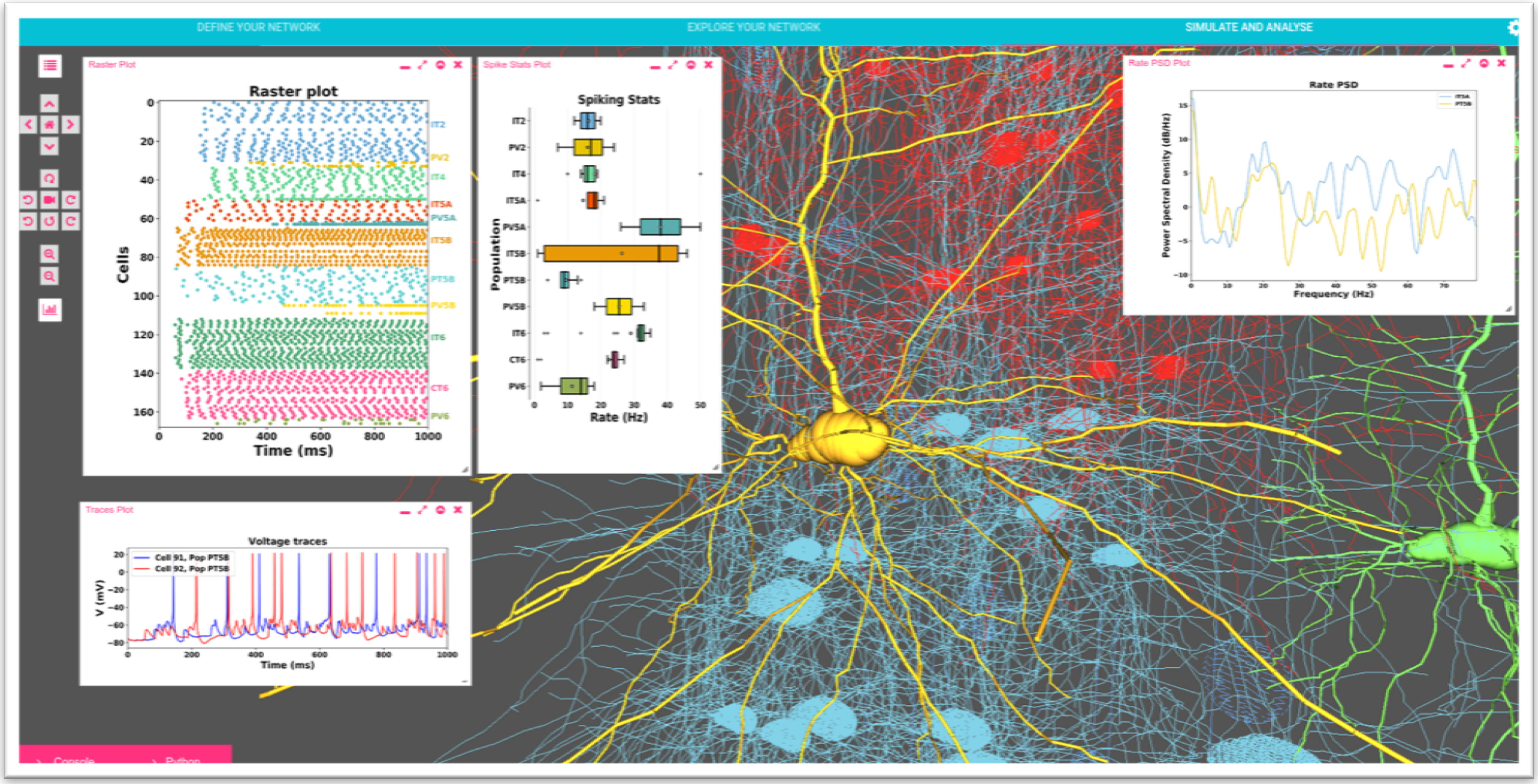
NetPyNE GUI Tutorial



NetPyNE GUI Tutorial: Installation

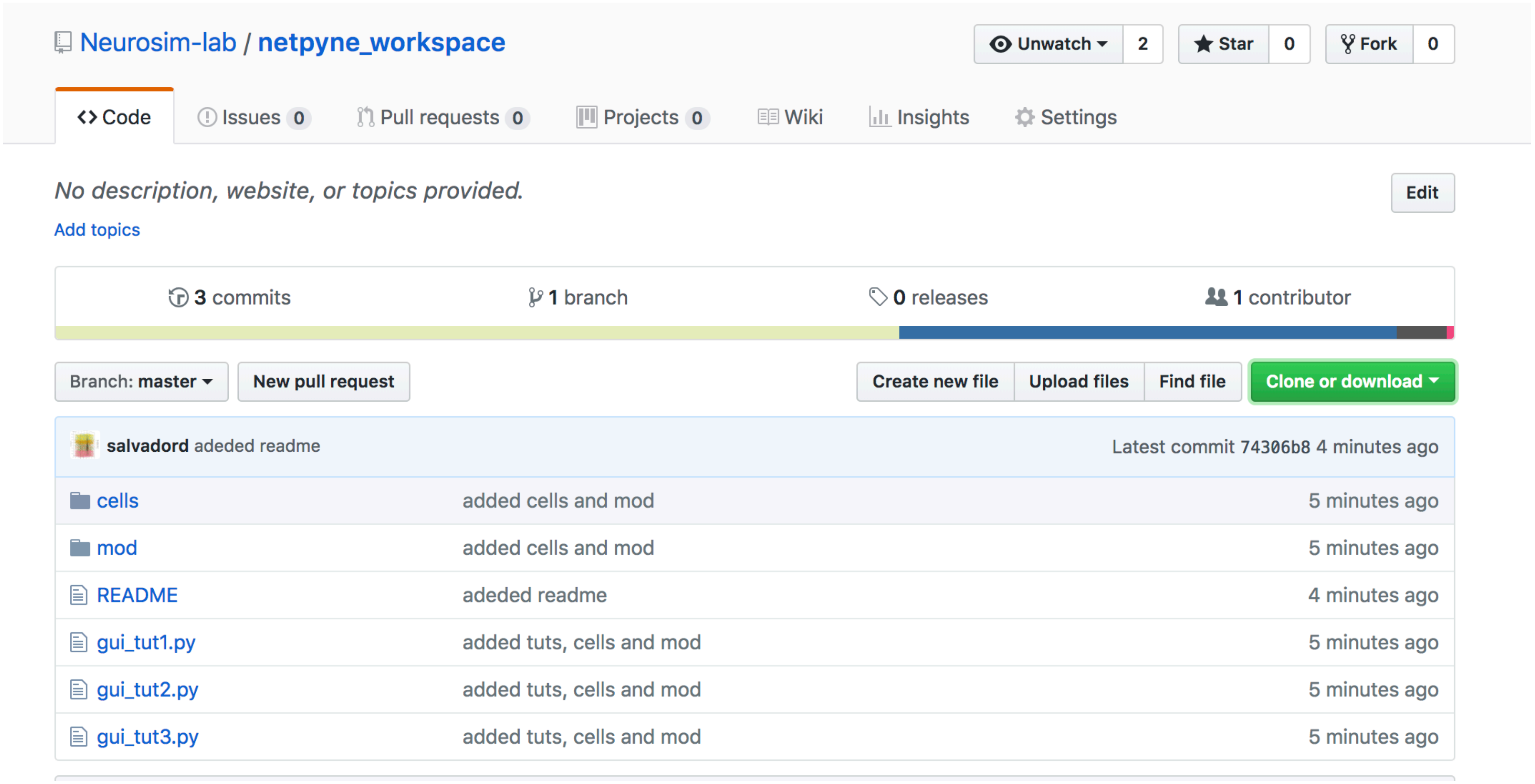
Download this tutorial PDF:

bit.ly/netpyne-ui-tut

NetPyNE GUI Tutorial: Installation

Clone or download this Github repo with this tutorial and the workspace files:

https://github.com/Neurosim-lab/netpyne_workspace.git



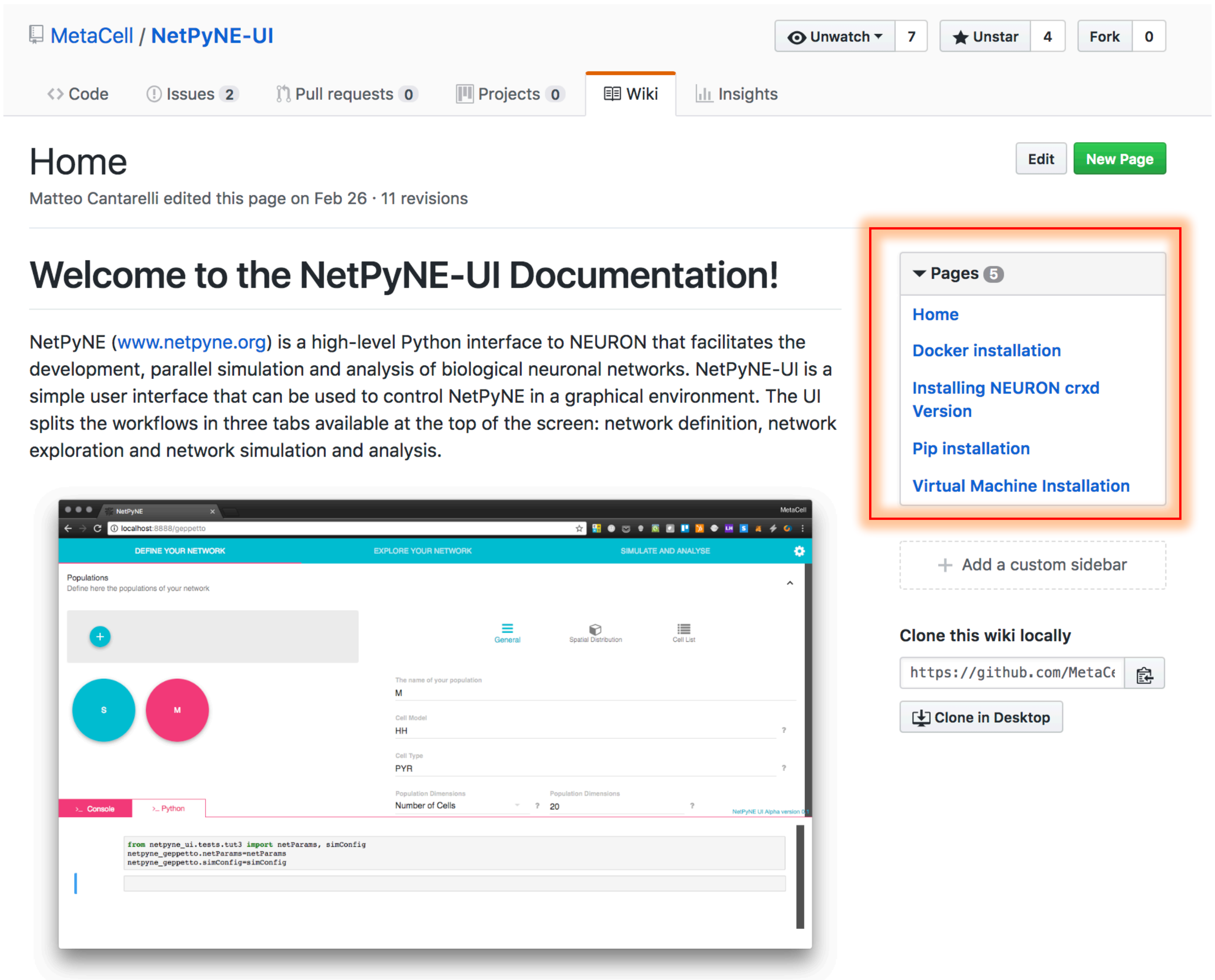
No description, website, or topics provided.

salvadord adeded readme — Latest commit 74306b8 4 minutes ago

cells	added cells and mod	5 minutes ago
mod	added cells and mod	5 minutes ago
README	adeded readme	4 minutes ago
gui_tut1.py	added tuts, cells and mod	5 minutes ago
gui_tut2.py	added tuts, cells and mod	5 minutes ago
gui_tut3.py	added tuts, cells and mod	5 minutes ago

NetPyNE GUI Tutorial: Installation

Instructions: https://github.com/MetaCell/NetPyNE-UI/wiki



MetaCell / **NetPyNE-UI**

Home

Matteo Cantarelli edited this page on Feb 26 · 11 revisions

Welcome to the NetPyNE-UI Documentation!

NetPyNE (www.netpyne.org) is a high-level Python interface to NEURON that facilitates the development, parallel simulation and analysis of biological neuronal networks. NetPyNE-UI is a simple user interface that can be used to control NetPyNE in a graphical environment. The UI splits the workflows in three tabs available at the top of the screen: network definition, network exploration and network simulation and analysis.

The name of your population: M

Cell Model: HH

Cell Type: PYR

Population Dimensions: Number of Cells

Population Dimensions: 20

```
from netpyne_ui.tests.tut3 import netParams, simConfig
netpyne_geppetto.netParams=netParams
netpyne_geppetto.simConfig=simConfig
```

Pages 5

- Home
- Docker installation
- Installing NEURON crxd Version
- Pip installation
- Virtual Machine Installation

Clone this wiki locally

`https://github.com/MetaCe`

NetPyNE GUI Tutorial: Installation

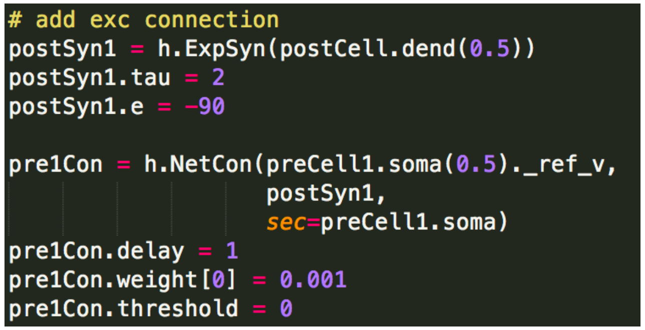
Option 1: Install NEURON crxd from **sources** (Github) and NetPyNE-UI via **pip**

Option 2: Use pre-packaged **Docker** with all you need

Option 3: Use pre-packaged **Virtual Machine** with all you need

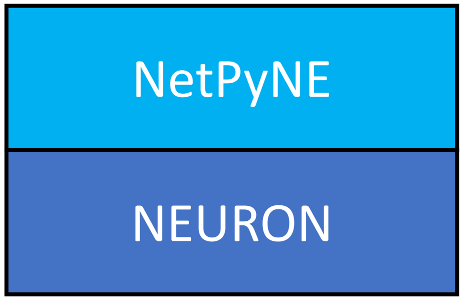
What is what…

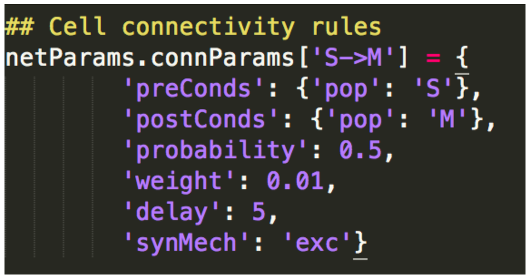
NEURON



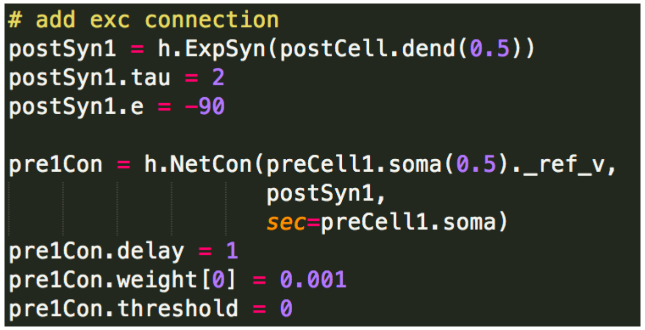
```python
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

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

What is what…



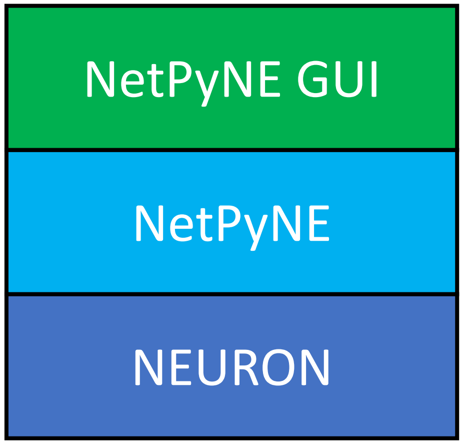


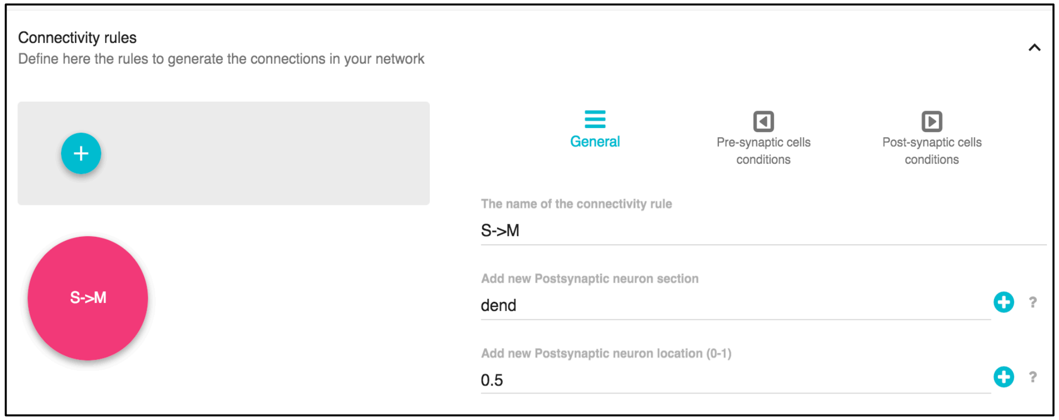
```
## Cell connectivity rules
netParams.connParams['S->M'] = {
        'preConds': {'pop': 'S'},
        'postConds': {'pop': 'M'},
        'probability': 0.5,
        'weight': 0.01,
        'delay': 5,
        'synMech': 'exc'}
```



