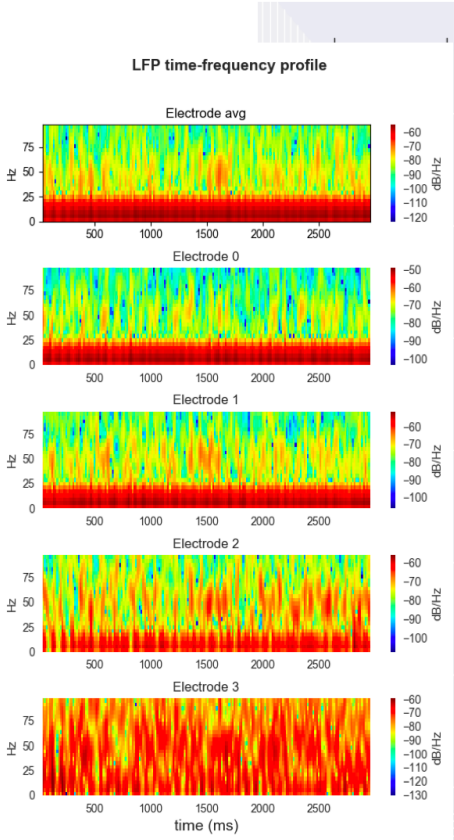


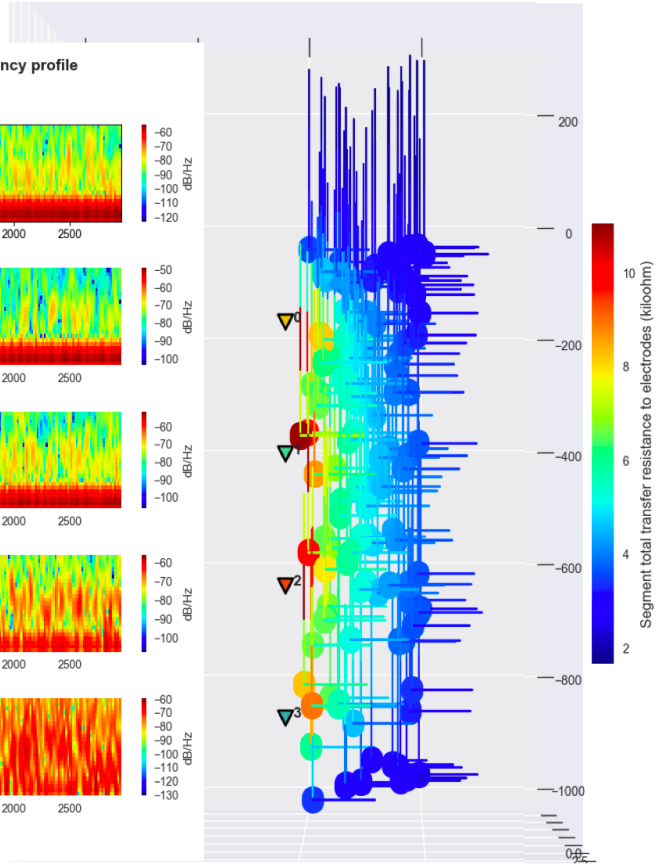
Analysis

- **LFP** time-series, PSD, spectrogram and electrode locations (network)





