A python package to facilitate the development, parallel simulation, optimization and analysis of biological neuronal networks using the NEURON simulator.
NetPyNE motivation

Separate model parameters from standardized implementation

Standardize format – easy to read, interpret, edit, share, reproduce, etc

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

for gid in range(pop.numCells):
    cell = sim.Cell()
    cell.y = numpy.random(100, 400)
    cell.type = 'PYR'
    pc.cell(gid, h.NetCon(v_soma, threshold))
```

Standard, efficient and tested backend implementation

Potentially very different implementations (arbitrary functions, variables, file names etc.)
NetPyNE motivations

• Facilitate model **parallelization** (reproducible)

• **Batch** parameter exploration/optimization
NetPyNE tool: Publication in eLife

NetPyNE, a tool for data-driven multiscale modeling of brain circuits

Salvador Dura-Bernal, Benjamin A Suter, Padraig Gleeson, Matteo Cantarelli, Adrian Quintana, Facundo Rodriguez, David J Kedziora, George L Chadderdon, Cliff C Kerr, Samuel A Neymotin, Robert A McDougal, Michael Hines, Gordon M G Shepherd, William W Lytton

State University of New York Downstate Medical Center, United States; Northwestern University, United States; University College London, United Kingdom; Metacell LLC, United States; EyeSeeTea Ltd, United Kingdom; University of Sydney, Australia; Yale University, United States
NetPyNE

**Batch simulation module (parameter exploration, MPI/HPC job submission, etc)**

- **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, ...

- **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**

**NEURON cell models**

**NeuroML cell and network models**
High level specifications

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

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

preCon = h.NetCon(preCell1.soma(.5), _ref_v, ...
    postSyn1,
    sec=preCell1.soma)
preCon.delay = 1
preCon.weight[0] = 0.001
preCon.threshold = 0
```
High level specifications

NetPyNE GUI

NetPyNE

NEURON

Connectivity rules
Define here the rules to generate the connections in your network

S→M

Add new Prosthetic neurons section
dend
Add new Prosthetic neurons location (S→M)
0.5

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

preCon = h.NetCon(preCell.soma(0.5)._ref_v,
postSyn1,
sec=postCell.soma)
preCon.delay = 1
preCon.weight[0] = 0.001
preCon.threshold = 0
A **standardized, declarative** Python format (JSON-like, lists and dicts) to define:

- **Populations**: cell type, number of neurons or density, spatial extent, ...
- **Cell properties**: Morphology, biophysics, molecular processes ...
- **Synaptic mechanisms**: Time constants, reversal potential, ...
- **Stimulation**: Spike generators, current clamps, spatiotemporal patterns, ...
- **Connectivity rules**: conditions of pre- and post-synaptic cells, functions, ...
- **Simulation configuration**: duration, saving and analysis, visualization, ...
High level specifications

Molecular reaction-diffusion (RxD)

- Intra- and extracellular diffusion of ions, proteins (eg, calcium, potassium, IP3, ...)
- Cell internal structures/organelles (eg, endoplasmic reticulum, mitochondria,...)
- Molecular processes (eg, phosphorylation, buffering, 2nd messenger cascades,...)
- Interaction with cell and network scales (eg, firing, plasticity, ...)
High level specifications

Connectivity

- **Flexible connectivity rules** based on pre- and post-synaptic cell properties (e.g., type or location).
- Connectivity **functions** available: probabilistic, convergent, divergent, custom, ...
- Parameters (e.g., probability, weight, delay) as a **function of pre/post-synaptic spatial properties**, e.g., delays or probability that depend on distance between cells or cortical depth.
- Easily add synapses with **learning** mechanisms (STDP and RL) and **gap junctions**.

NetPyNE facilitates building models based on experimental data.
High level specifications

Connectivity

- Specify dendritic **distribution of synapses** as 1D or 2D density map, or based on distance from section
- Synaptic distribution automatically **adapted to morphology** of each cell model
Network Instantiation

Network instance as **standardized hierarchical** Python structure (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
Local Field Potentials

Single cell

Network
Data Saving and Exporting

- **Save and load** high-level specifications, network instance, simulation config, simulation results.
- **Multiple formats** supported: pickle, JSON and Matlab (CSV and HDF5 in progress)
- **Export/import** network instance to/from **NeuroML** and **SONATA**
Useful for
Students and beginners learning
Model prototyping (can export to script)
Exploring/Visualizing existing models
Batch Parallel Simulation

- **Easy specification** of parameters and range of values to explore in batch simulations (evolutionary + grid search)
- **Pre-defined, configurable** setups to automatically **submit jobs** in multicore machines (Bulletin board) or supercomputers (SLURM or PBS Torque)
Batch Parallel Simulation