```
# 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
```

What is what...

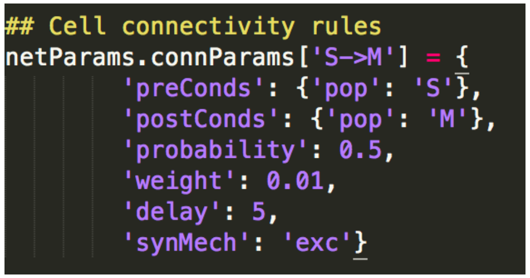


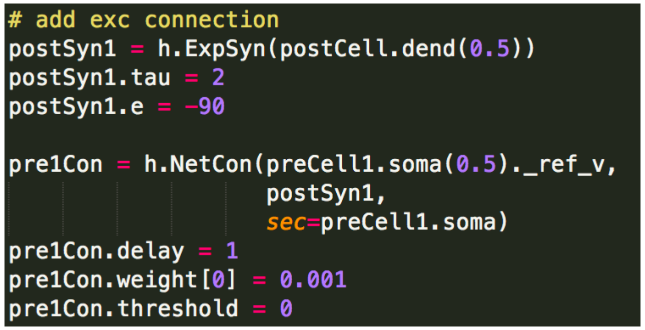

Connectivity rules
Define here the rules to generate the connections in your network

The name of the connectivity rule
S->M

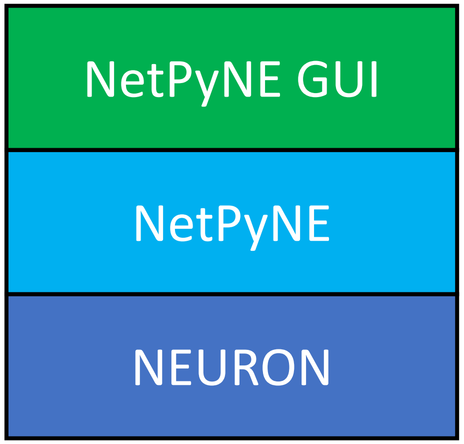
Add new Postsynaptic neuron section
dend

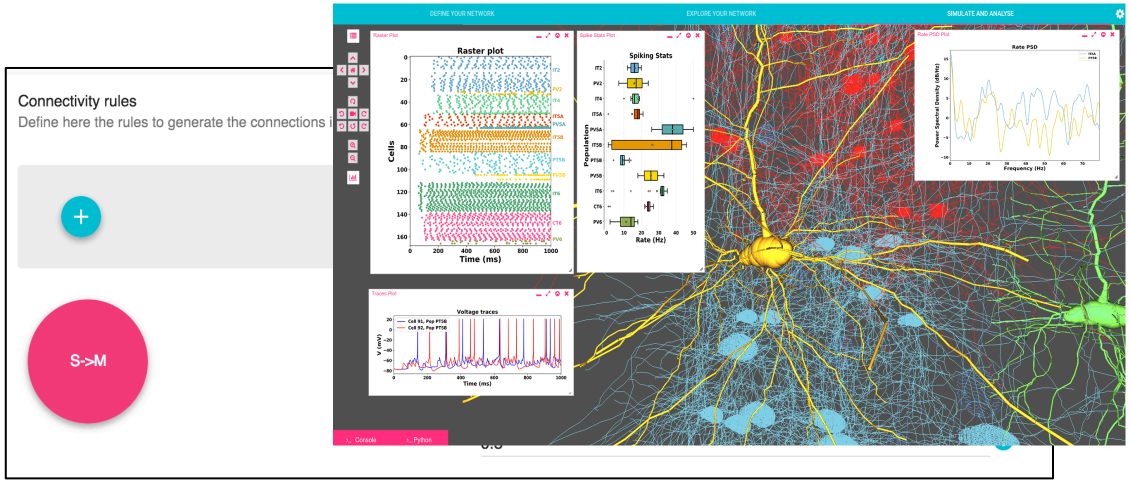
Add new Postsynaptic neuron location (0-1)
0.5


```
## 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

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

What is what…



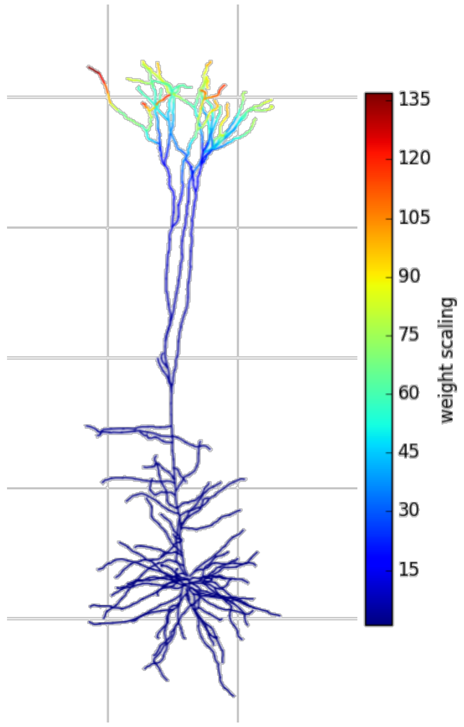

Connectivity rules
Define here the rules to generate the connections

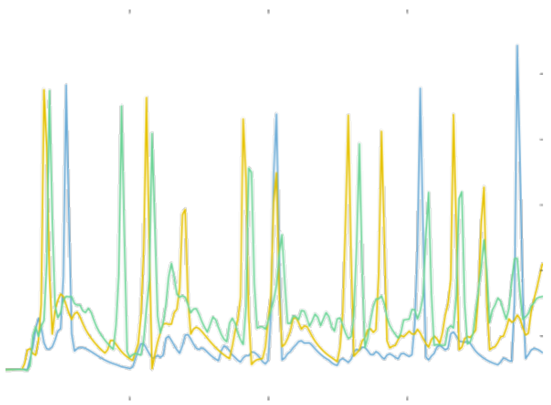
```python
## Cell connectivity rules
netParams.connParams['S->M'] = {
        'preConds': {'pop': 'S'},
        'postConds': {'pop': 'M'},
        'probability': 0.5,
        'weight': 0.01,
        'delay': 5,
        'synMech': 'exc'}
```

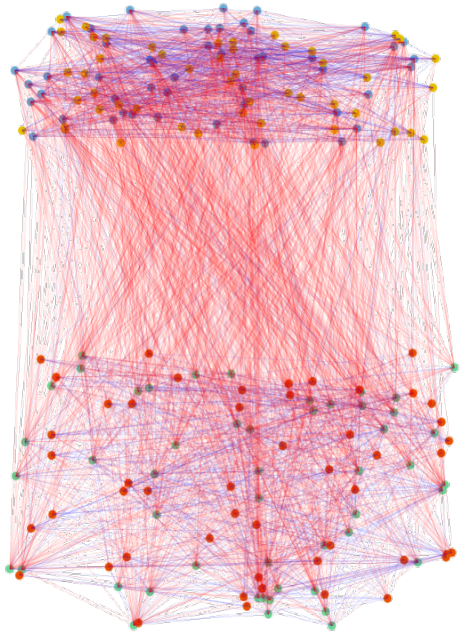
```python
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

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

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

www.netpyne.org

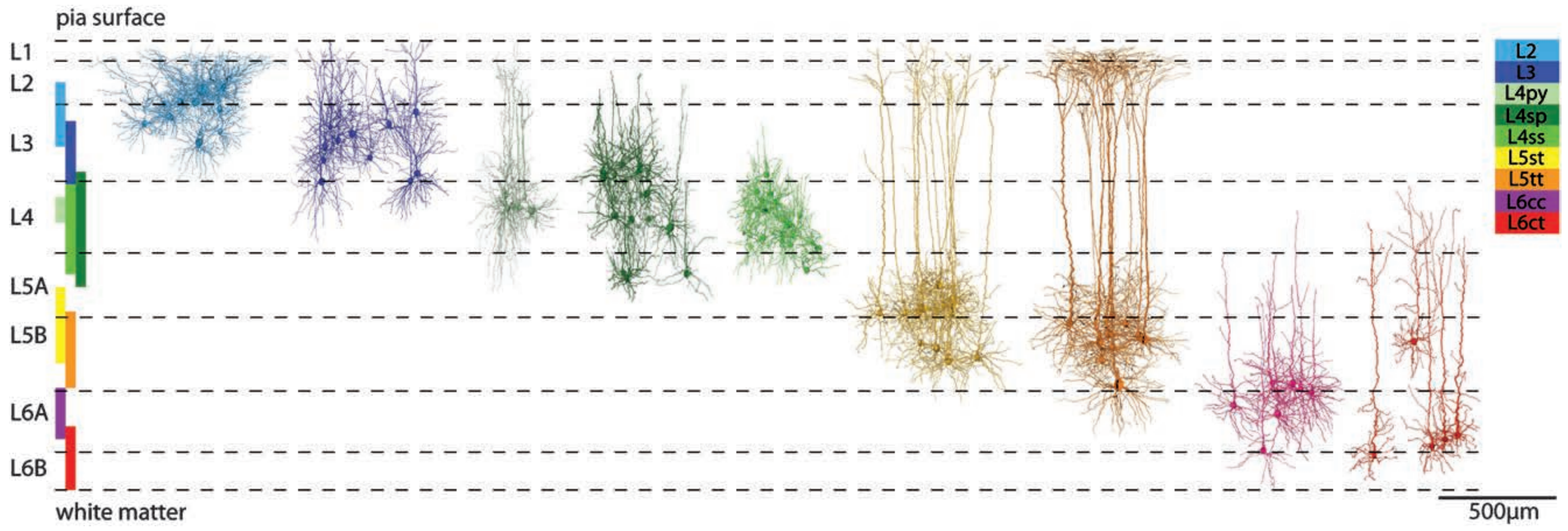






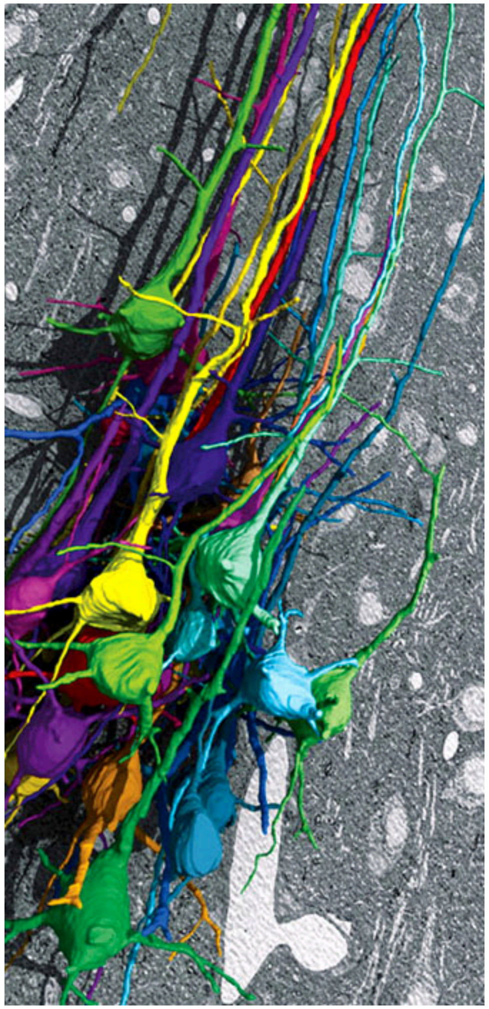
NetPyNE: Motivation

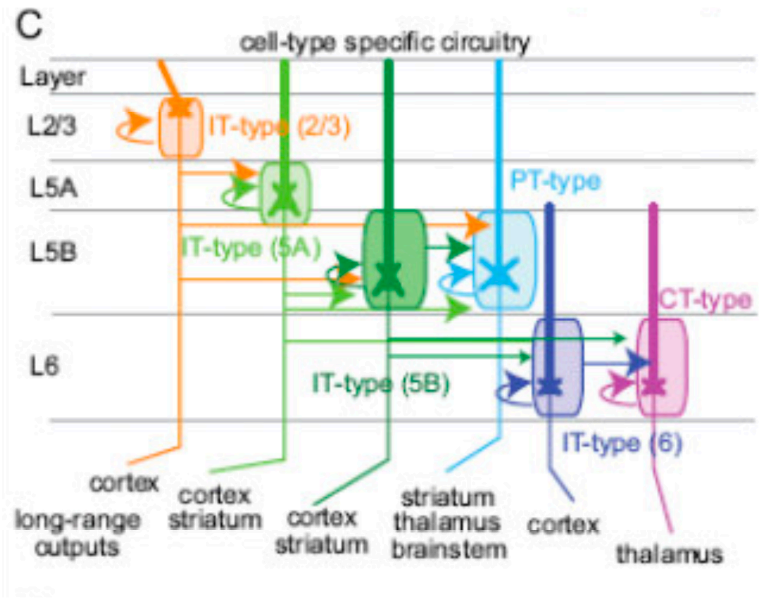
- Facilitate incorporation of experimental data at multiple scales



NetPyNE: Motivation

- Facilitate incorporation of experimental data at multiple scales

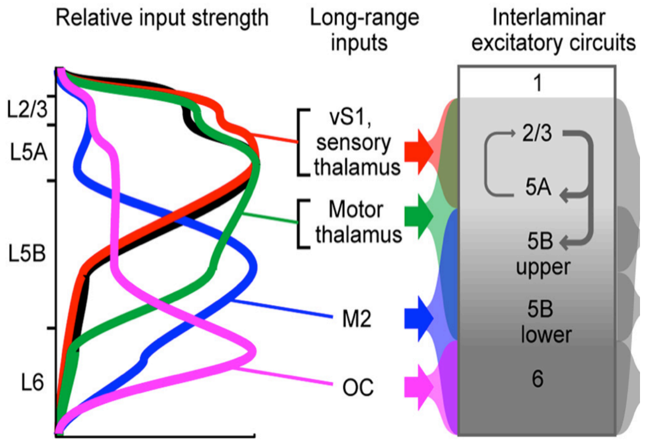




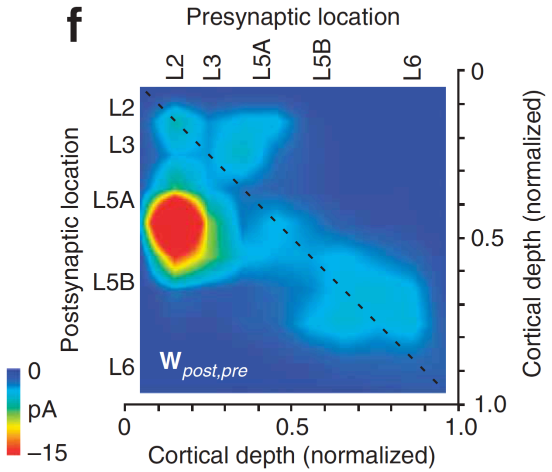
NetPyNE: Motivation

- Facilitate incorporation of experimental data at multiple scales

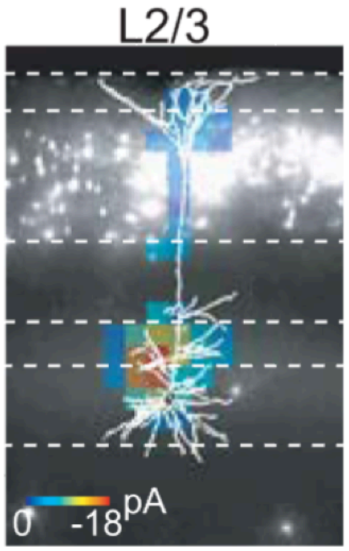
Long-range inputs



Local microcircuits



Dendritic inputs



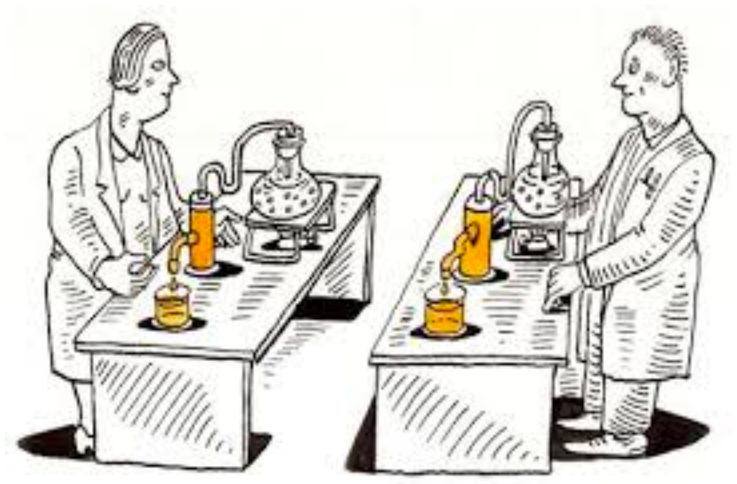
NetPyNE: Motivation

- 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'
```