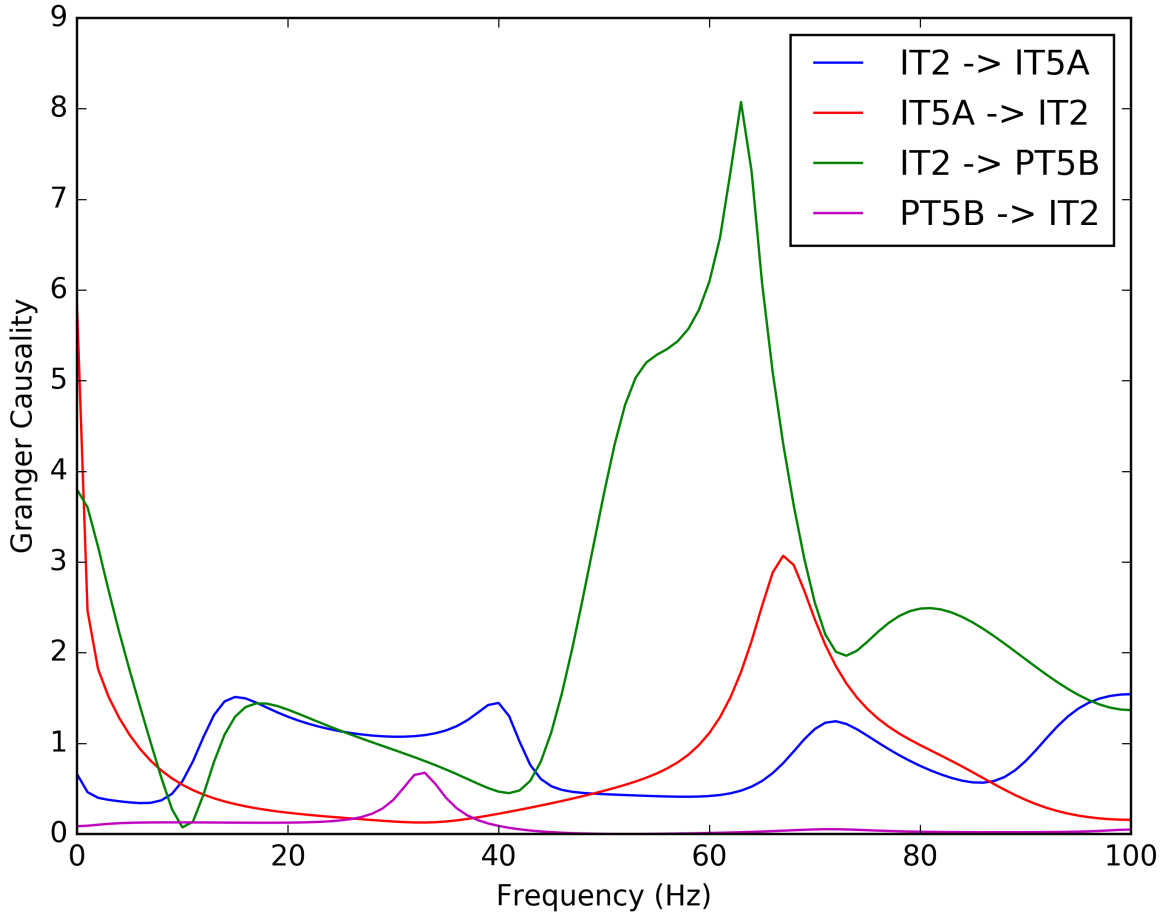




`plotLFP(...)`

Analysis

- Spectral **Granger causality**
- Normalized **transfer entropy**



`plotGranger(…)`

Batch Parallel Simulations

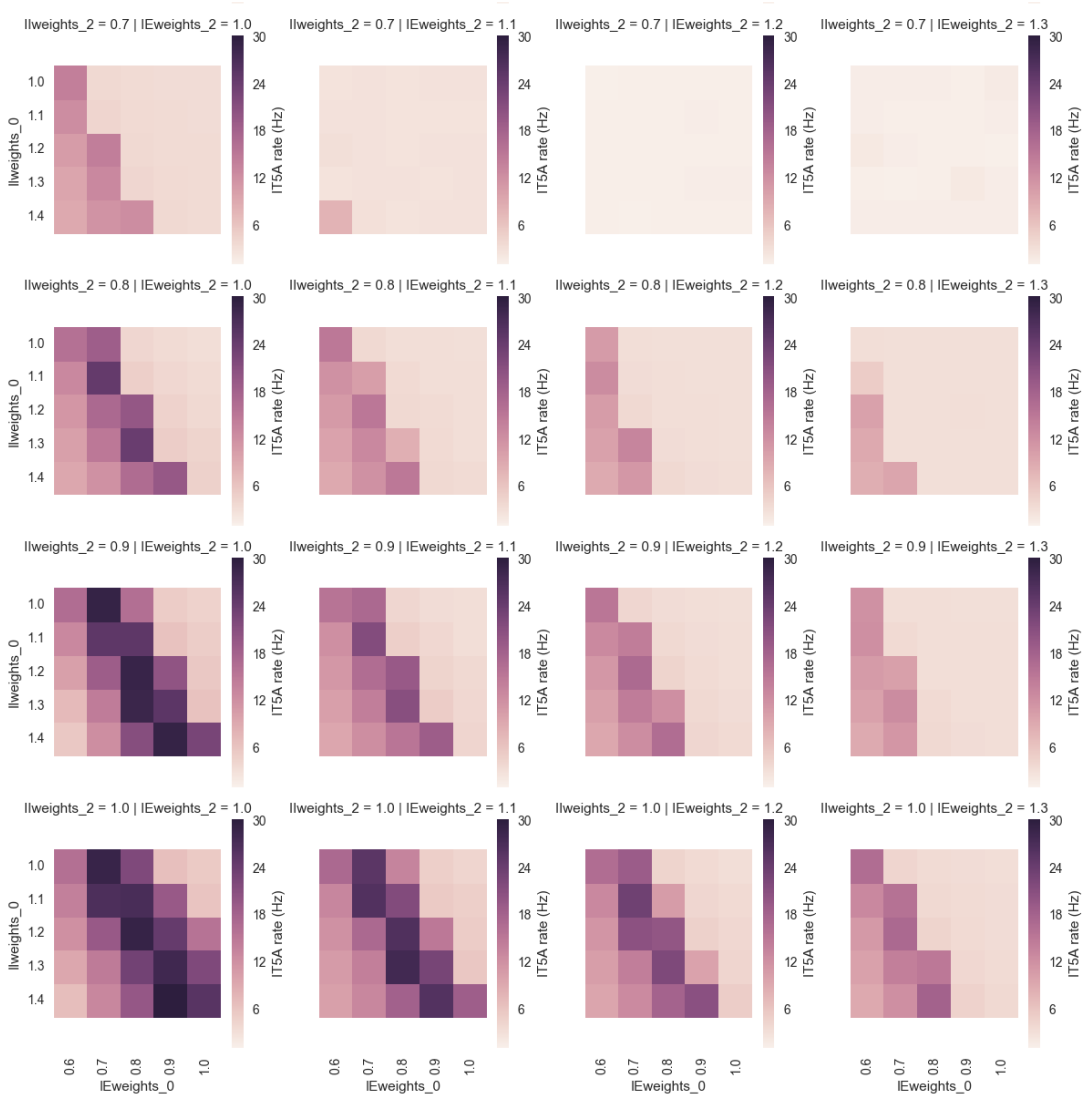
- **Easy specification** of parameters and range of values to explore in batch simulations.

- **Pre-defined, configurable** setups to automatically **submit jobs** in multicore machines (Bulletin board) or supercomputers (SLURM or PBS Torque)

SDSC SAN DIEGO SUPERCOMPUTER CENTER

Batch Simulation Analysis

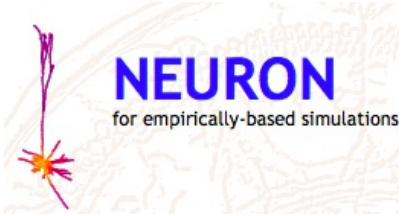
- **Analysis and visualization** of multidimensional batch simulation results.



Data saving and exporting

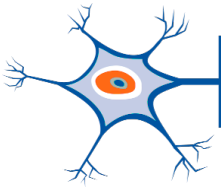
- **Save and load** high-level specifications, network instance, simulation config and/or simulation results.
- **Multiple formats** supported: pickle, Matlab, JSON, CSV, HDF5
- **Export/import** network instance to/from **NeuroML**, the standard format for neural models.

Data saving and exporting



netPyne

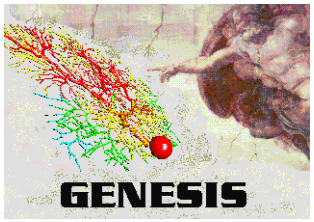
Import/export to standard format



Import/export to other simulators

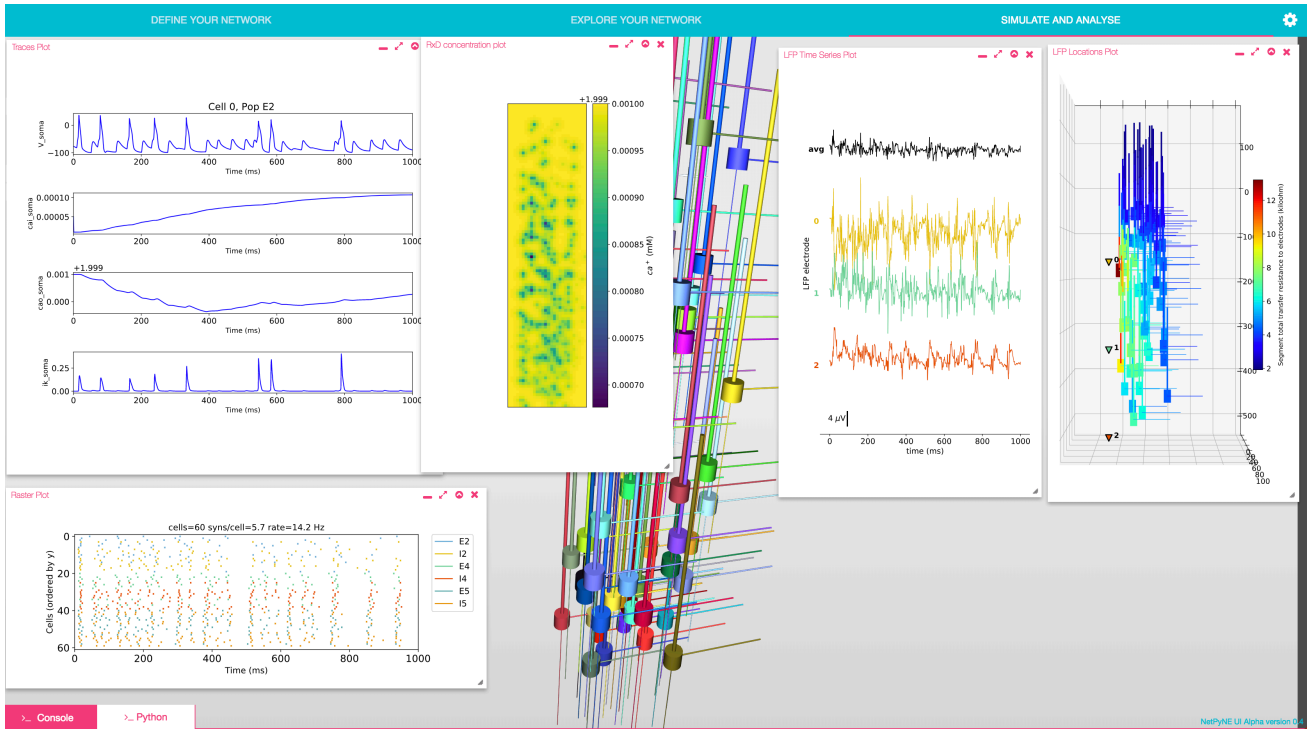
nest ::