- University cluster: 500 cores
- Scientific shared cluster: 7200 cores
- Google Cloud: 50,000 cores

(number of simultaneous cores)
Batch Parallel Simulation

Analysis and visualization of multidimensional batch simulation results.
Documentation and Tutorials

Table of Contents

- NetPyNE Overview
  - What is NetPyNE?
  - What can I do with NetPyNE?
  - NetPyNE structure
  - Major Features
  - Questions, suggestions and contributions
  - Publications
    - About NetPyNE
    - Using NetPyNE
- Installation
  - Requirements
  - Install via pip (latest released version)
  - Install via pip (development version)
  - Install NetPyNE GUI (alpha version)
- Tutorial
  - Very simple and quick example (Tutorial 1)
  - Network parameters (Tutorial 2)
    - Populations
    - Cell property rules
    - Synaptic mechanisms parameters
    - Simulation
    - Connectivity rules
  - Simulation configuration options
  - Network creation and simulation
  - Adding a compartment (dendrite) to cells (Tutorial 3)
  - Using a simplified cell model (Izhikevich) (Tutorial 4)
  - Position and distance based connectivity (Tutorial 5)
  - Adding stimulation to the network (Tutorial 6)
  - Modifying the instantiated network interactively (Tutorial 7)
  - Running batch simulations (Tutorial 8)
  - Recording and plotting LFPs (Tutorial 9)
- Package Reference

Finally, we add inhibitory connections which will project only onto excitatory cells, specified here using the `gtype` attribute, for illustrative purposes (an equivalent rule would be: `postconds`:

```python
{ 'celltype': 'E' }
```

To make the probability of connection decay exponential as a function of distance with a given length constant (`probDecayDistance`), we can use the following distance-based expression:

```python
'probability': $0.4 \times \exp(-\text{dist}_{ij}/\text{probDecayDistance})$
```

The code for the inhibitory connectivity rule is therefore:

```python
netParams.connParams['I-E'] = {
    'preConds': { 'celltype': I },
    'postConds': { 'gtype': [E2, E4, E5] }, # I -> E
    'probability': $0.4 \times \exp(-\text{dist}_{ij}/\text{probDecayDistance})$, # probability of connection
    'weight': 0.011,
    'synaptic': 'exp',  # synaptic type
    'synMech': 'fastExc', # synaptic mechanism
    'stimMech': 'ion' }  # stimulation mechanism
```

Notice that the 2D network diagram now shows inhibitory connections in blue, and these are mostly local/lateral within layers, due to the distance-related probability restriction. These local inhibitory connections reduce the overall synchrony, introducing some richness into the temporal firing patterns of the network.

http://netpyne.org

The full tutorial code for this example is available here: `tutorial.py`
Q&A and Forums

NetPyNE is a high-level Python interface to NEURON that facilitates the development, parallel simulation, and analysis of biological neuronal networks. This Q&A forum enables users and developers to post questions, answers, and comments about the tool. Our previous Q&A forum with many posts can be found here: https://www.neuron.yale.edu/pynl/eviewforum.php?45.

New Topic Search this forum...
Example brain models
Existing NetPyNE Models

- Traub **thalamocortical** network (P. Gleeson, UCL / S. Crook, Arizona)
- Hippocampus **CA3** (B. Tessler, SUNY DMC)
- **Auditory cortex** circuits (S. Neymotin, Nathan Kline Institute)
- **Striatal** microcircuits (Hanbing/C. Weaver, Franklin and Marshall College)
- **V1** network (Vinicius/A. Roque, Sao Paulo University)
- **Cerebellum** (S. Solinas, University of Pavia)
- **Dentate Gyrus** (F. Rodriguez, SUNY Downstate)
- **Ischemia** in cortical network (A. Newton, Yale)
- **TMS/tDCS** network (A.Aberra, Duke University)
- Cortical **seizures** with RxD (W. Truccolo, Brown)
- **Claustrum** network (Jing Lin, A*STAR Singapore)
- Thalamocortical **epilepsy** network (A. Knox, Cincinatti Hospital)

Full list of 53 models (many under development): [www.netpyne.org/models](http://www.netpyne.org/models)
Human Neocortical Neurosolver

- Stephanie Jones (Brown), NIH BRAIN R01
- Tool to reproduce/understand EEG/MEG signals using biophysical circuit model (hnn.brown.edu)

Human Neocortical Neurosolver

- Converted circuit model to NetPyNE
- Facilitate scaling, extension and customization (bit.ly/hnn-gui)
Tutorial useful links

- These slides: bit.ly/cns19-netpyne
- Code used in GUI tutorials: https://github.com/Neurosim-lab/netpyne_workspace
- Step-by-step Coding tutorials: www.netpyne.org/tutorial.html