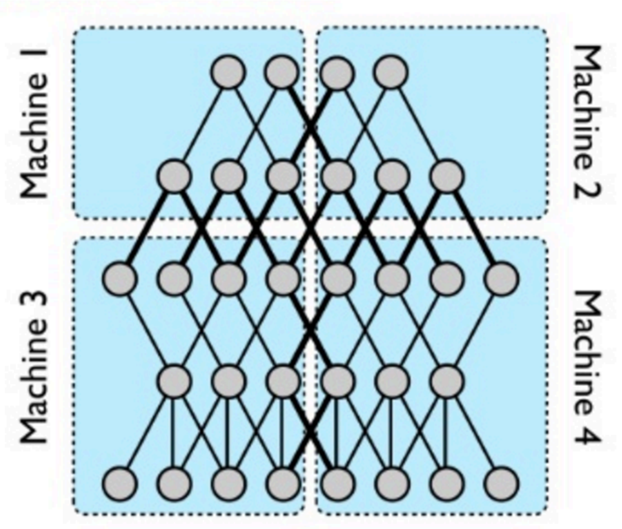


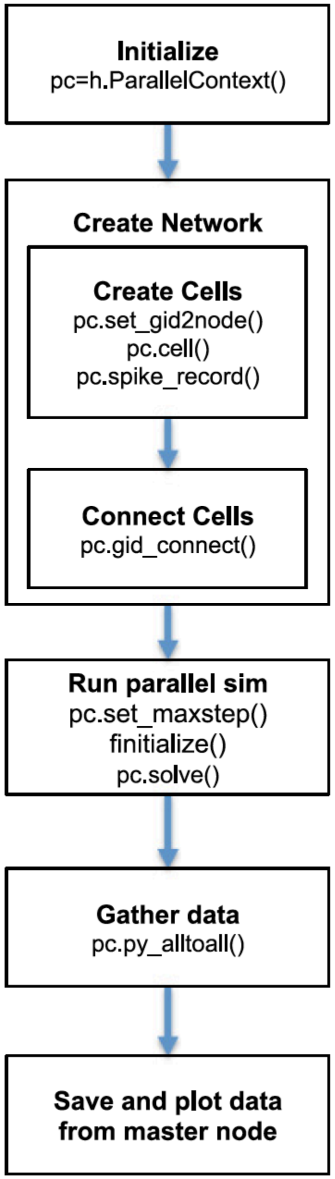
Replicate: get same thing to run again

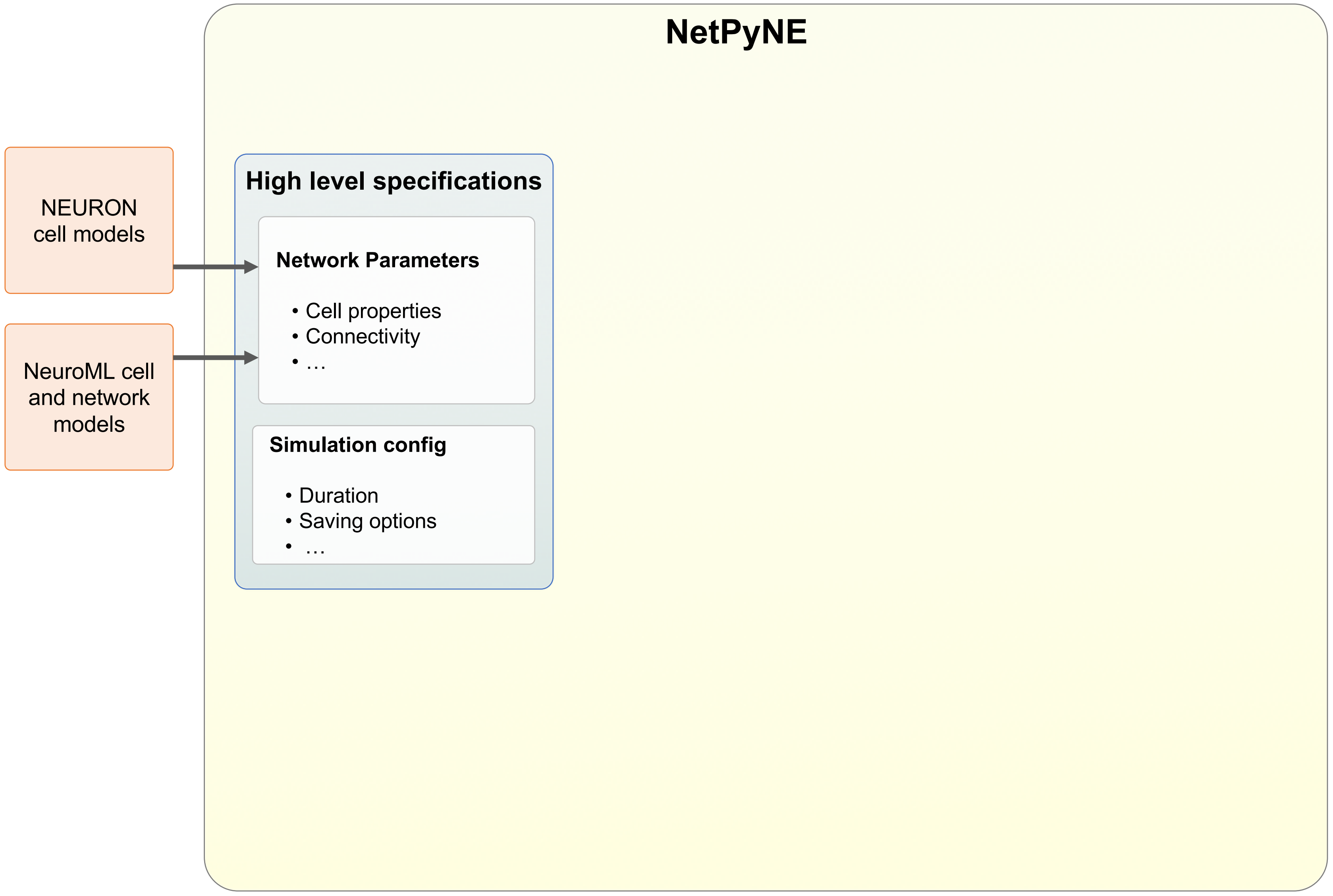
Reproduce: make it youself

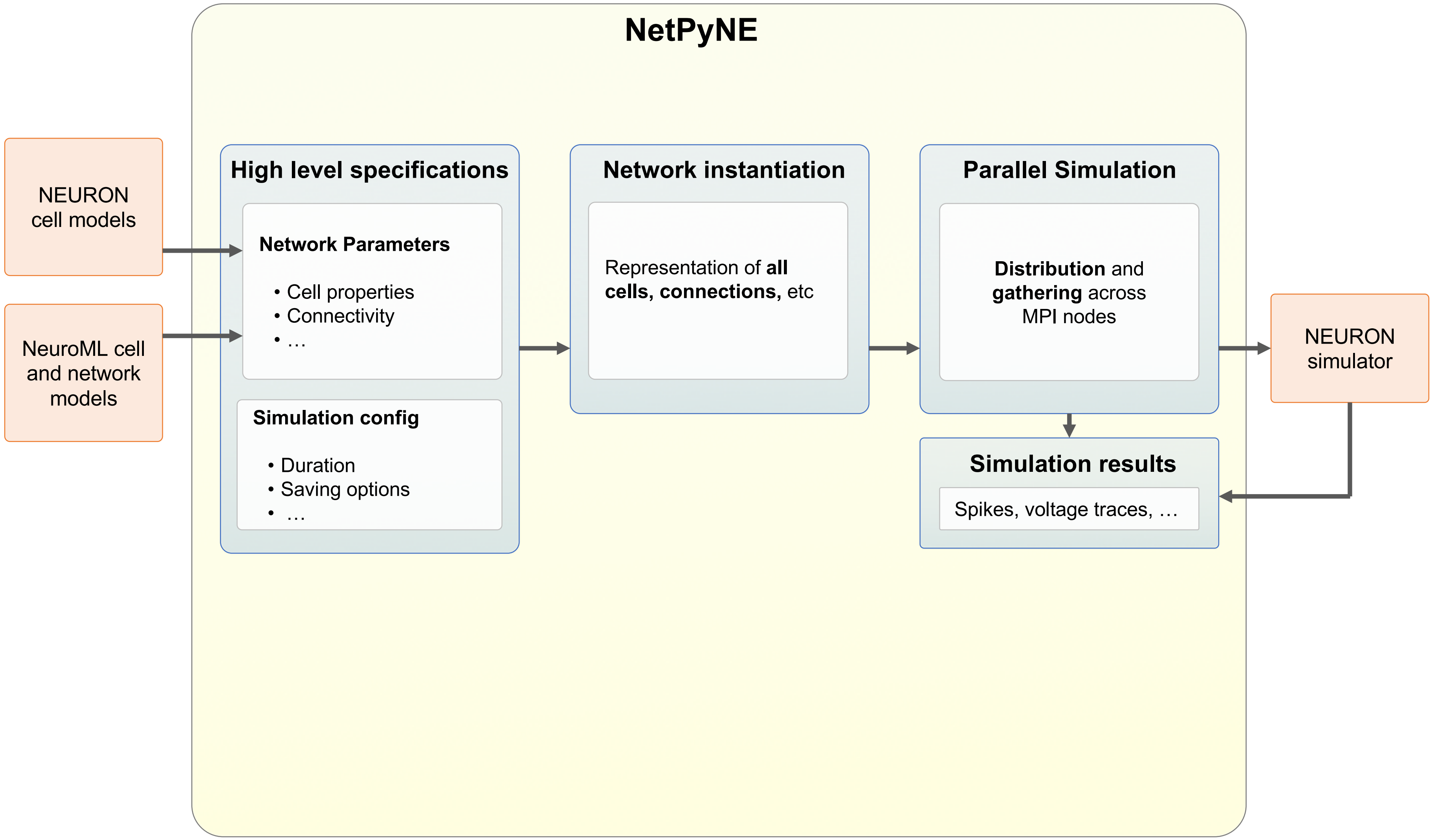
NetPyNE: Motivation

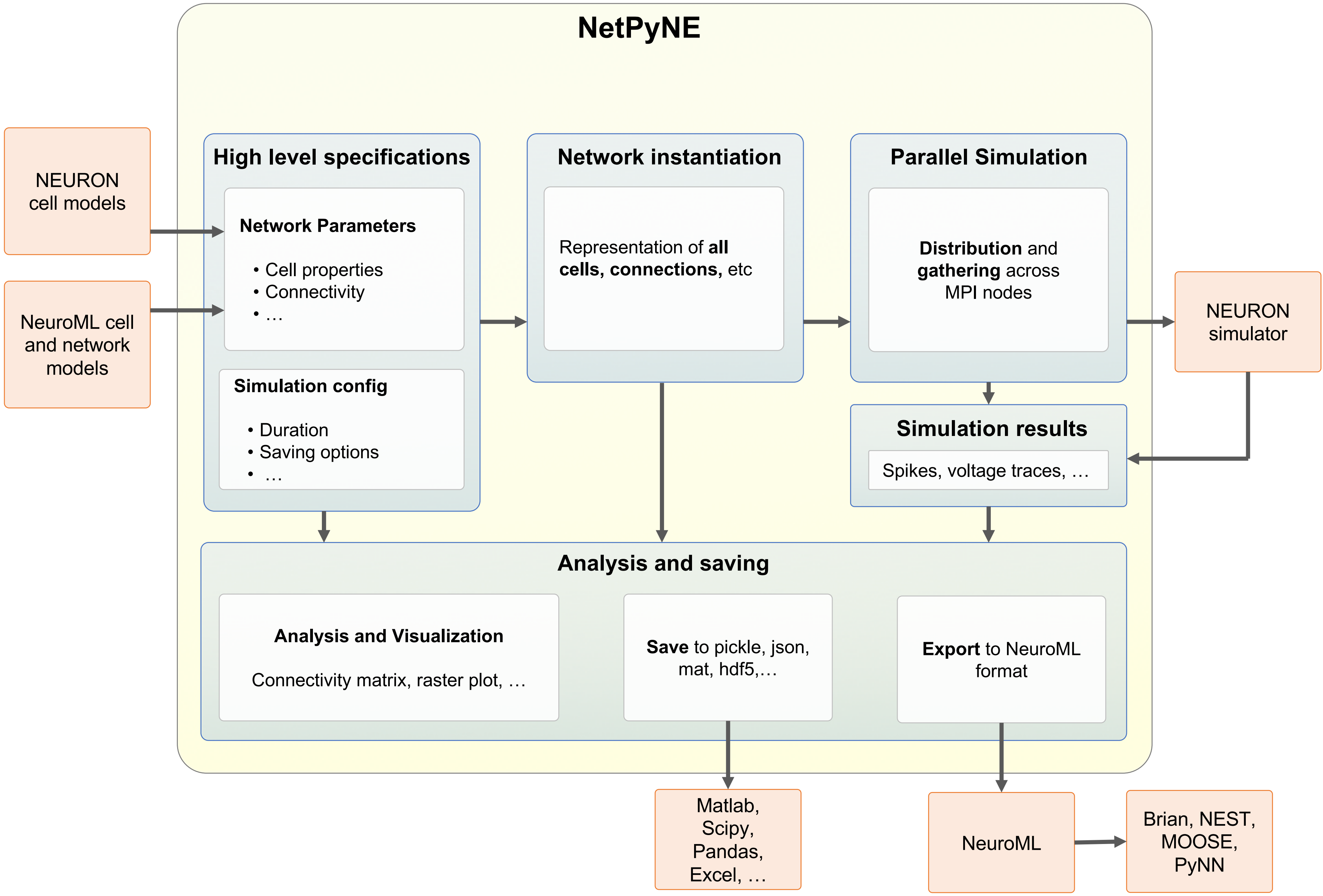
- Facilitate model parallelization (HPCs)
- Batch parameter exploration/optimization

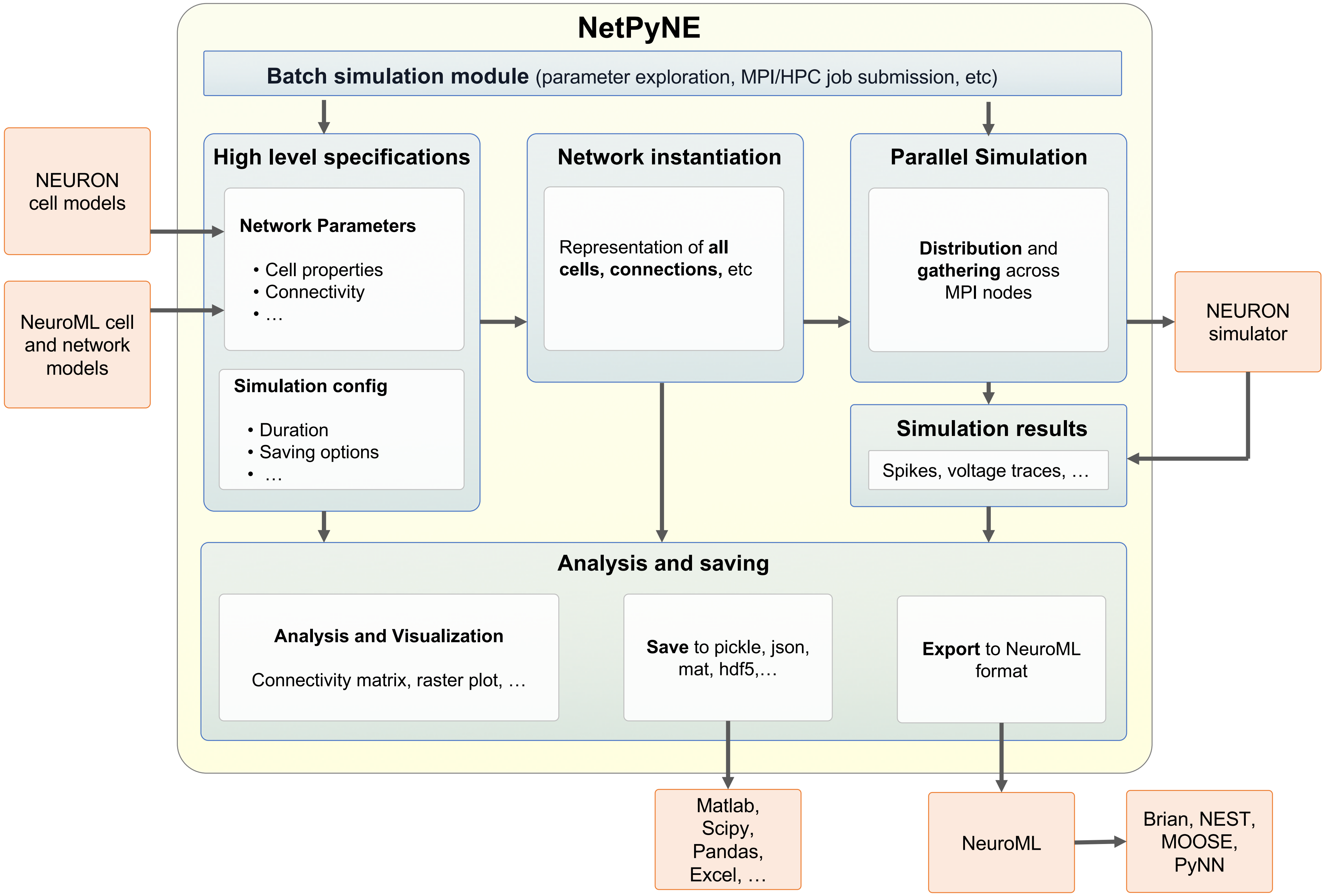





NetPyNE
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …

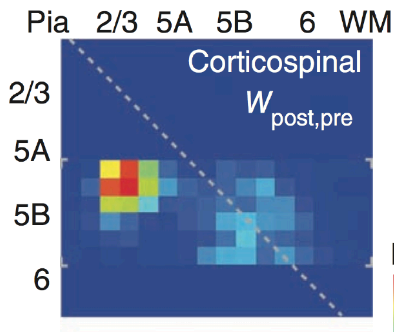

NetPyNE
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …
Network instantiation
Representation of **all cells, connections,** etc
Parallel Simulation
Distribution and **gathering**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


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

NetPyNE: 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.

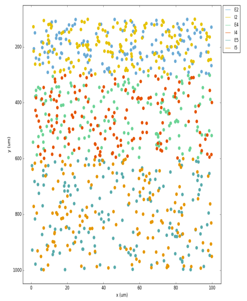


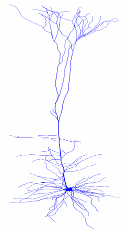
```
connParams['bin3->CSP'] = {
 'preConds':     {'y': [100, 150]},
 'postConds':    {'pop': 'CSP'},
 'probability': 0.15,
 'weight':       0.4,
 'delay':        5,
 'synMech':      'AMPA'}
```

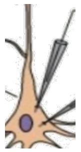
NetPyNE facilitates building models based on experimental data

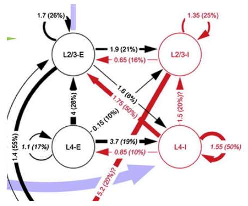
NetPyNE: High level specifications

- 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, ...