PyNN

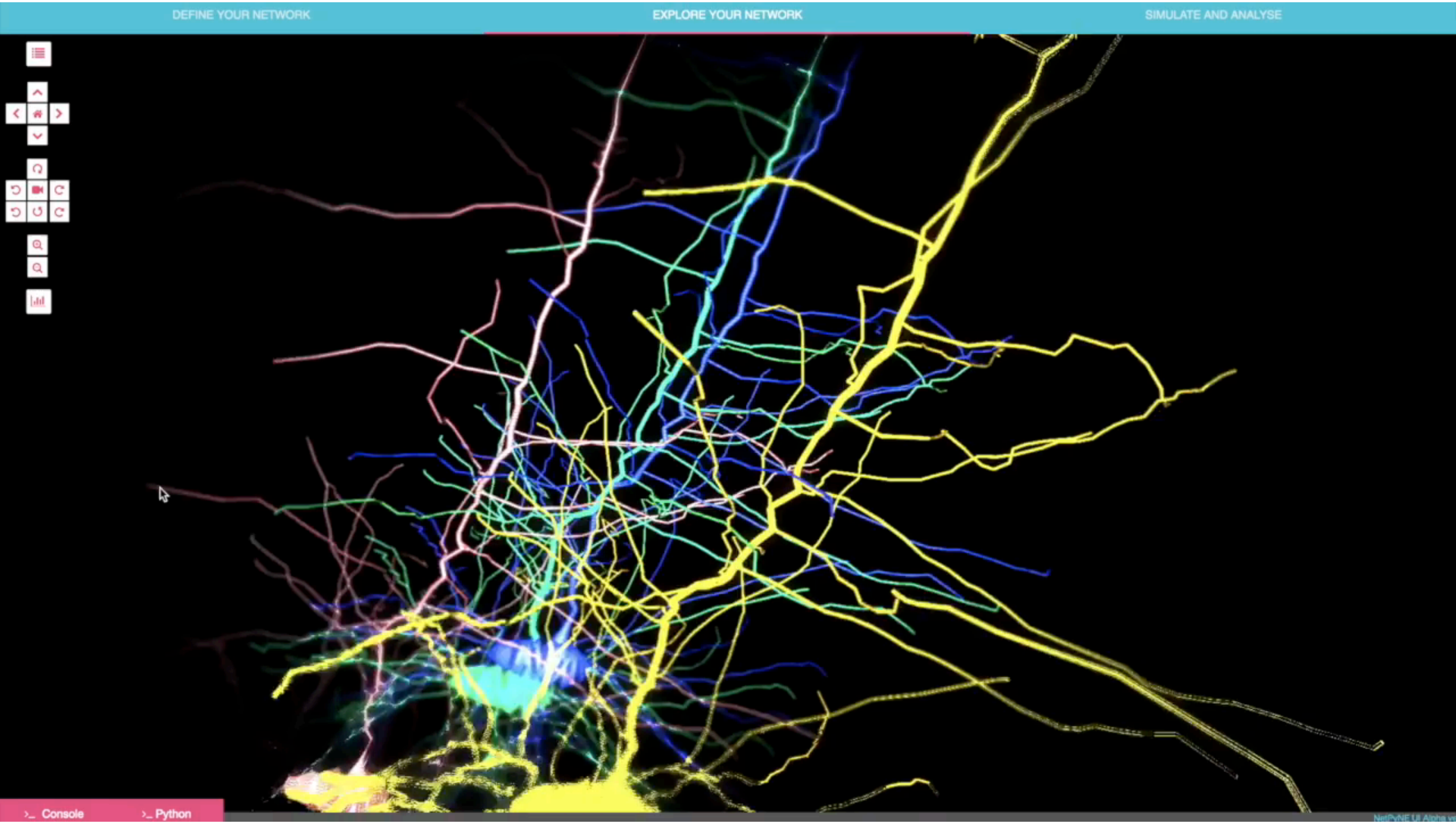


Development, simulation and analysis in GUI

- Useful for:
  1) Students/beginners
  2) Prototypng model (can export to script)
  3) Exploring/visualizing existing models



Development, simulation and analysis in GUI



NetPyNE: Documentation and Tutorials

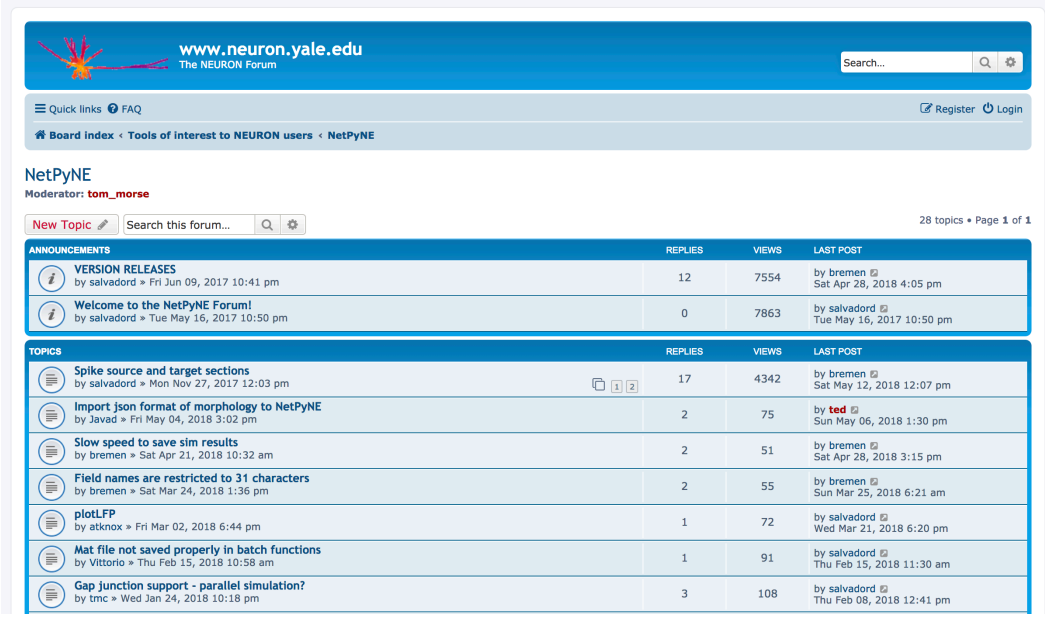
www.netpyne.org

Welcome to NetPyNE's documentation!

NetPyNE is a python package to facilitate the development and parallel simulation of biological cell networks using the NEURON simulator.

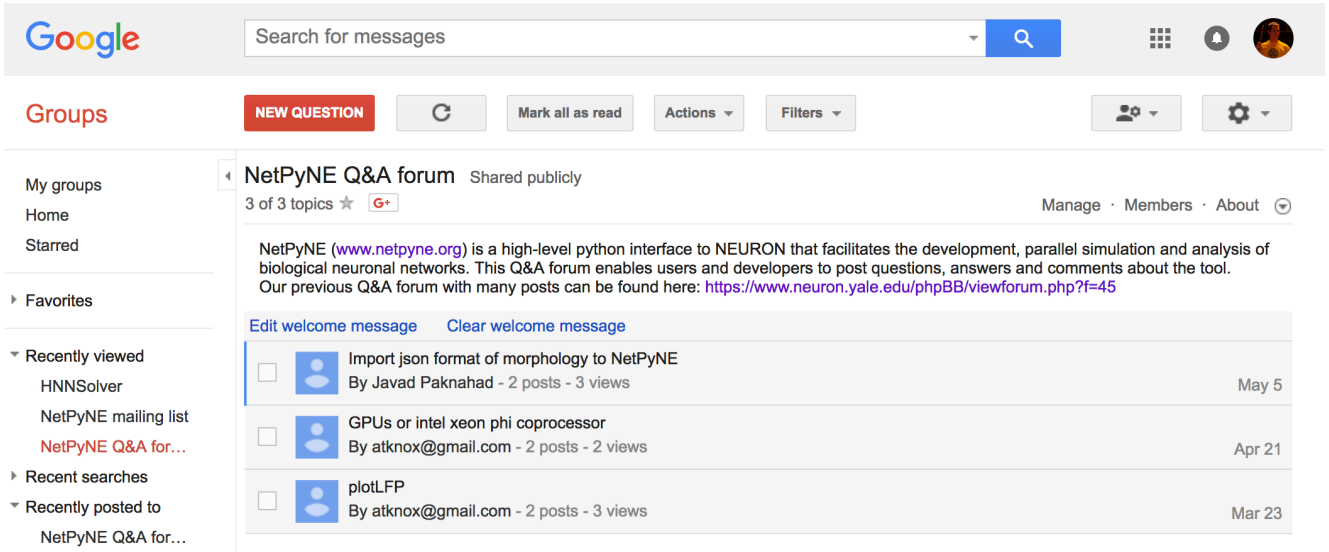
Table of Contents

NetPyNE: Q&A Forums



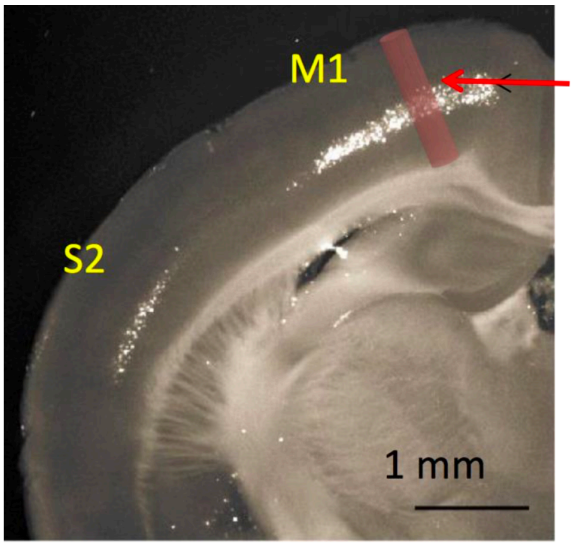
https://www.neuron.yale.edu/phpBB/viewforum.php?f=45&sid=99554ea5df10540d9b31e0c74929eaf0

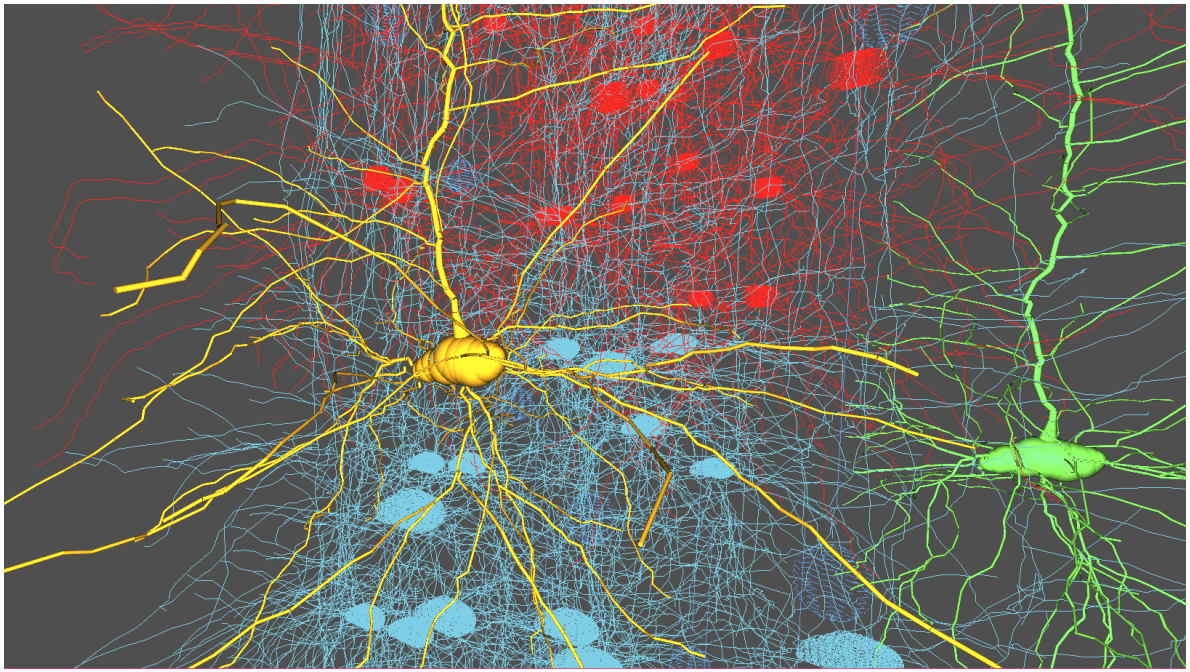
https://groups.google.com/forum/#!forum/netpyne-forum

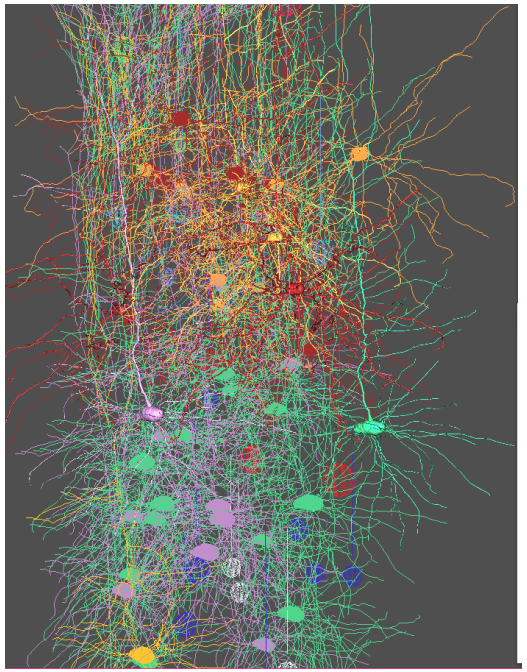


M1 microcircuits model

- Data-driven multiscale network model of **M1 microcircuits**
- Full scale cylindric volume of **300 µm** (diameter) x **1350 µm** (cortical depth)
- **~10,000 neurons** of 5 classes distributed in 15 populations
- ~**30M** synapses