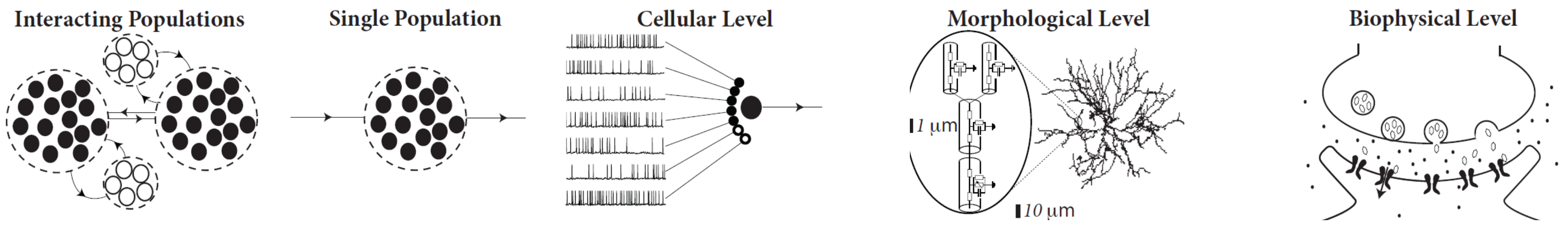


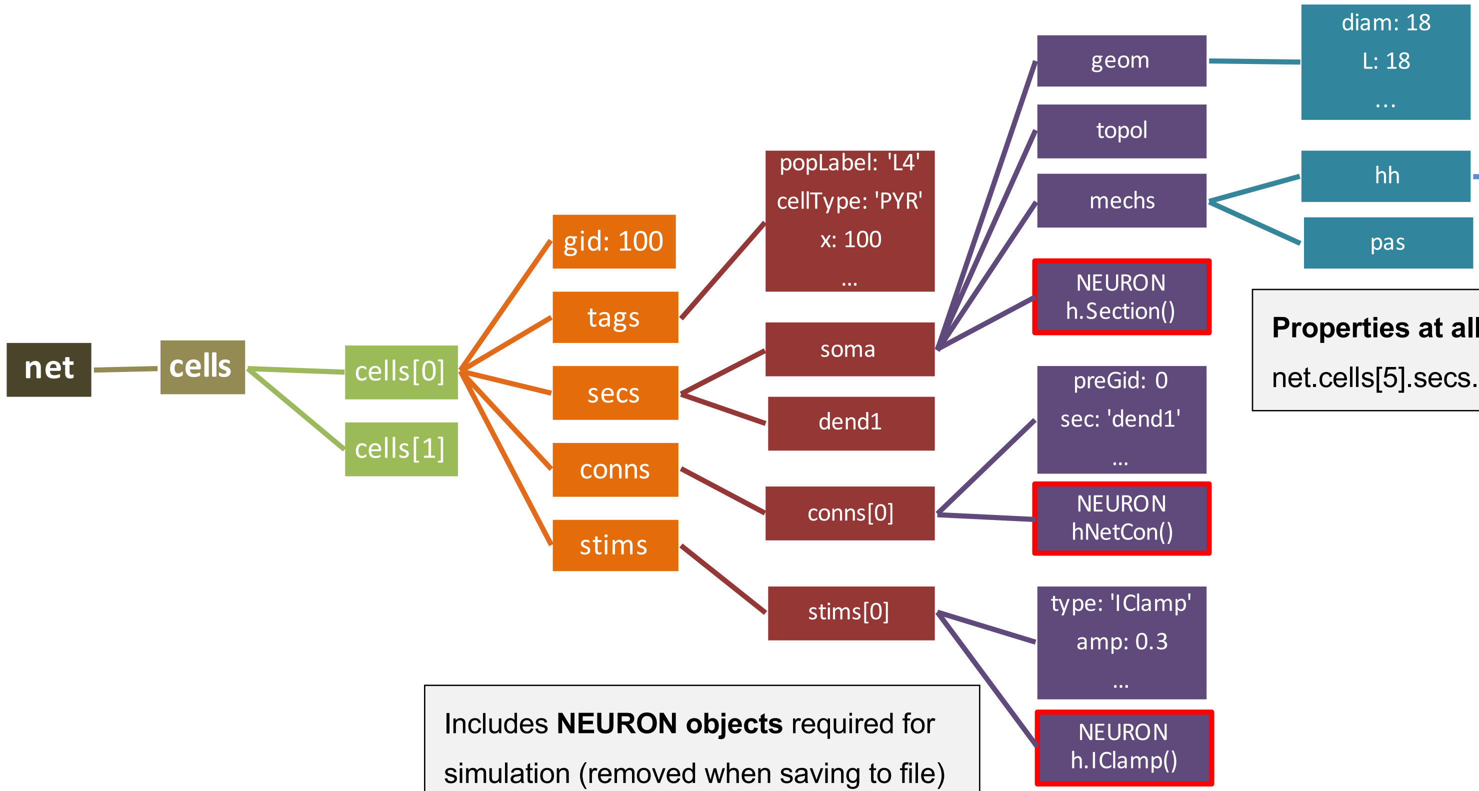




NetPyNE: Network Instantiation

- Network is created as Python-based **standardized hierarchical data structure.**





Properties at all scales easy to access:

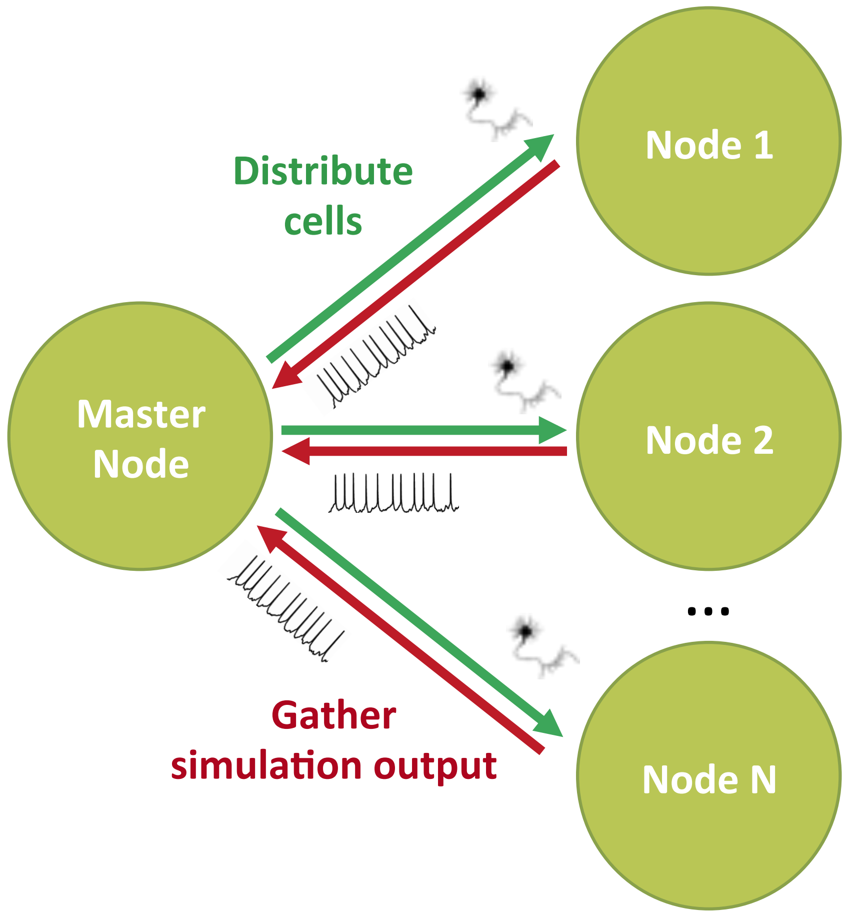
net.cells[5].secs.soma.mechs.hh.gnabar

Includes **NEURON objects** required for simulation (removed when saving to file)

NetPyNE: Parallel Simulation

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

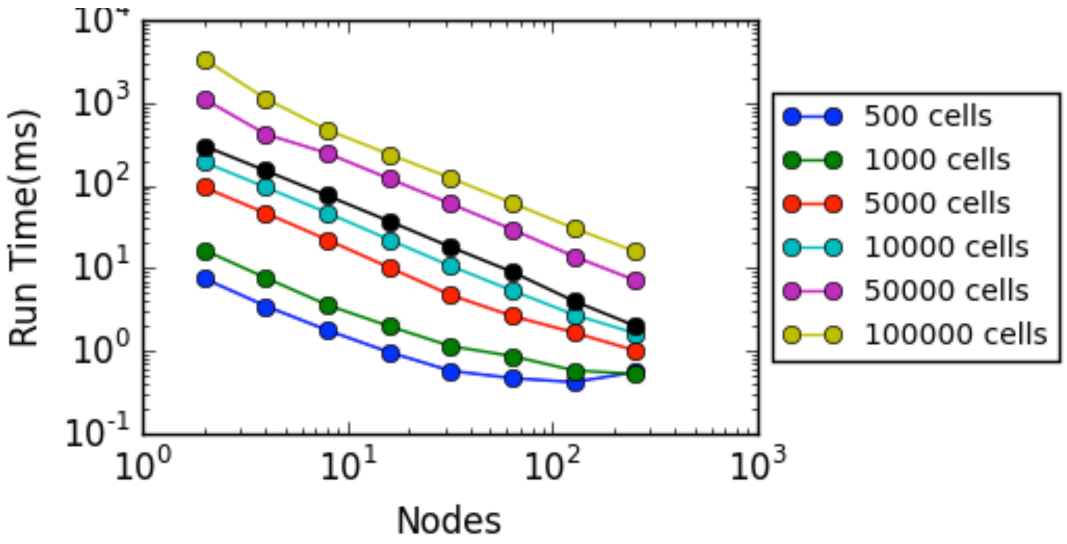
- Takes care of balanced **distribution** of cells and **gathering** of simulation output from nodes.



NetPyNE: Parallel Simulation

- NetPyNE available on **the Neuroscience Gateway (NSG)** supercomputing platform.

NEUROSCIENCE GATEWAY
A Portal for Computational Neuroscience



Simulation **run time** as a function of number of cells and number of nodes *(Neural Comput, 2016).*

Results obtained using **NetPyNE on NSG**.

NetPyNE: Batch Parallel Simulations

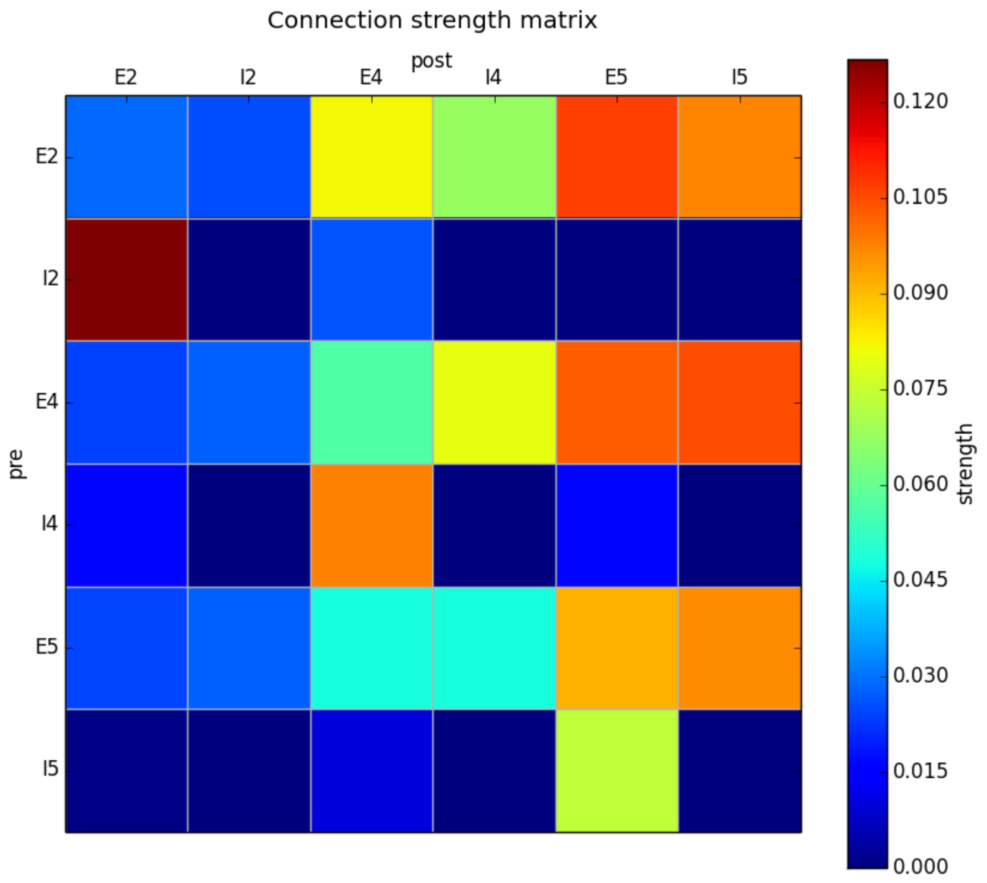
- **Easy specification** of parameters and range of values to explore in batch simulations.

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

SDSC SAN DIEGO SUPERCOMPUTER CENTER

NetPyNE: Analysis

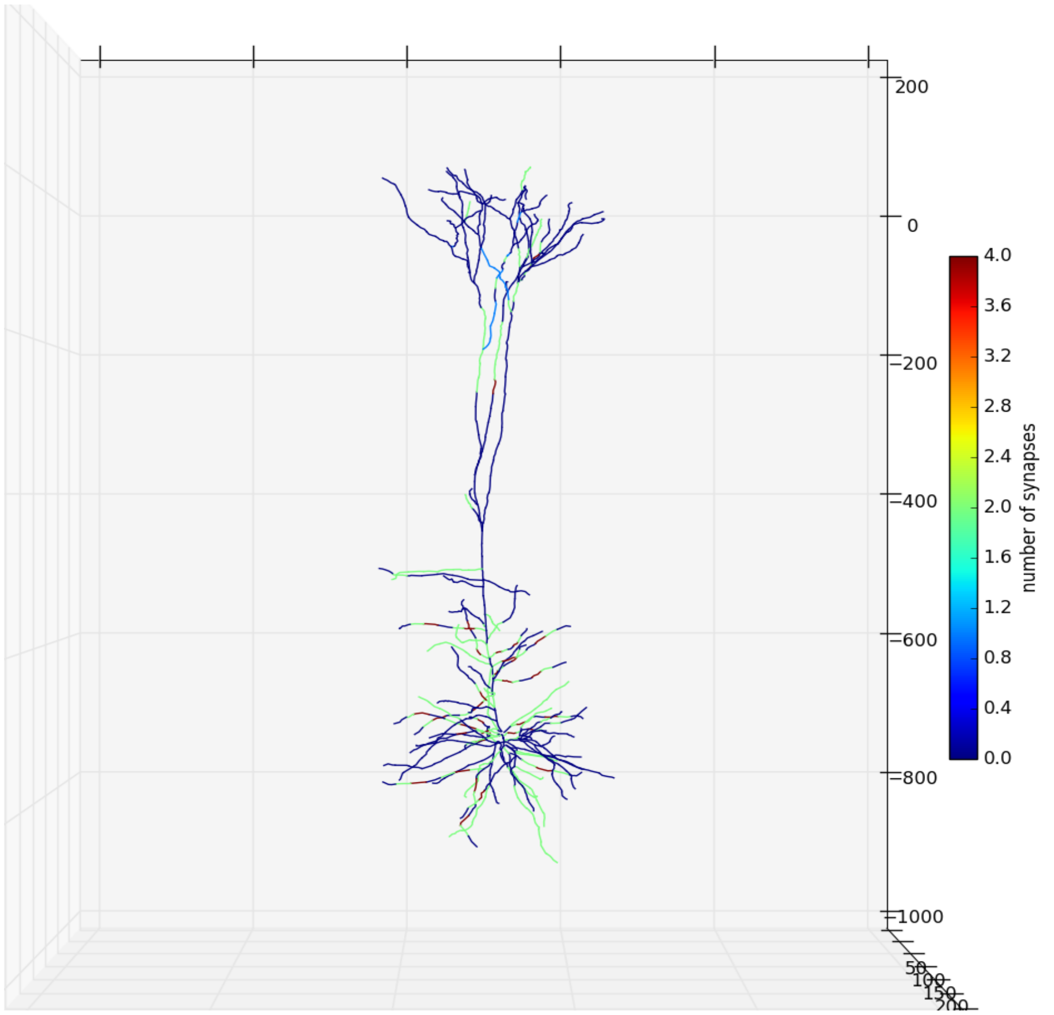
- **Connectivity matrix** at cell or population level (weights, num connections, probability,...)



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

NetPyNE: Analysis

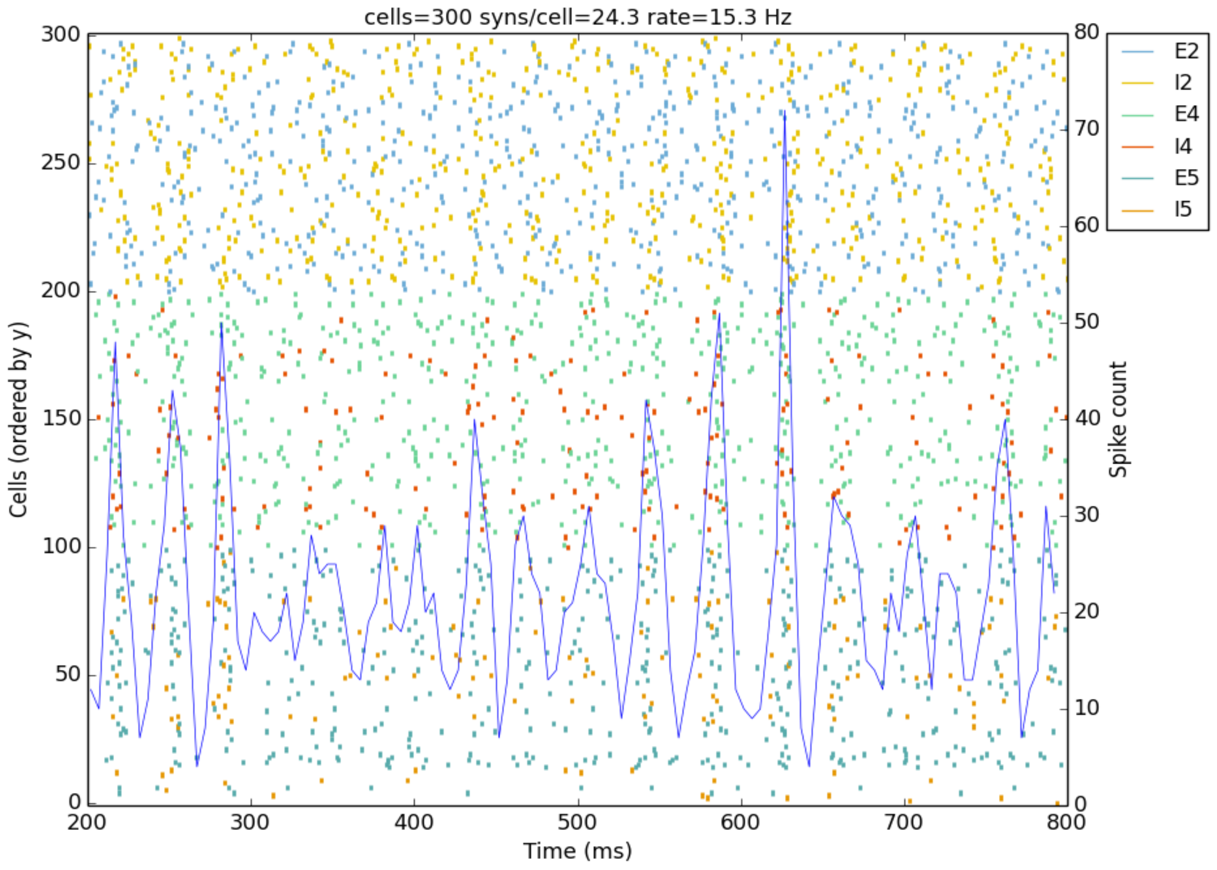
- 3D **cell shape plot**
- Option to include **color-coded variables** (eg, num of synapses)