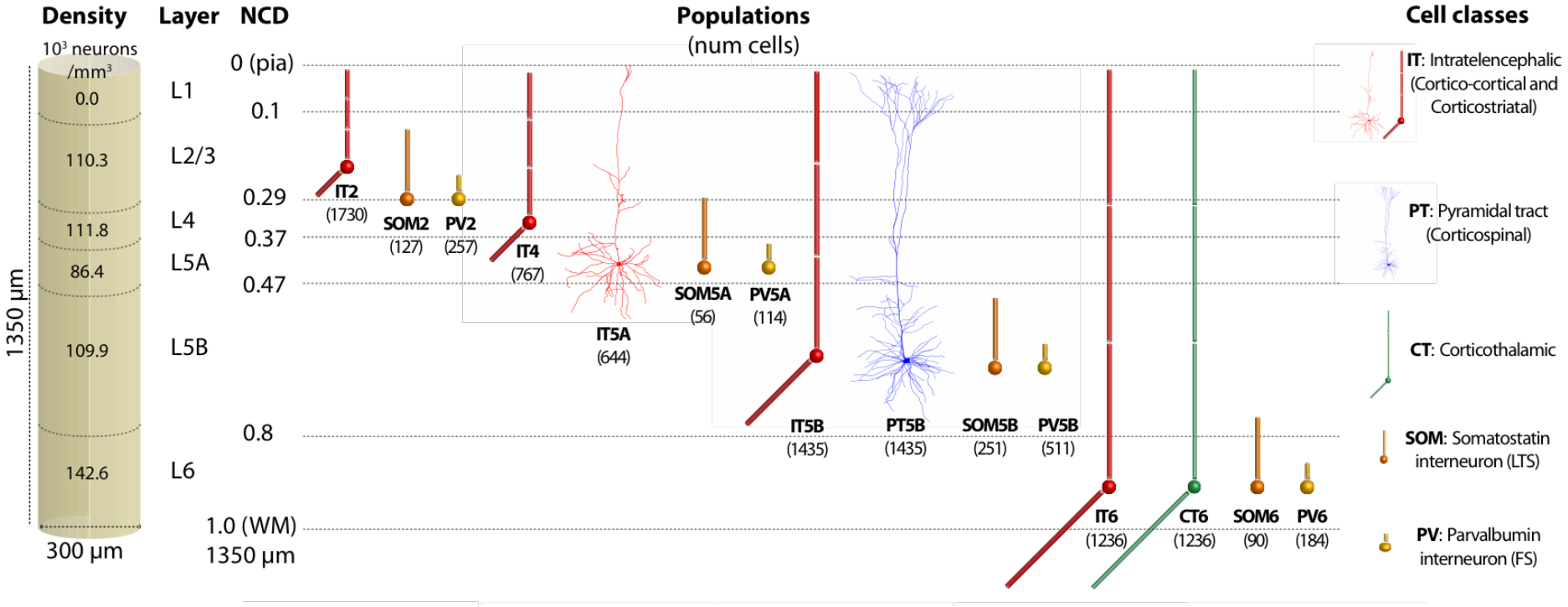




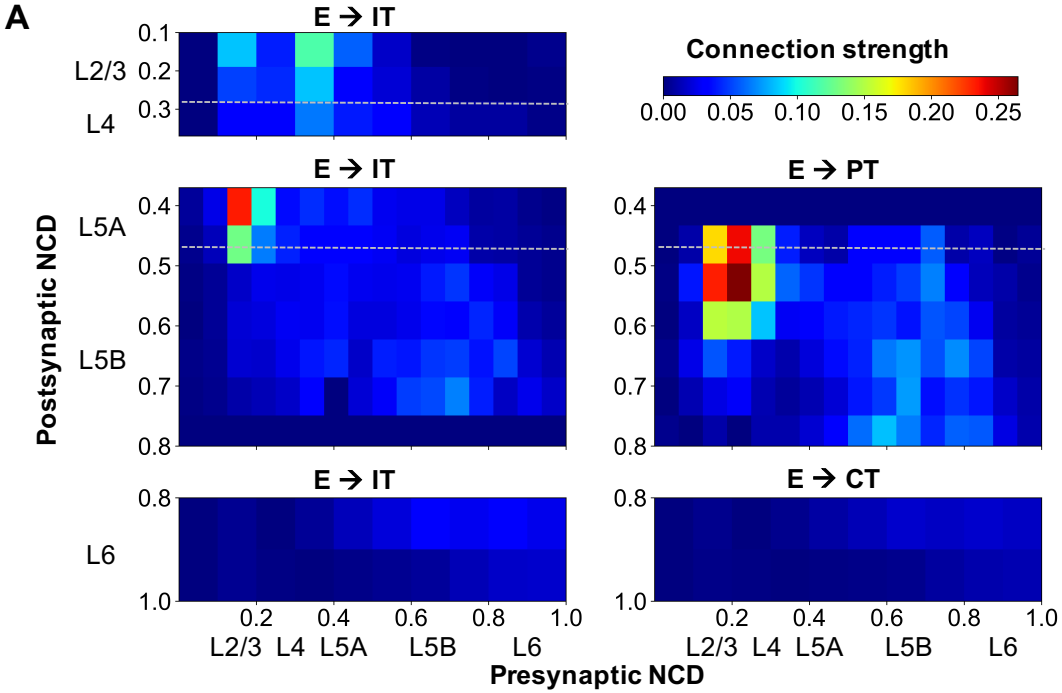


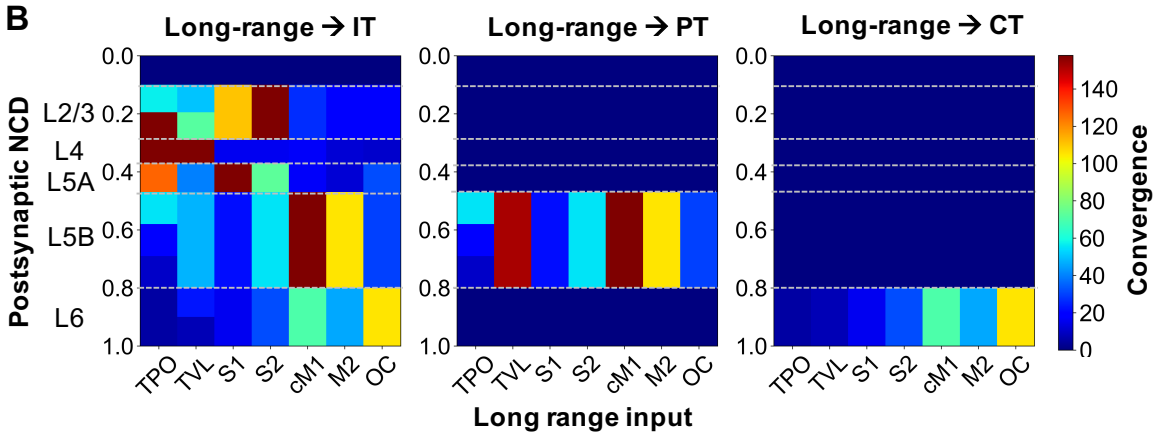
M1 microcircuits model

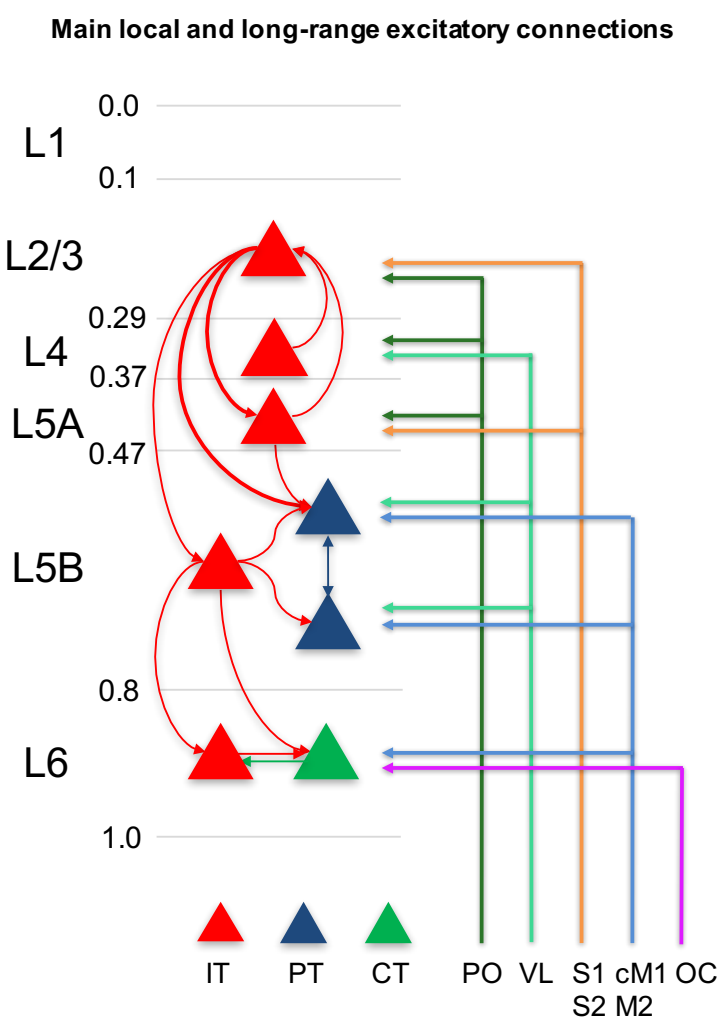
- Data-driven multiscale network model of **M1 microcircuits**



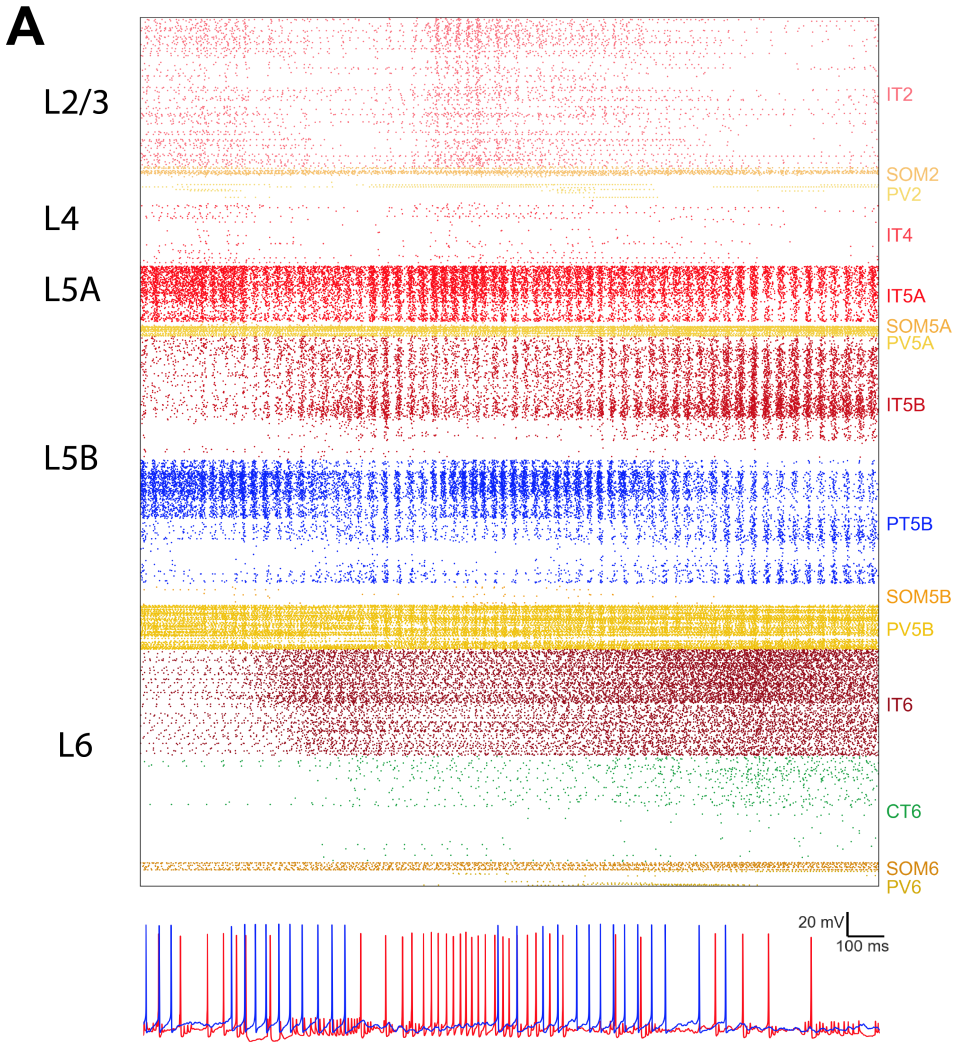
M1 microcircuits model

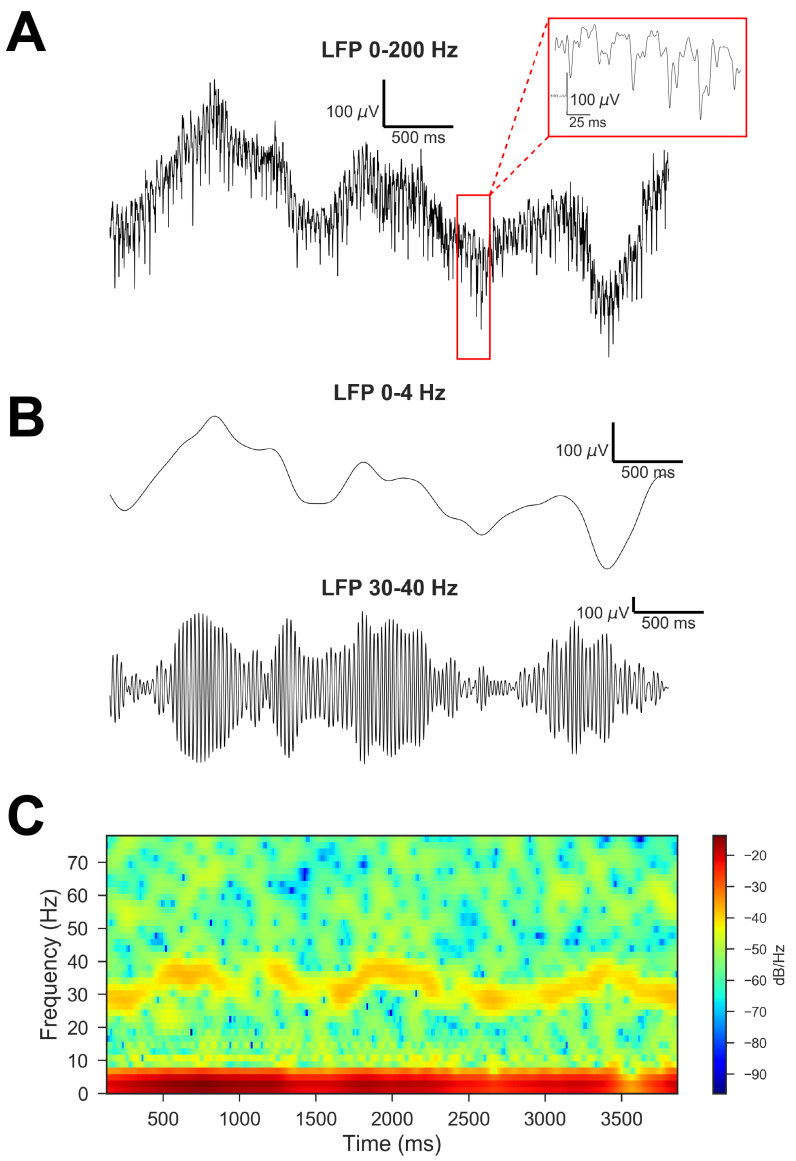






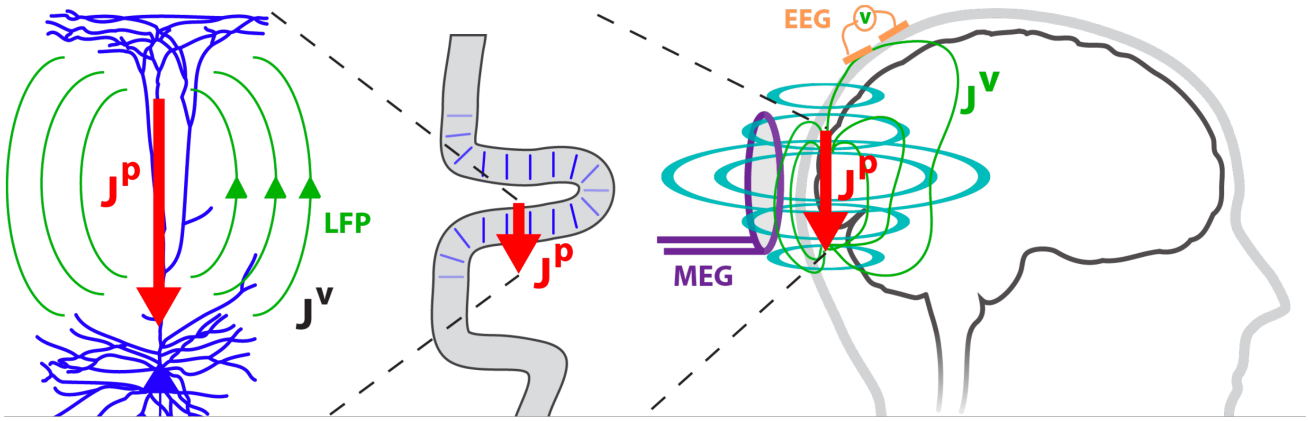
M1 microcircuits model





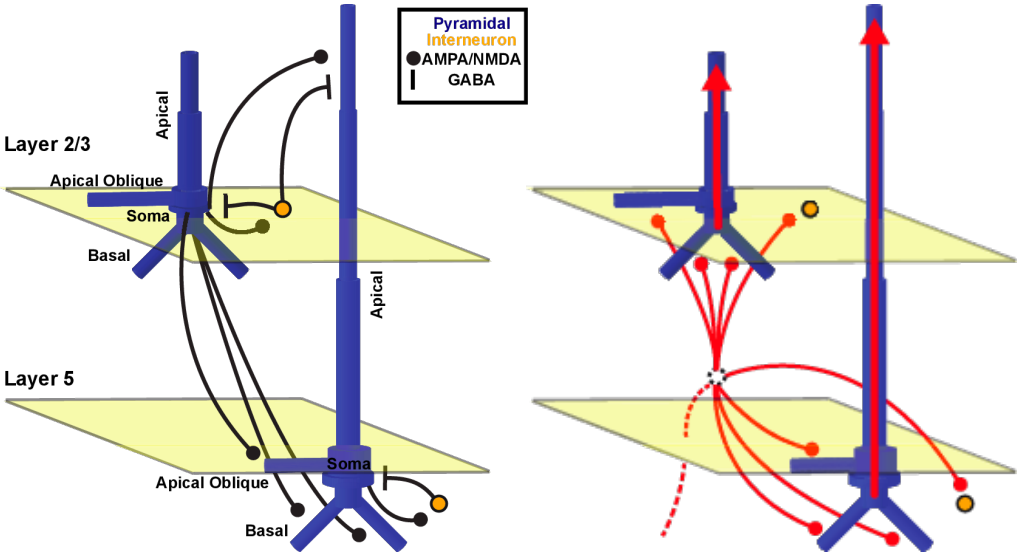
Human Neocortical Neurosolver

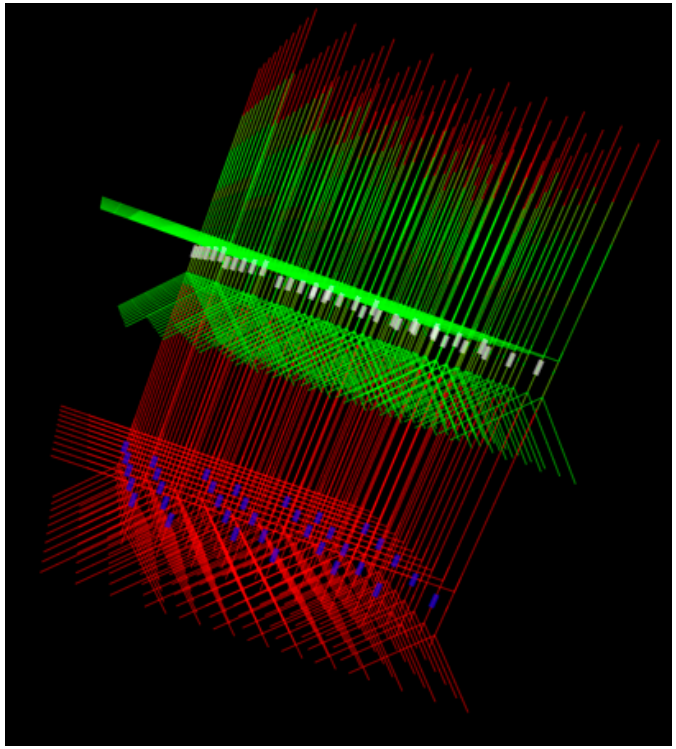
- Stephanie Jones (Brown University), PI of NIH BRAIN R01