`plotShape(…)`

NetPyNE: Analysis

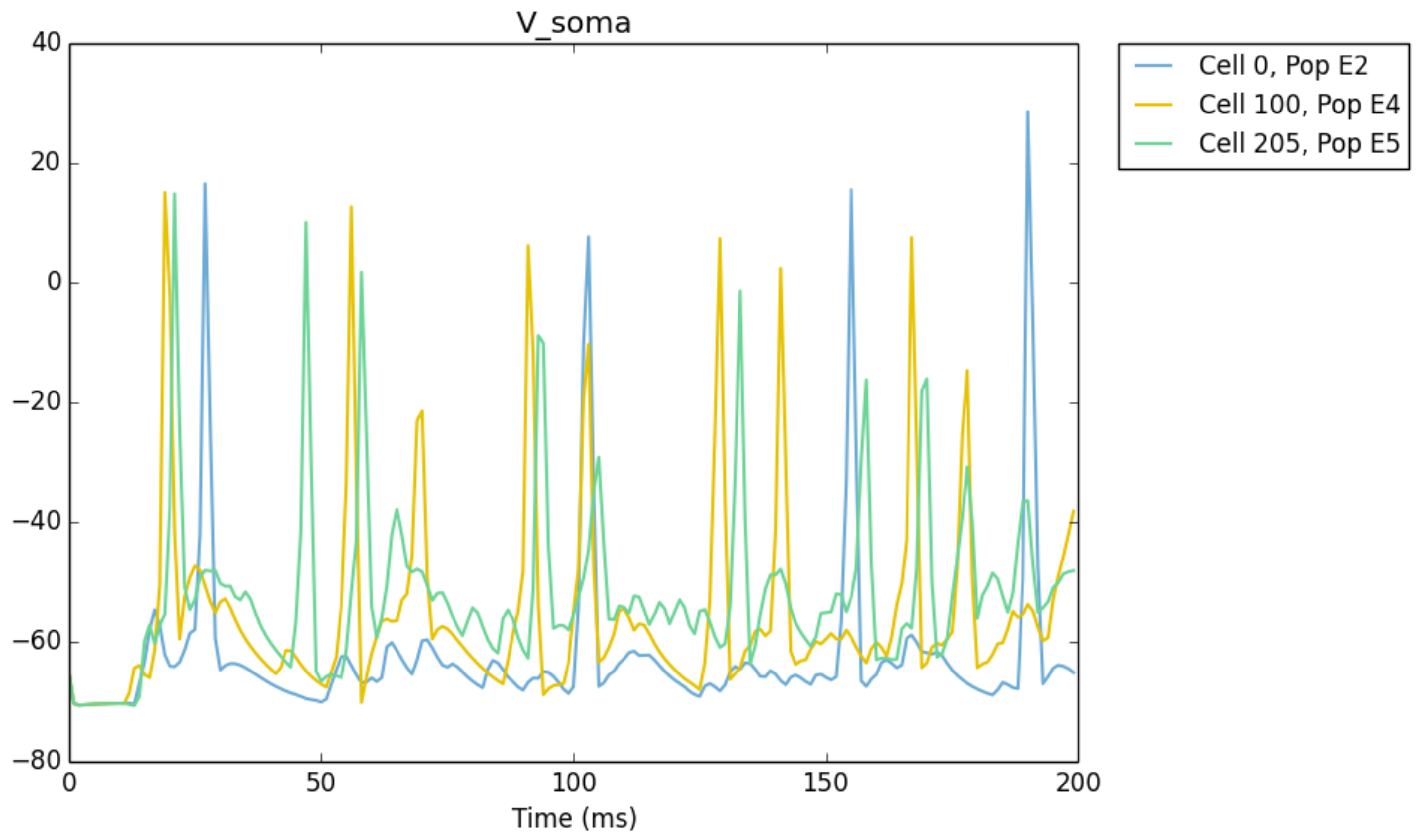
- Easy-to-use functions for **analysis and plotting** of network and simulation output
  - **Raster plot** of any subset of cells
  - **Spike histogram** of populations or subsets of cells



```
plotRaster(include=['allCells'], timeRange=[200,800], orderBy='y',
orderInverse=True, spikeHist='overlay', spikeHistBin=5)
```

NetPyNE: Analysis

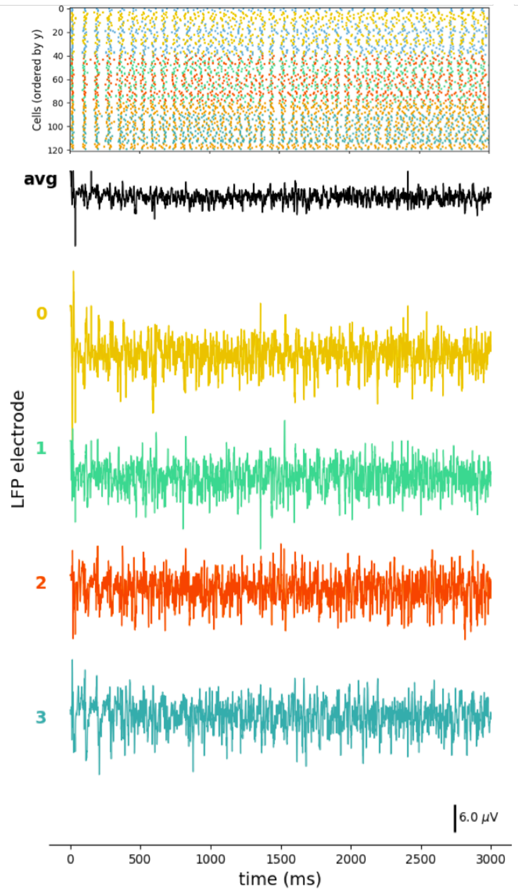
- Intrinsic cell variables (voltages, currents, conductance) **trace plots**

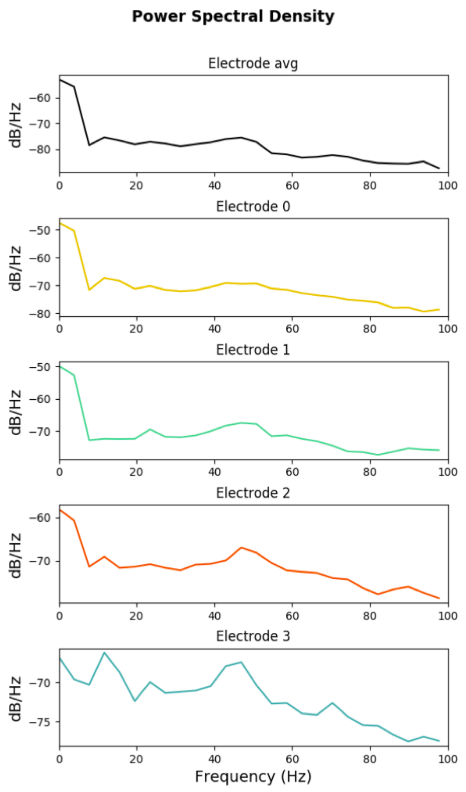


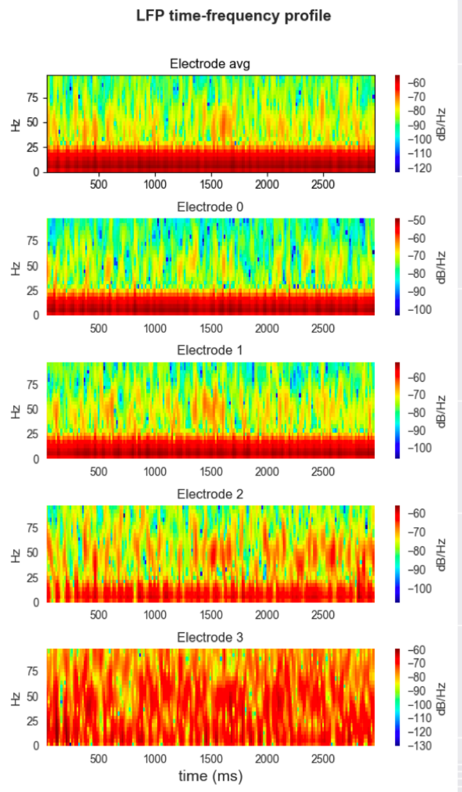
```
plotTraces(include=[('E2',0), ('E4',0), ('E5',5)],
timeRange=[0,200], overlay=True, oneFigPer='trace')
```

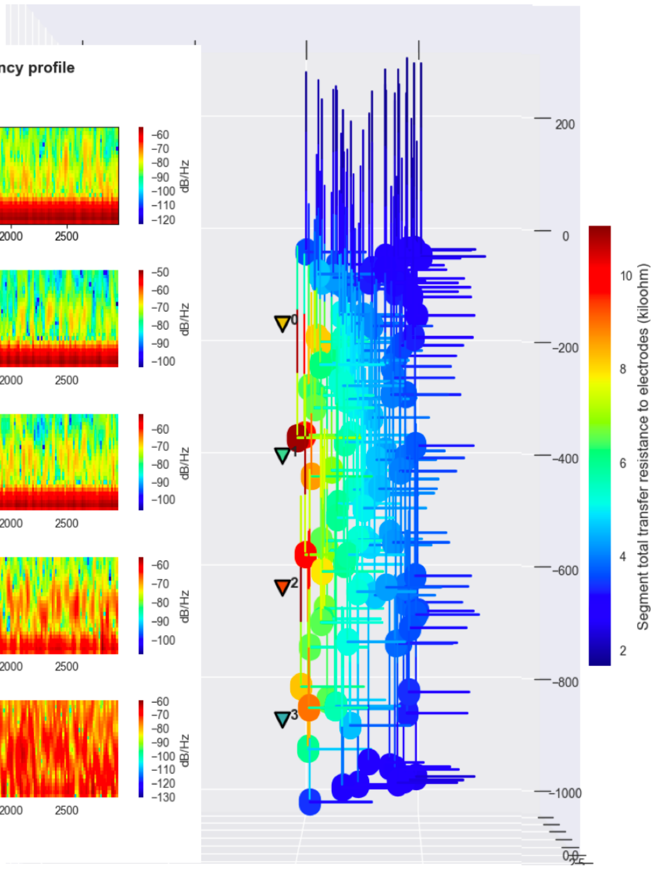
NetPyNE: Analysis

- **LFP** time-series, PSD, spectrogram and electrode locations





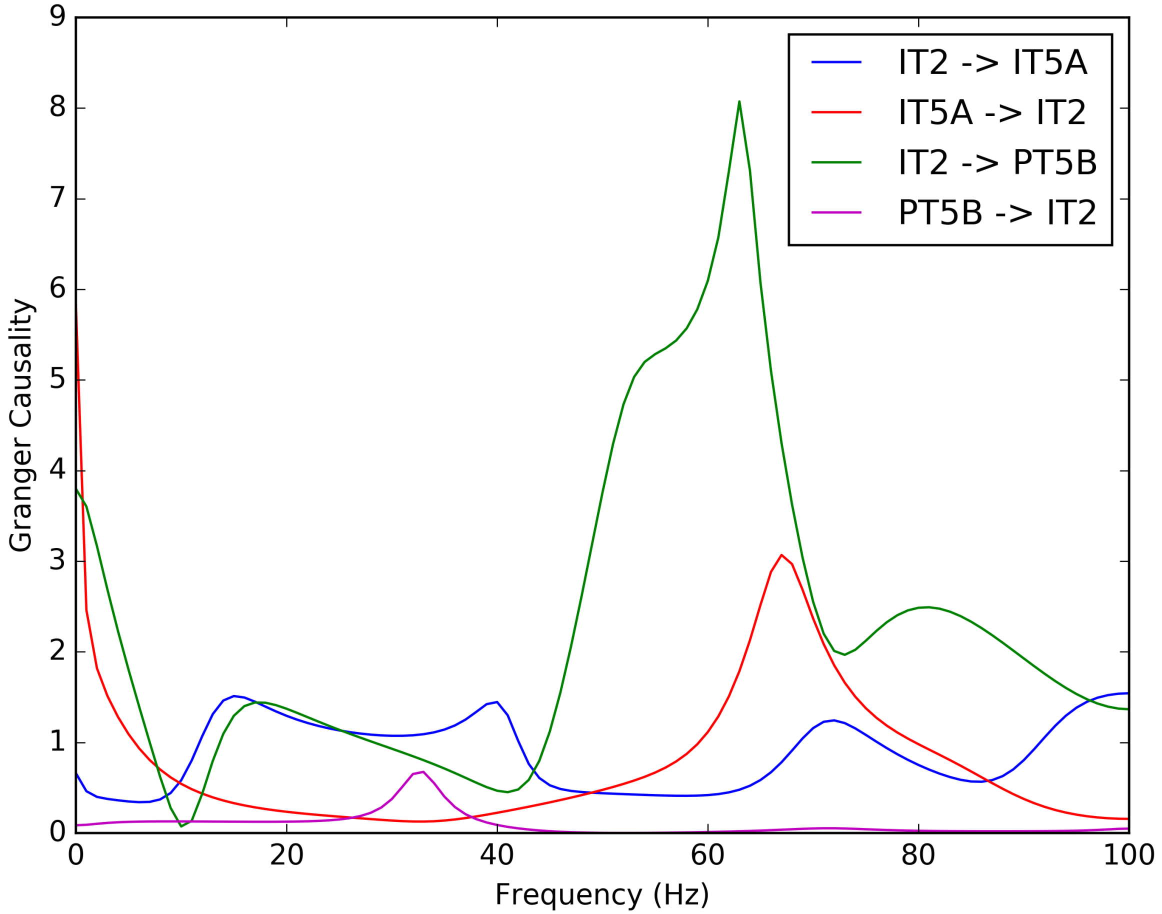




`plotLFP(…)`

NetPyNE: Analysis

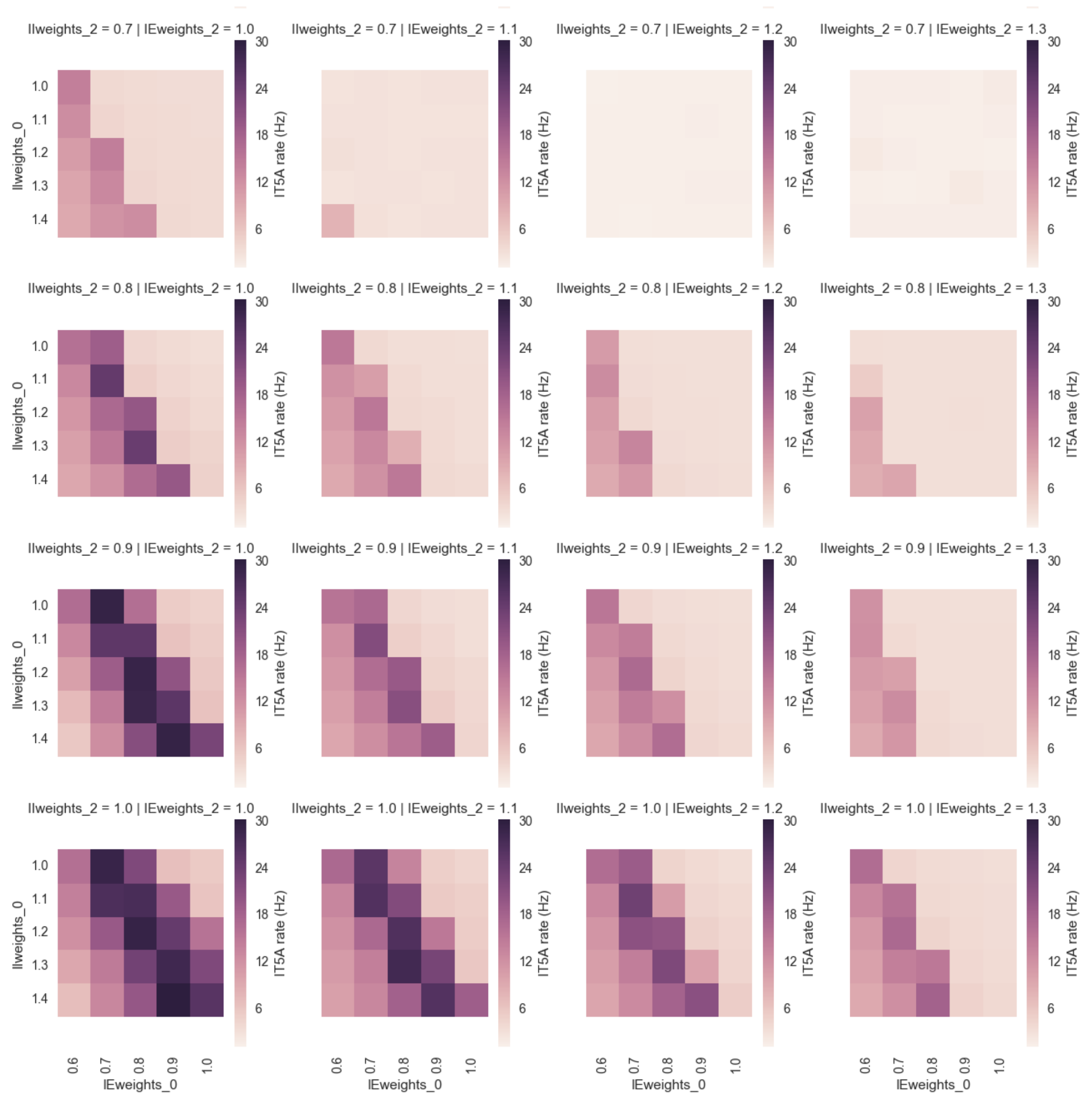
- Spectral **Granger causality**
- Normalized **transfer entropy**



`plotGranger(…)`

NetPyNE: Analysis

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



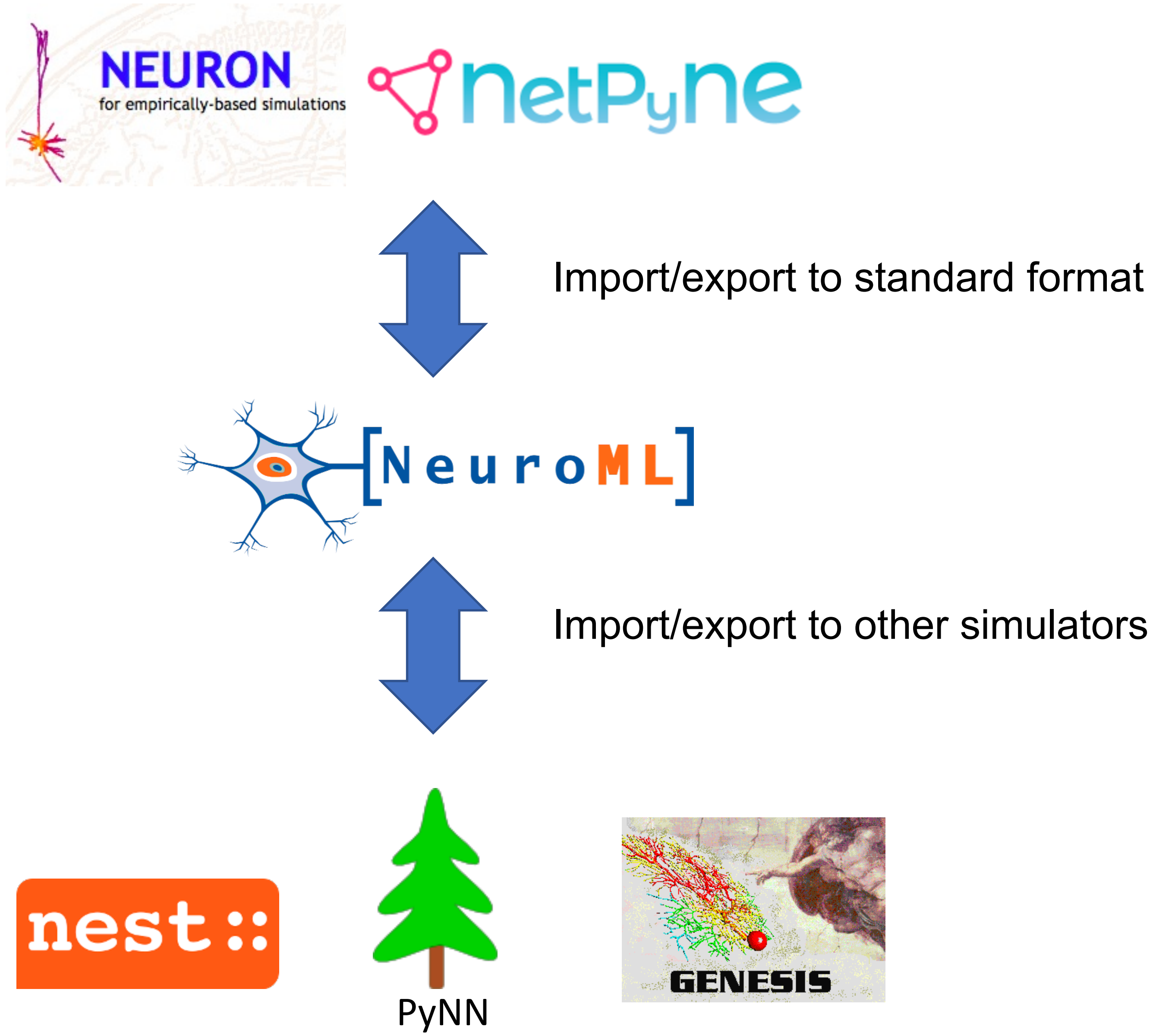
NetPyNE: Data saving and exporting

- ❑ **Save and load** high-level specifications, network instance, simulation config and/or simulation results.

- ❑ **Multiple formats** supported: pickle, Matlab, JSON, CSV, HDF5

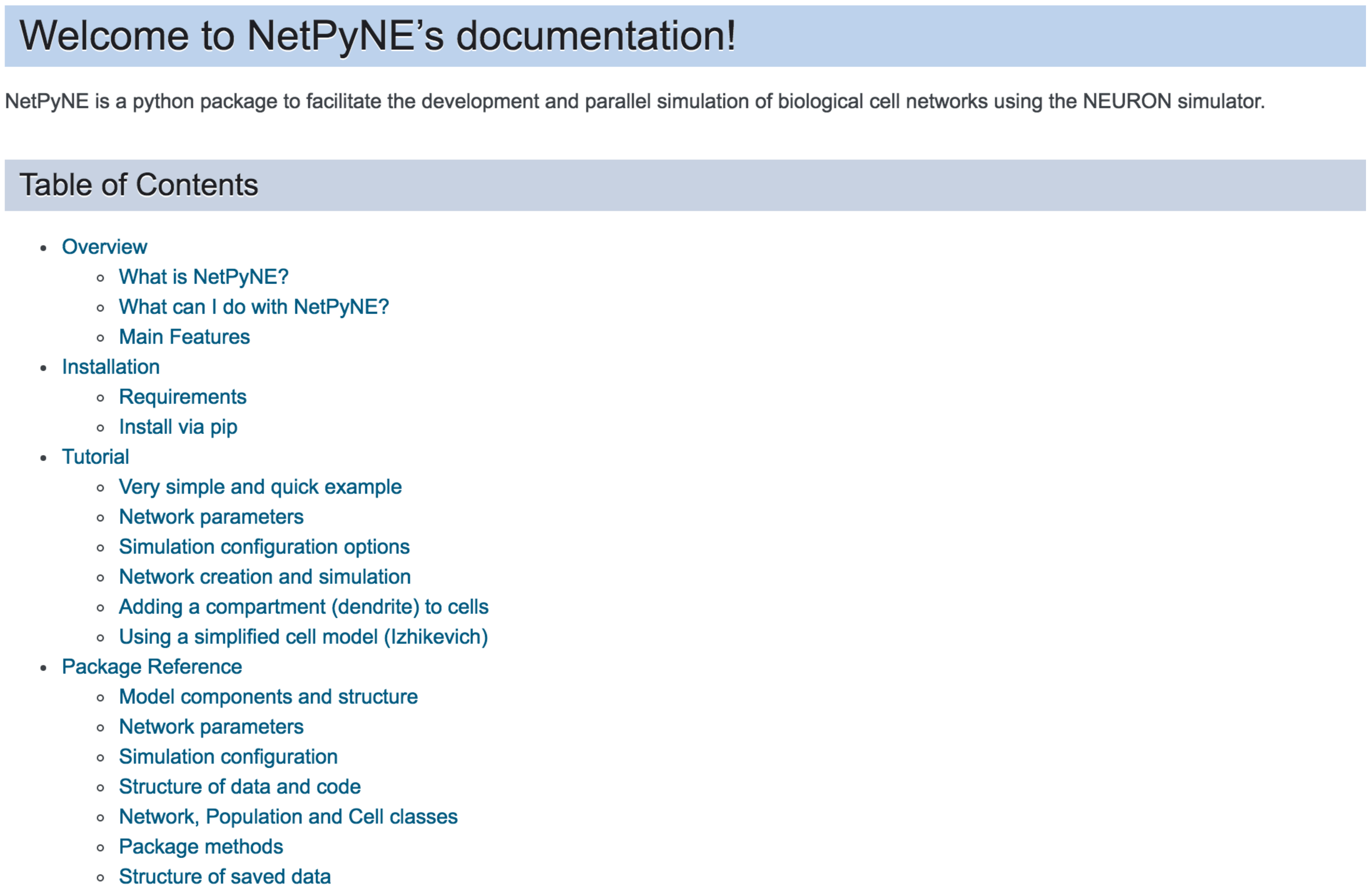
- ❑ **Export/import** network instance to/from **NeuroML**, the standard format for neural models.

NetPyNE: Data saving and exporting


NEURON
for empirically-based simulations
NetPyNE
Import/export to standard format
NeuroML
Import/export to other simulators
nest::
PyNN
GENESIS

NetPyNE: Documentation and Tutorials

www.netpyne.org

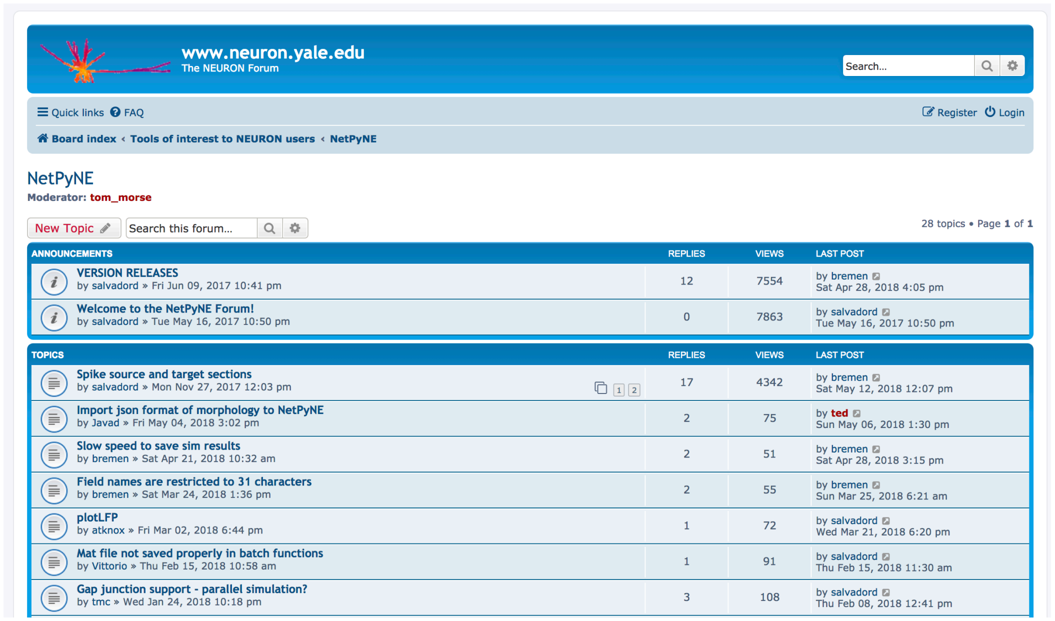


Welcome to NetPyNE's documentation!

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



www.neuron.yale.edu
The NEURON Forum

Quick links · FAQ · Register · Login

Board index ‹ Tools of interest to NEURON users ‹ NetPyNE

NetPyNE
Moderator: tom_morse

New Topic · Search this forum...

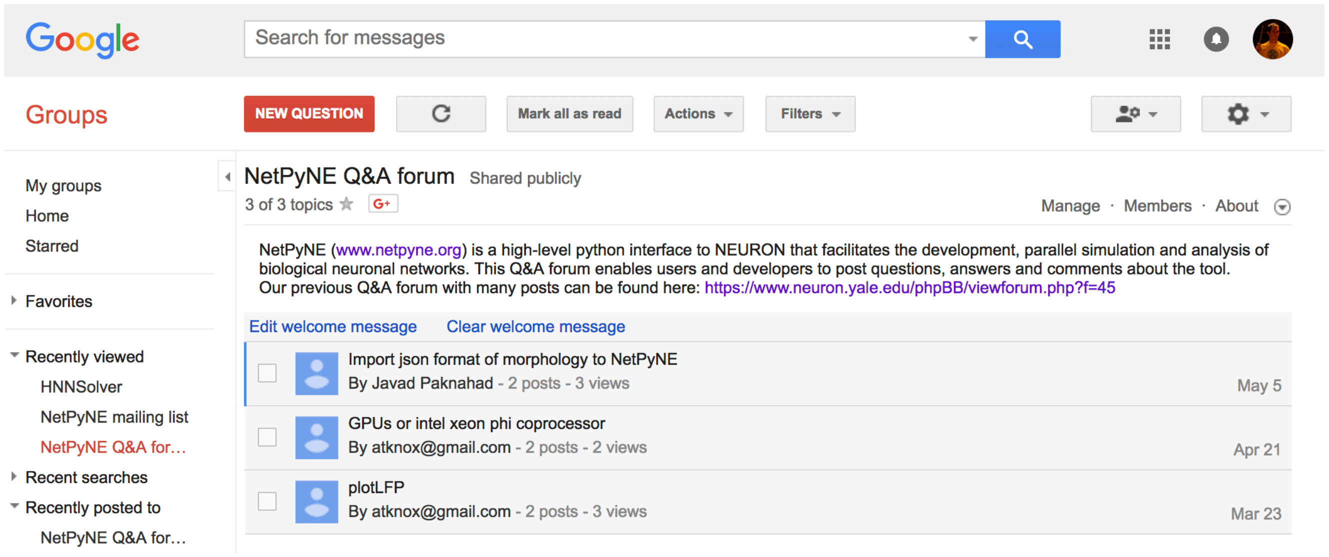
28 topics • Page 1 of 1

ANNOUNCEMENTS	REPLIES	VIEWS	LAST POST
VERSION RELEASES by salvadord » Fri Jun 09, 2017 10:41 pm	12	7554	by bremen Sat Apr 28, 2018 4:05 pm
Welcome to the NetPyNE Forum! by salvadord » Tue May 16, 2017 10:50 pm	0	7863	by salvadord Tue May 16, 2017 10:50 pm

TOPICS	REPLIES	VIEWS	LAST POST
Spike source and target sections by salvadord » Mon Nov 27, 2017 12:03 pm 1 2	17	4342	by bremen Sat May 12, 2018 12:07 pm
Import json format of morphology to NetPyNE by Javad » Fri May 04, 2018 3:02 pm	2	75	by ted Sun May 06, 2018 1:30 pm
Slow speed to save sim results by bremen » Sat Apr 21, 2018 10:32 am	2	51	by bremen Sat Apr 28, 2018 3:15 pm
Field names are restricted to 31 characters by bremen » Sat Mar 24, 2018 1:36 pm	2	55	by bremen Sun Mar 25, 2018 6:21 am
plotLFP by atknox » Fri Mar 02, 2018 6:44 pm	1	72	by salvadord Wed Mar 21, 2018 6:20 pm
Mat file not saved properly in batch functions by Vittorio » Thu Feb 15, 2018 10:58 am	1	91	by salvadord Thu Feb 15, 2018 11:30 am
Gap junction support - parallel simulation? by tmc » Wed Jan 24, 2018 10:18 pm	3	108	by salvadord Thu Feb 08, 2018 12:41 pm

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

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



Google · Search for messages

Groups · NEW QUESTION · Mark all as read · Actions · Filters

My groups
Home
Starred
Favorites
Recently viewed
HNNSolver
NetPyNE mailing list
NetPyNE Q&A for...
Recent searches
Recently posted to
NetPyNE Q&A for...

NetPyNE Q&A forum Shared publicly
3 of 3 topics · Manage · Members · About

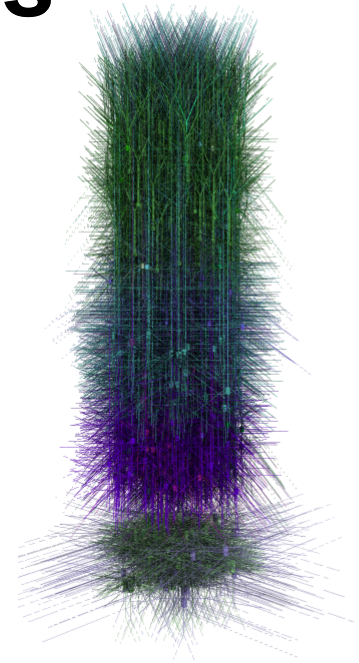
NetPyNE (www.netpyne.org) is a high-level python interface to NEURON that facilitates the development, parallel simulation and analysis of biological neuronal networks. This Q&A forum enables users and developers to post questions, answers and comments about the tool.
Our previous Q&A forum with many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

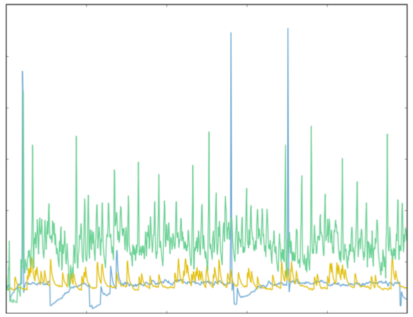
Edit welcome message · Clear welcome message

Import json format of morphology to NetPyNE By Javad Paknahad - 2 posts - 3 views	May 5
GPUs or intel xeon phi coprocessor By atknox@gmail.com - 2 posts - 2 views	Apr 21
plotLFP By atknox@gmail.com - 2 posts - 3 views	Mar 23

NetPyNE: Existing models

- Other models in progress:
  - Traub **thalamocortical** network (Padraig Gleeson, UCL)
  - Hippocampus **CA3** (Ben Tessler, SUNY DMC)
  - **Ischemia** in cortical network (Alex Seidenstein, SUNY DMC)
  - **STDP** in biophysically detailed networks (Anatoly Buchin, Allen Brain)
  - **Basal Ganglia** network (Lucas, UCD)
  - **LFP** oscillations (Christian Fink, Ohio Wesleyan)
  - **Dendritic** computations (Birgit Kriener, Oslo)
  - Thalamocortical **epilepsy** network (Andrew Knox, Cincinatti Hospital)
  - **V1** network with Allen Brain cells (SUNY DMC)
  - **Schizophrenia** in cortical network (Cristoph Metzner, Hertfordshire)
  - **Spinal cord** circuits (Vittorio Caggiano, IBM Watson)

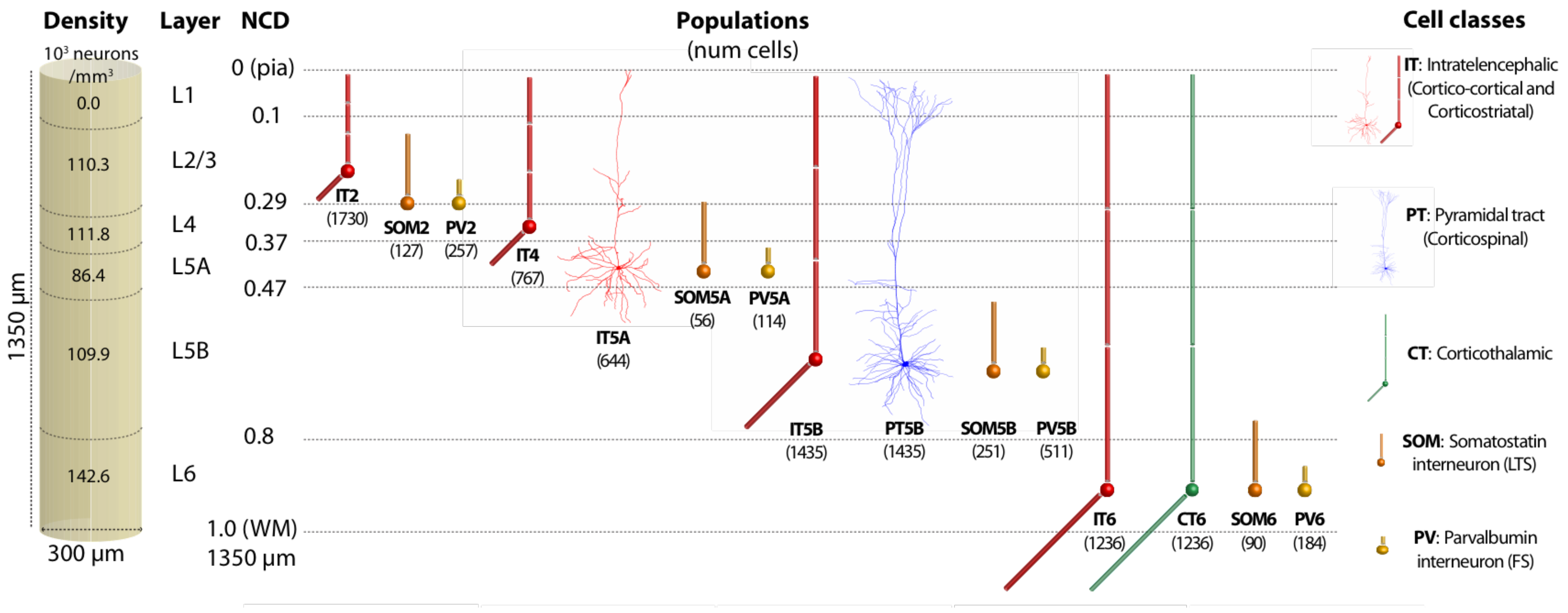




  - Full list of 43 models: https://drive.google.com/open?id=1bkWHakgZoEkYIkzrAS8sIKCvO5PSuUXLLRjNdN2pseY