- Tool to reproduce/understand EEG/MEG signals using biophysical circuit model



Human Neocortical Neurosolver

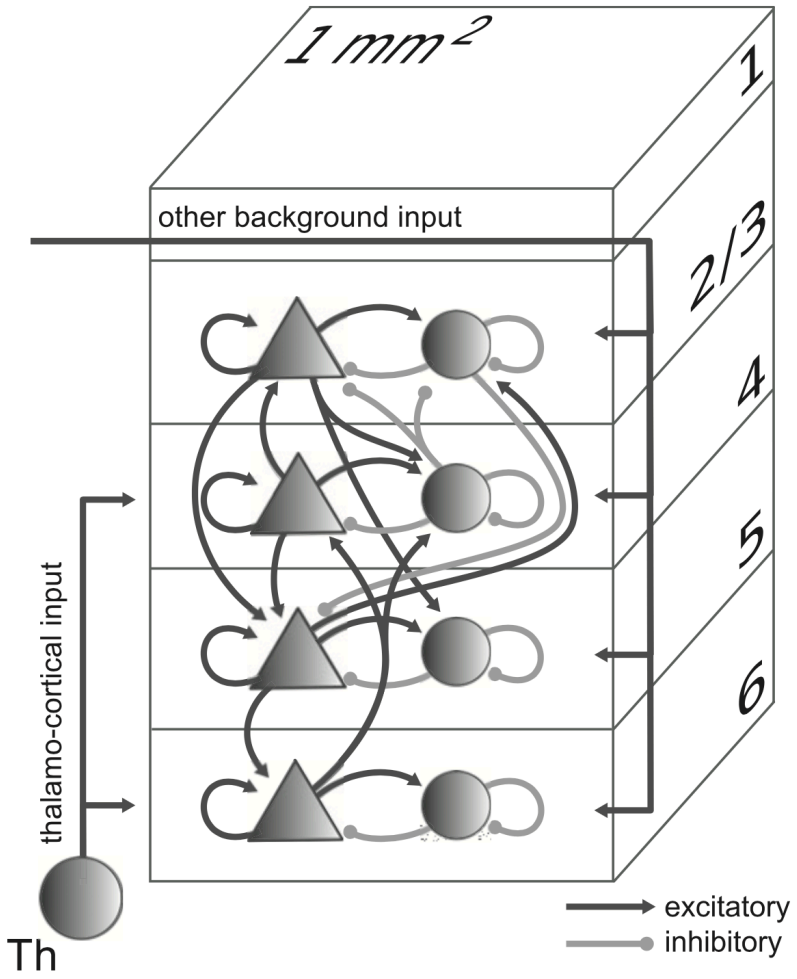
- ❑ Converted circuit model to NetPyNE
- ❑ Facilitate scaling, extension and customization





Potjan's & Diesmann model

- ~80k neurons (point model in NMODL)
- ~300M synapses
- Converted to NetPyNE
- Executed on Google Cloud