NetPyNE: M1 microcircuits

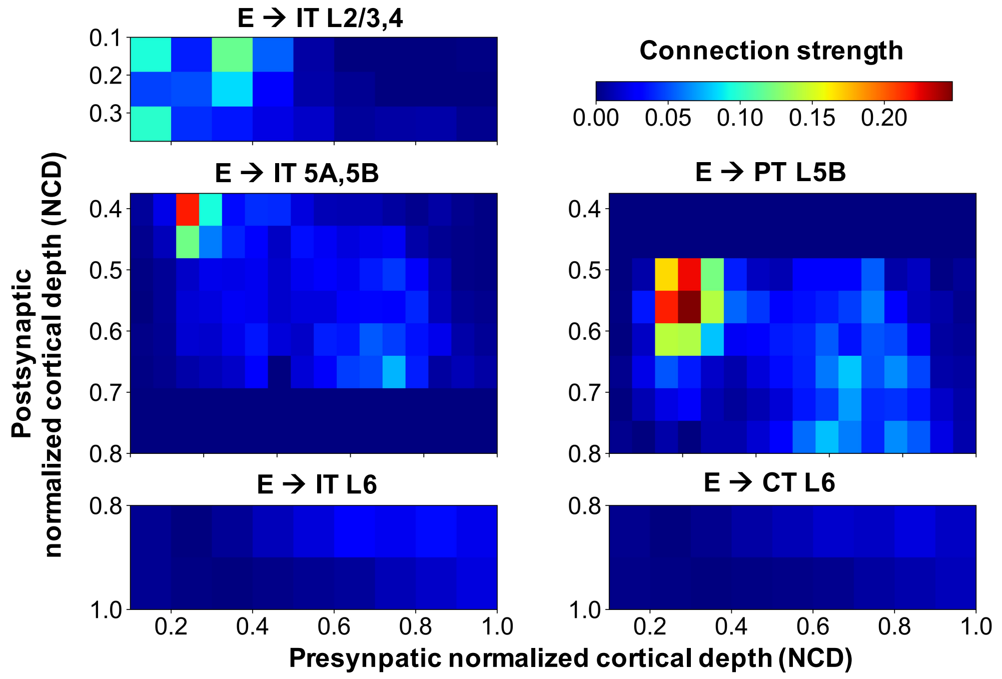
- Data-driven multiscale network model of **M1 microcircuits**



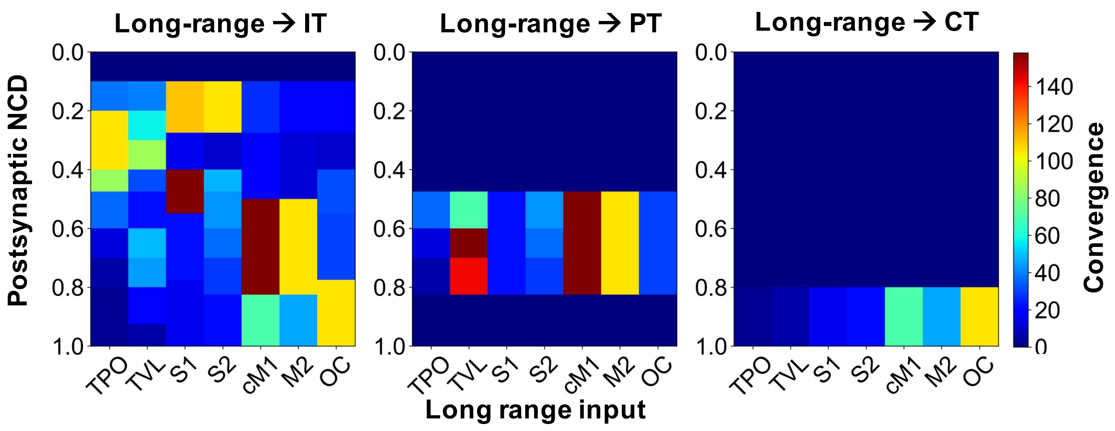
Mouse 6-layer M1 with **10,074 neurons** of 5 classes distributed in 15 populations;
Full scale cylindric volume of **300 µm** (diameter) x **1350 µm** (cortical depth)

NetPyNE: M1 microcircuits

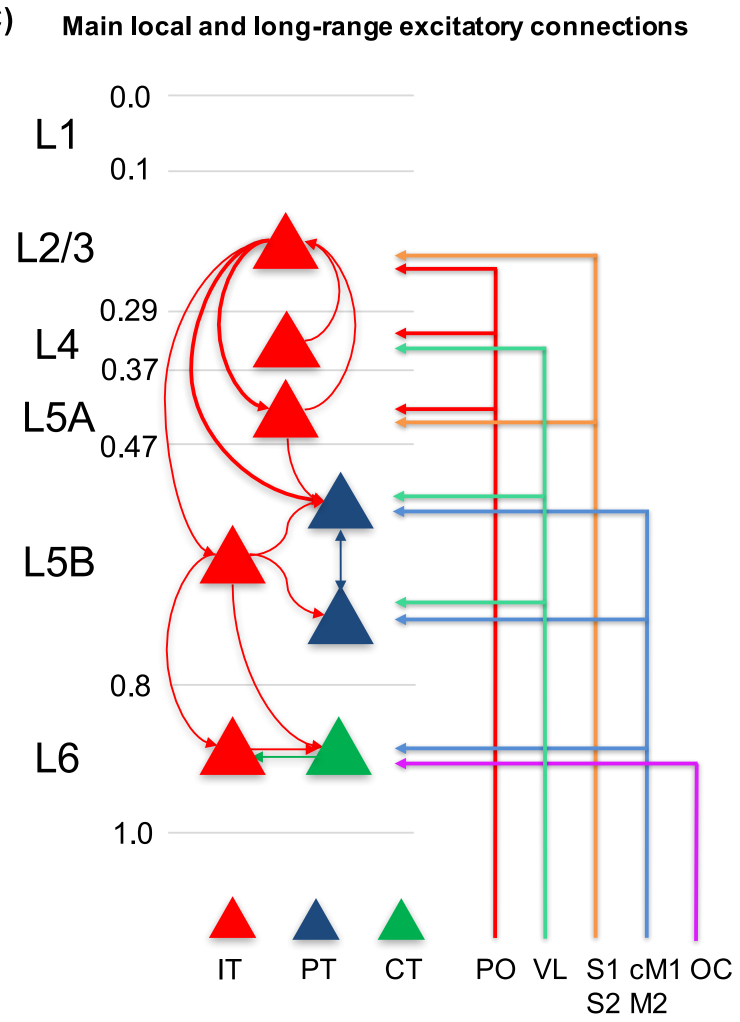
A)



B)



C)



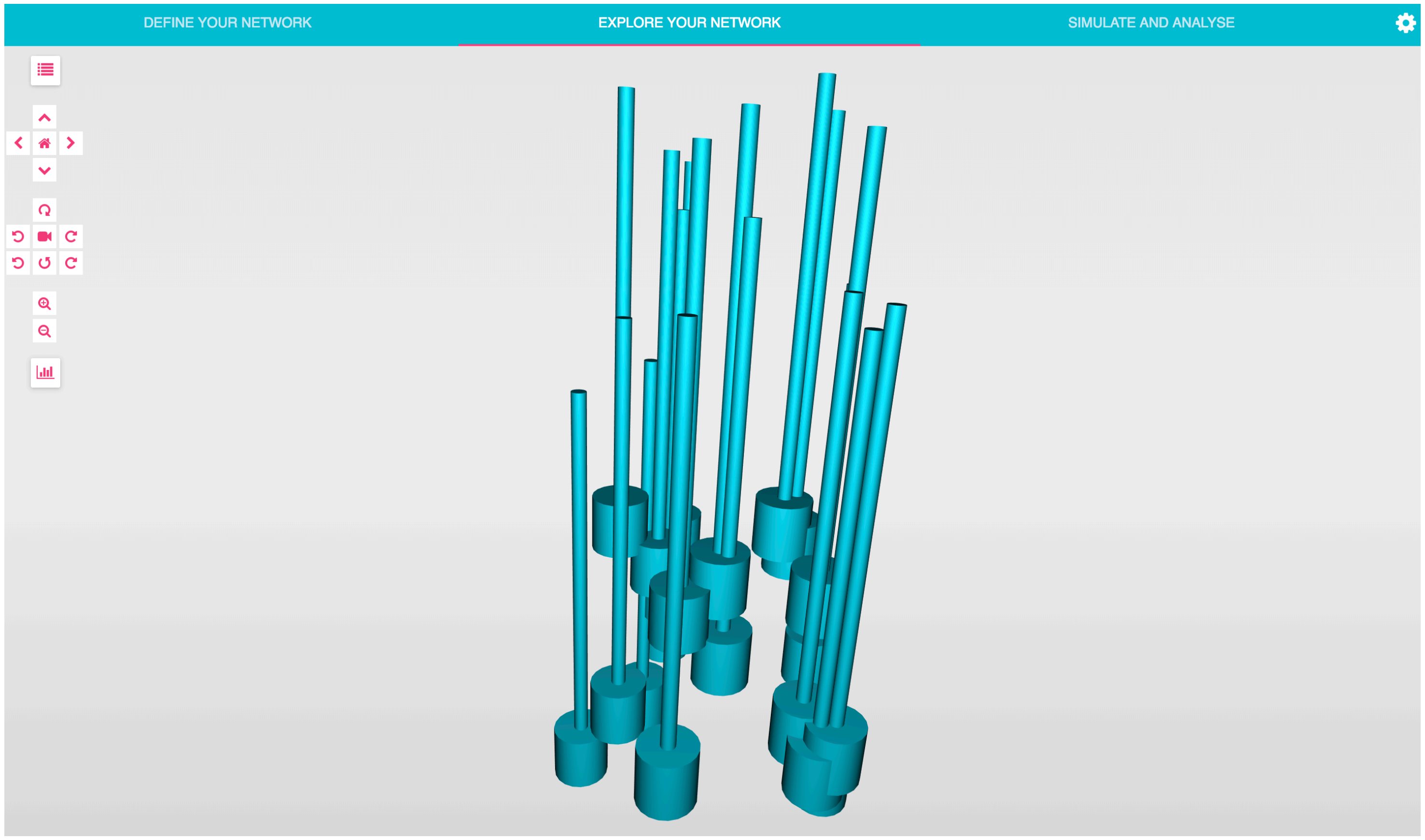
NetPyNE: Acknowledgments

- **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)
  - 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)

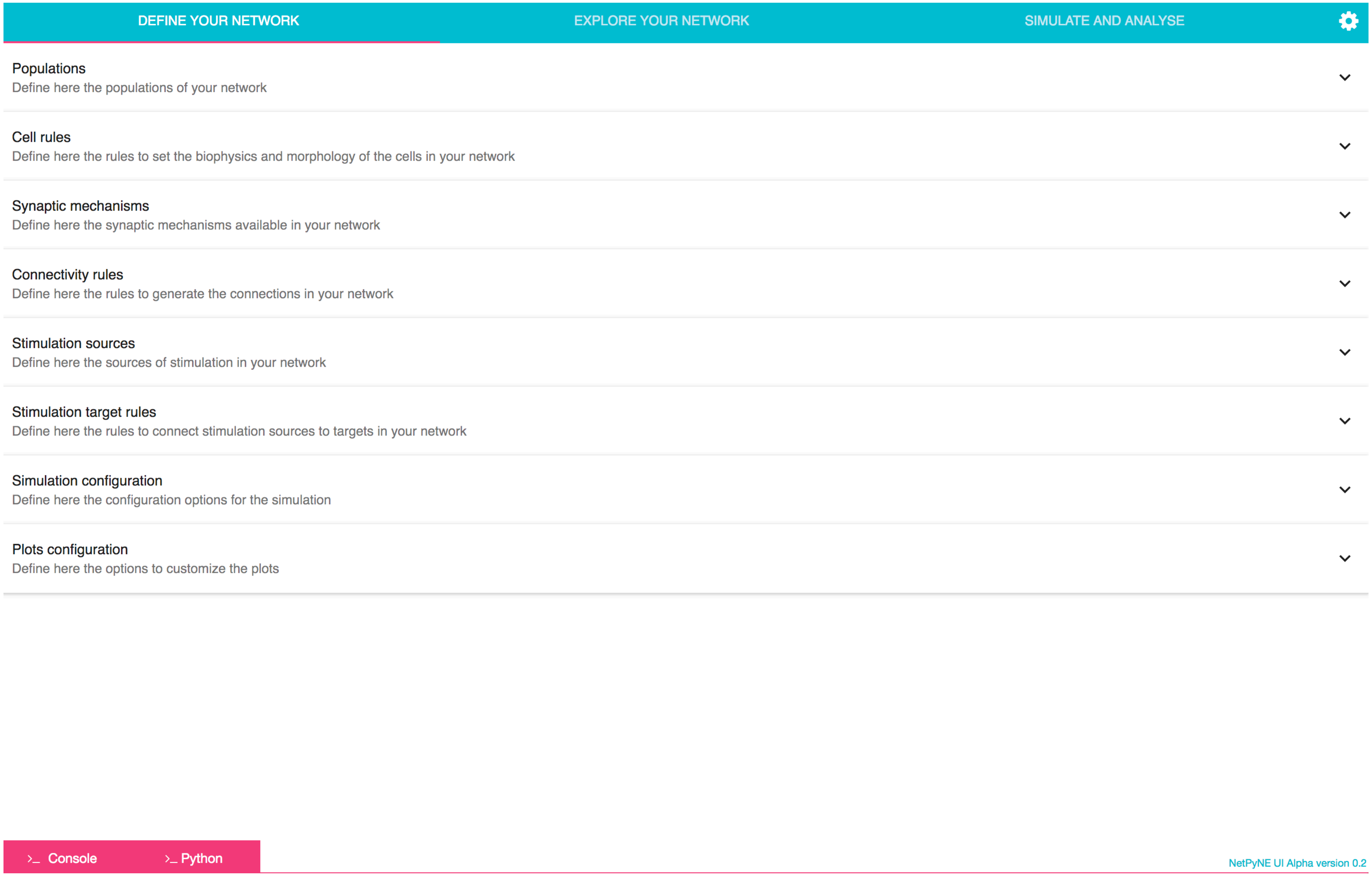
- **Funding:**
  - NIH Grant U01EB017695
  - NIH Grant R01EB022903
  - NIH Grant R01MH086638
  - NYS Grant DOH01-C32250GG-3450000

NetPyNE GUI Tutorial: Simple cell net



NetPyNE GUI Tutorial: Simple cell net

1) Open NetPyNE GUI on web browser



Populations
Define here the populations of your network

Cell rules
Define here the rules to set the biophysics and morphology of the cells in your network

Synaptic mechanisms
Define here the synaptic mechanisms available in your network

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

Stimulation sources
Define here the sources of stimulation in your network

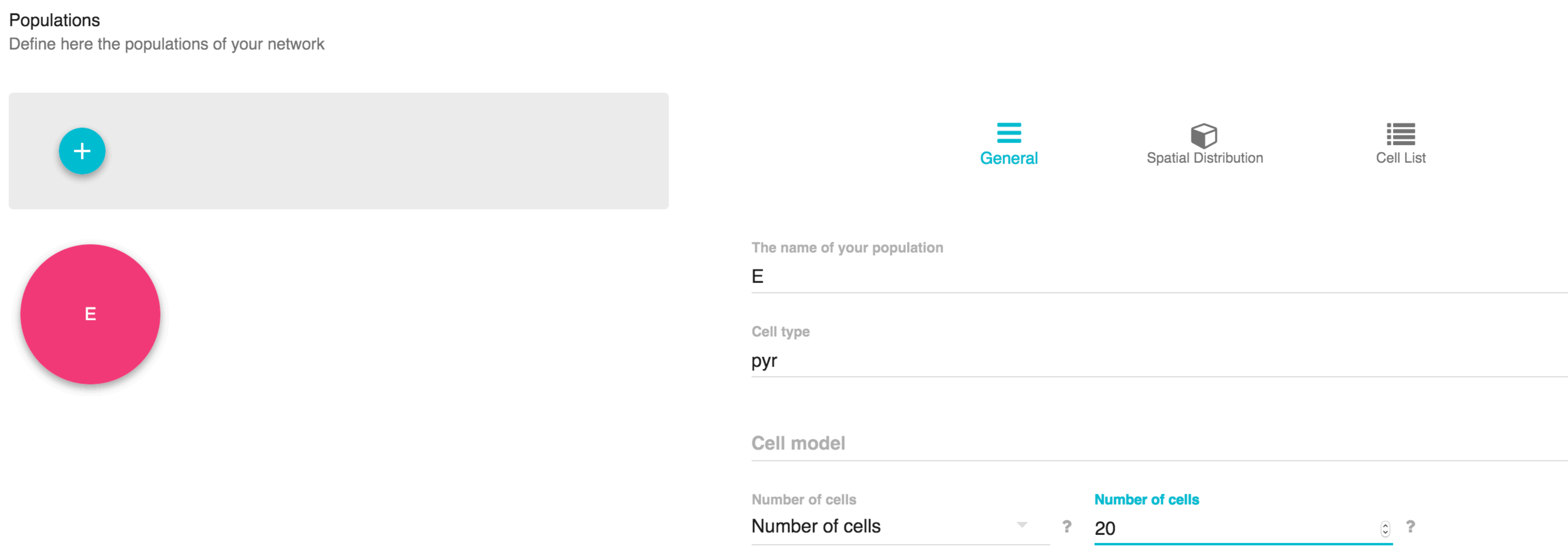
Stimulation target rules
Define here the rules to connect stimulation sources to targets in your network