Potjan's & Diesmann model

NEST **NetPyNE/NEURON**

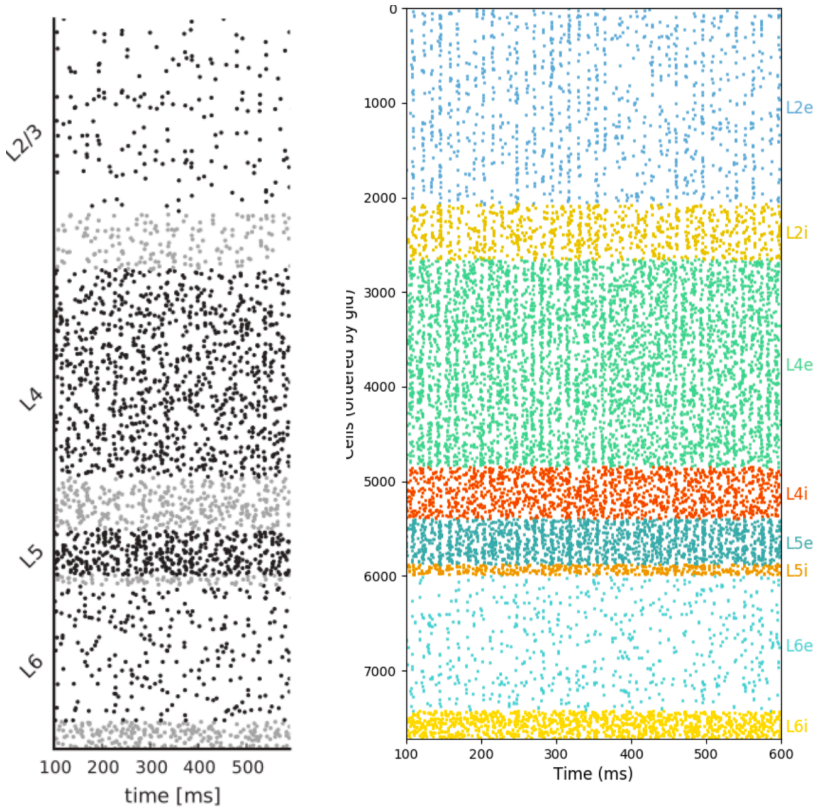


Figure 2: Raster plots network models scaled down to 100% of the original size (with around 80,000 neurons). NEST model on the left and NetPyNE model on the right.

NEST **NetPyNE/NEURON**

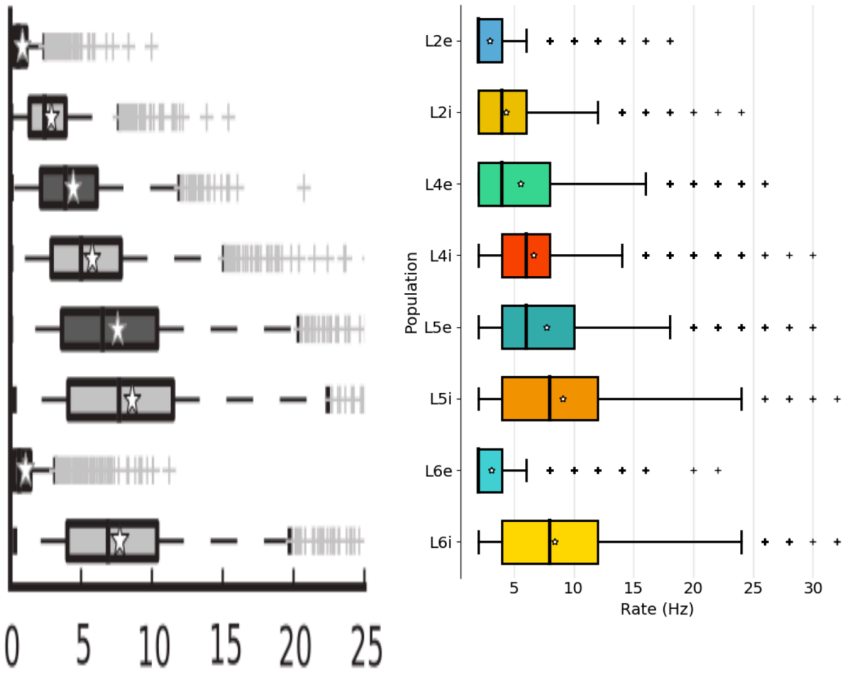
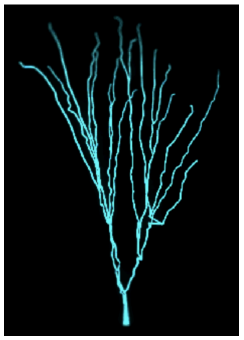
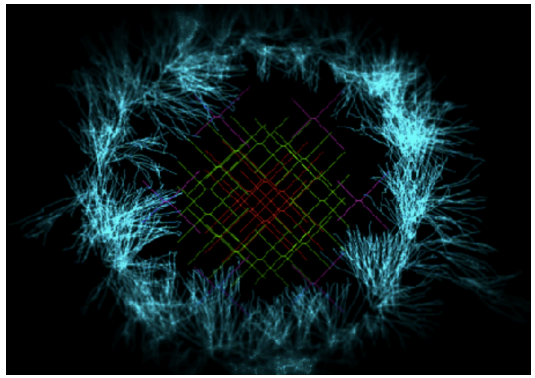


Figure 3: Statistics of spiking activity of all 8 neural populations for rescaling of the PD model to 100% of its original size (around 80,000 neurons). NEST model on the left and NetPyNe model on the right.

NetPyNE: Existing models



- Other models in progress:
  - Traub **thalamocortical** network (P. Gleeson, UCL / S. Crook, Arizona)
  - Hippocampus **CA3** (B. Tessler, SUNY DMC)
  - **Spinal cord** circuits (V. Caggiano, IBM Watson)
  - **Striatal** microcircuits (Hanbing/Christina Weaver, Franklin and Marshall College)
  - **V1** network (Vinicius/Antonio Roque, Sao Paulo University)
  - **Cerebellum** (Sergio Salines/Stefano Masoli, University of Pavia)
  - **Dentate Gyrus** (F. Rodriguez, SUNY DMC)
  - **Ischemia** in cortical network (Alex Seidenstein, SUNY DMC)
  - **TMS/tDCS** network (Aman Aberra, Duke University)
  - **LFP** oscillations (Christian Fink, Ohio Wesleyan)
  - **Dendritic** computations (Birgit Kriener, Oslo)
  - Thalamocortical **epilepsy** network (Andrew Knox, Cincinatti Hospital)
  - Full list of 43 models: https://drive.google.com/open?id=1bkWHakgZoEkYIkzrAS8sIKCvO5PSuUXLLRjNdN2pseY



NetPyNE: Acknowledgments

- **Contributors:**
  - Salvador Dura-Bernal (SUNY DMC)
  - Ben Suter (Northwestern)
  - Matteo Cantarelli (Metacell Ltd)
  - Adrian Quintana (EyeSeeTea Ltd)
  - Dario del Piano (Metacell Ltd)
  - Facundo Rodriguez (SUNY DMC)
  - Cliff Kerr (Sydney)
  - Padraig Gleeson (UCL)
  - Robert McDougal (Yale)
  - Michael Hines (Yale)
  - Gordon MG Shepherd (Northwestern)
  - William Lytton (SUNY DMC)

- **Lab website:** www.neurosimlab.org
- **NetPyNE Website:** www.netpyne.org
- **NetPyNE-UI Website:** www.github.com/MetaCell/NetPyNE-UI
- **Github:** www.github.com/Neurosim-lab/netpyne (open source development; contributions welcome)

- **Funding:**
  - NIH Grant U01EB017695
  - NIH Grant R01EB022903
  - NIH Grant R01MH086638
  - NYS Grant DOH01-C32250GG-3450000

SUNY Downstate Medical Center

Northwestern University

UCL

PYTHON 3:

To install the the package run:

pip3 install netpyne_py3 (Linux or Mac OS) or
python -m pip install netpyne_py3 (Windows)

To upgrade to a new version run:

pip3 install netpyne_py3 -U (Linux or Mac OS) or
python -m pip install -U netpyne_py3 (Windows)

TO TEST:
In python interpreter type: ***from netpyne import sim***