Simulation configuration
Define here the configuration options for the simulation

Plots configuration
Define here the options to customize the plots

NetPyNE GUI Tutorial: Simple cell net

2) Add a population 'E' of 20 'pyr' cells



Populations

Define here the populations of your network

The name of your population
E

Cell type
pyr

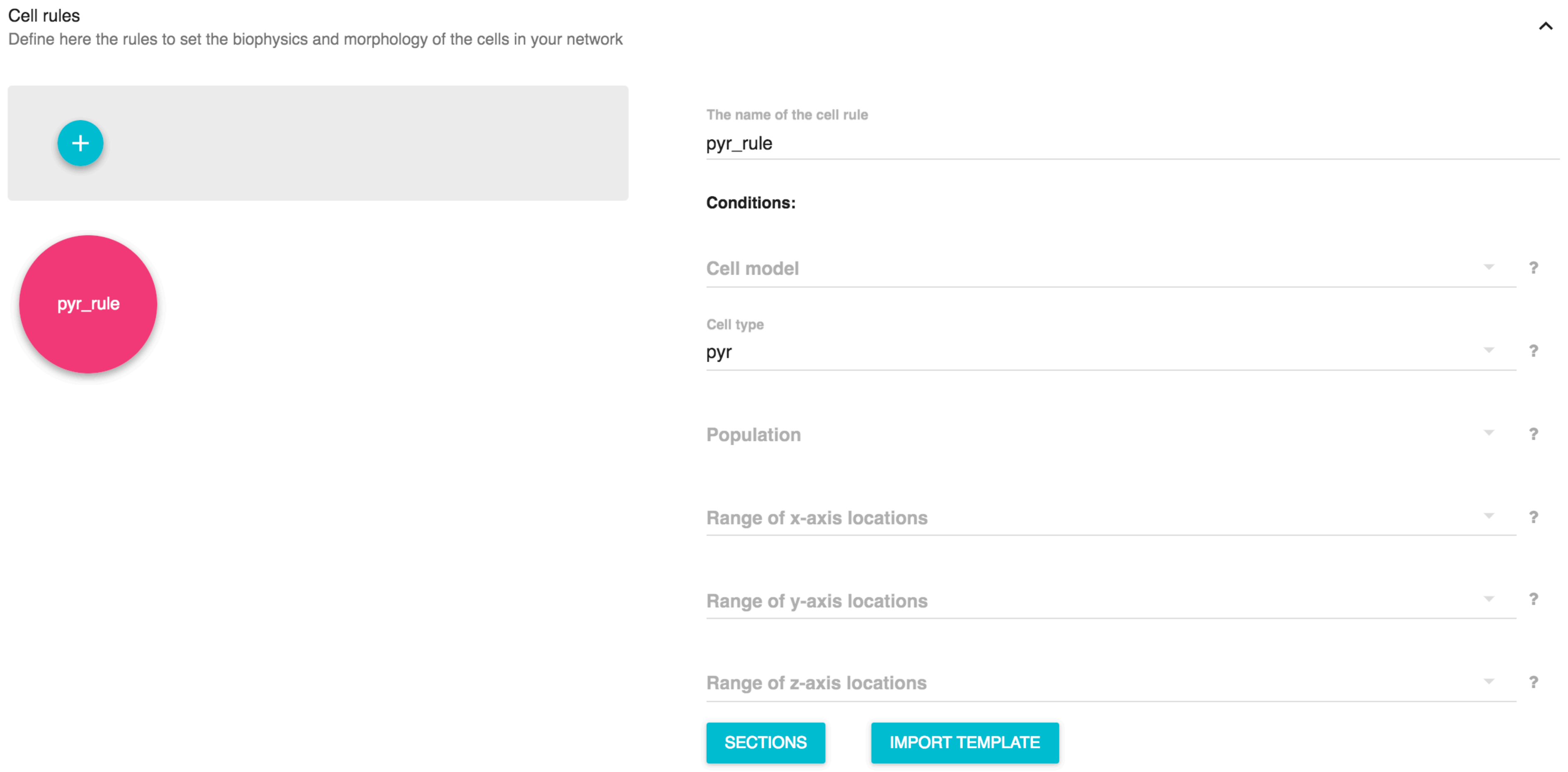
Cell model

Number of cells
Number of cells ?

Number of cells
20 ?

NetPyNE GUI Tutorial: Simple cell net

3) Add a cell rule ‘pyr_rule’ for ‘pyr’ cells



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

The name of the cell rule: pyr_rule

Conditions:

Cell model

Cell type: pyr

Population

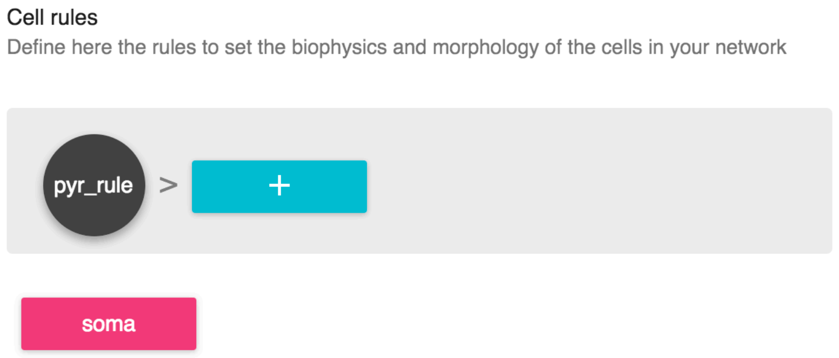
Range of x-axis locations

Range of y-axis locations

Range of z-axis locations

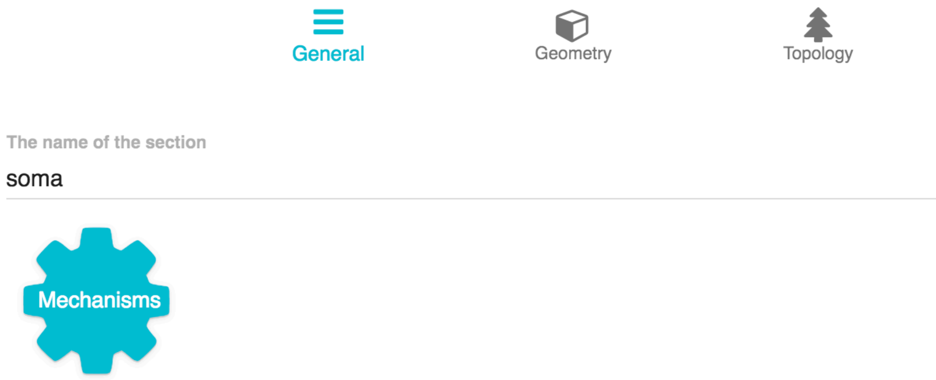
NetPyNE GUI Tutorial: Simple cell net

4) Add a ‘soma’ section in the ‘pyr_rule’



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

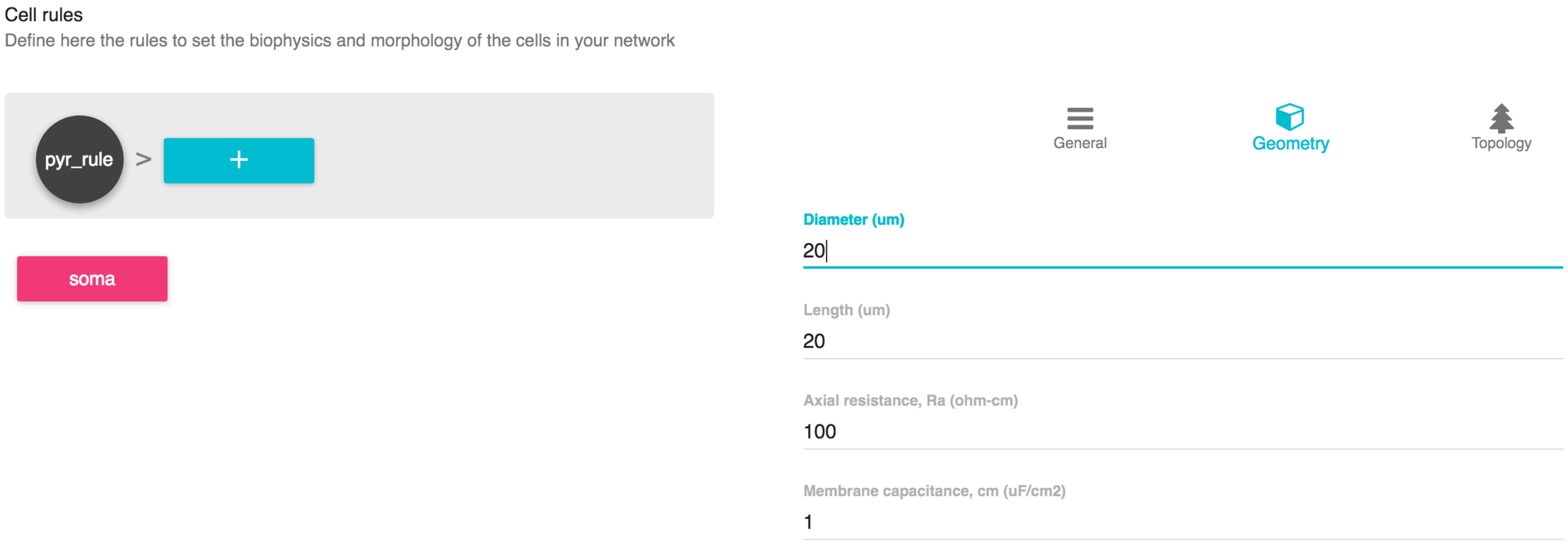


The name of the section

soma

NetPyNE GUI Tutorial: Simple cell net

5) Add the geometry of the ‘soma’ section in the ‘pyr_rule’



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

Diameter (um)
20

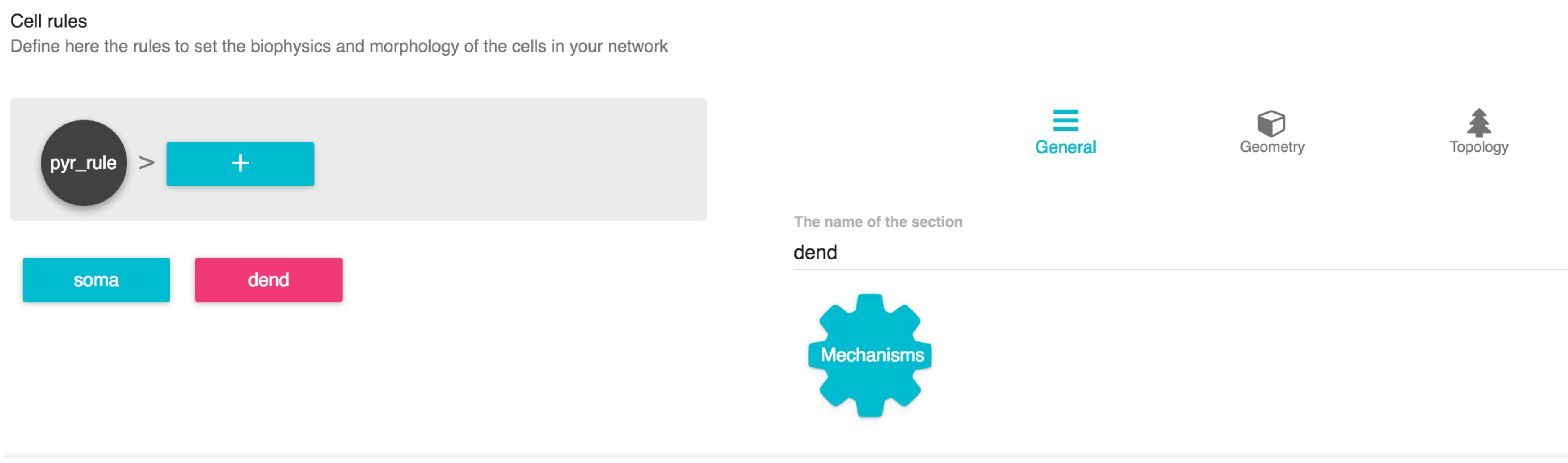
Length (um)
20

Axial resistance, Ra (ohm-cm)
100

Membrane capacitance, cm (uF/cm2)
1

NetPyNE GUI Tutorial: Simple cell net

6) Add a 'dend' section in the 'pyr_rule'



Cell rules

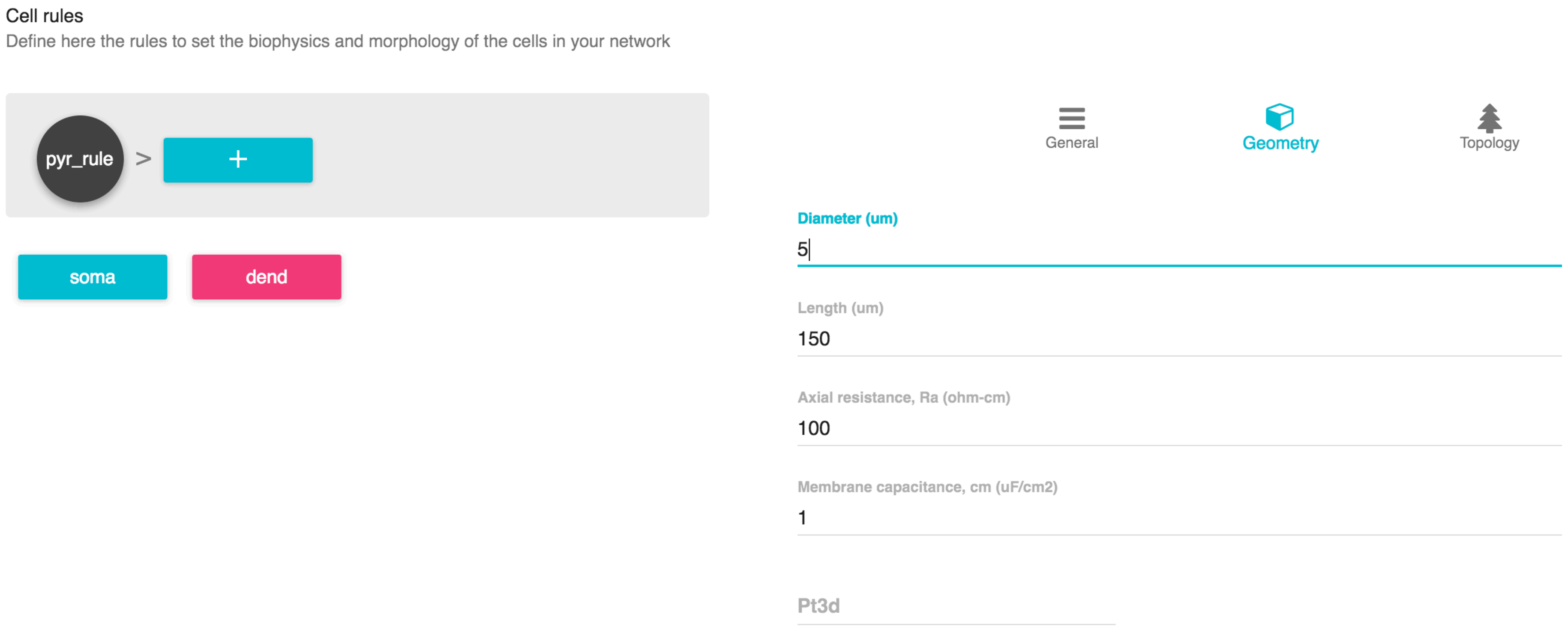
Define here the rules to set the biophysics and morphology of the cells in your network

The name of the section

dend

NetPyNE GUI Tutorial: Simple cell net

7a) Add the geometry of the 'dend' section in the 'pyr_rule'



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

Diameter (um)
5

Length (um)
150

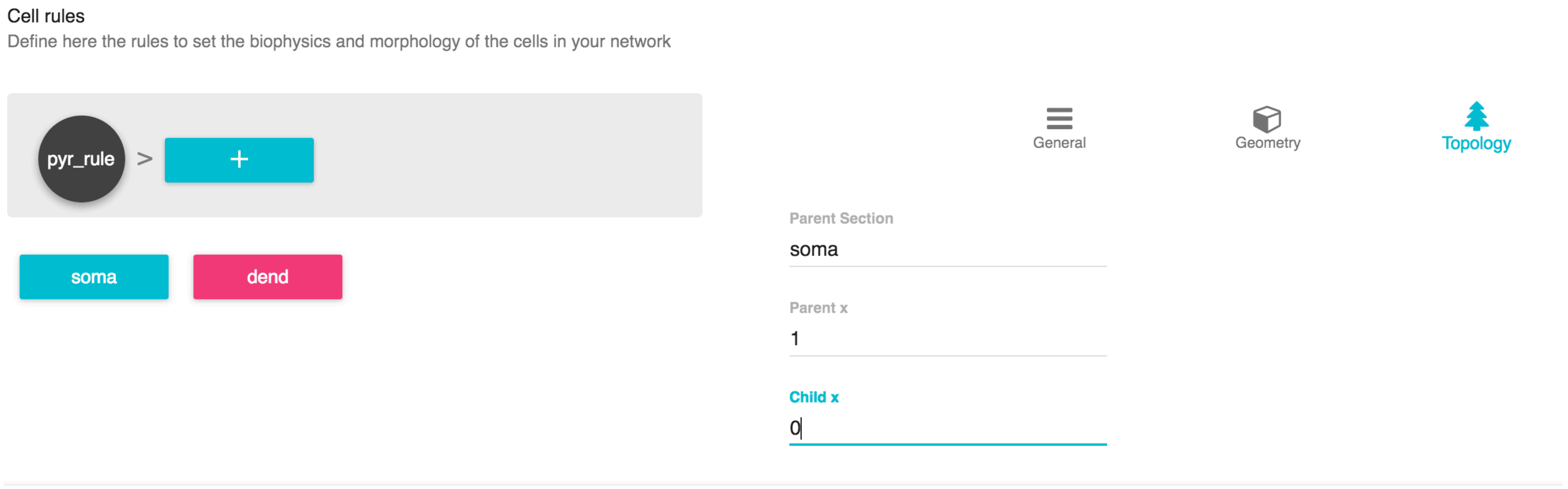
Axial resistance, Ra (ohm-cm)
100

Membrane capacitance, cm (uF/cm2)
1

Pt3d

NetPyNE GUI Tutorial: Simple cell net

7b) Connect the 'dend' section to the 'soma' section in Topology



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

Parent Section

soma

Parent x

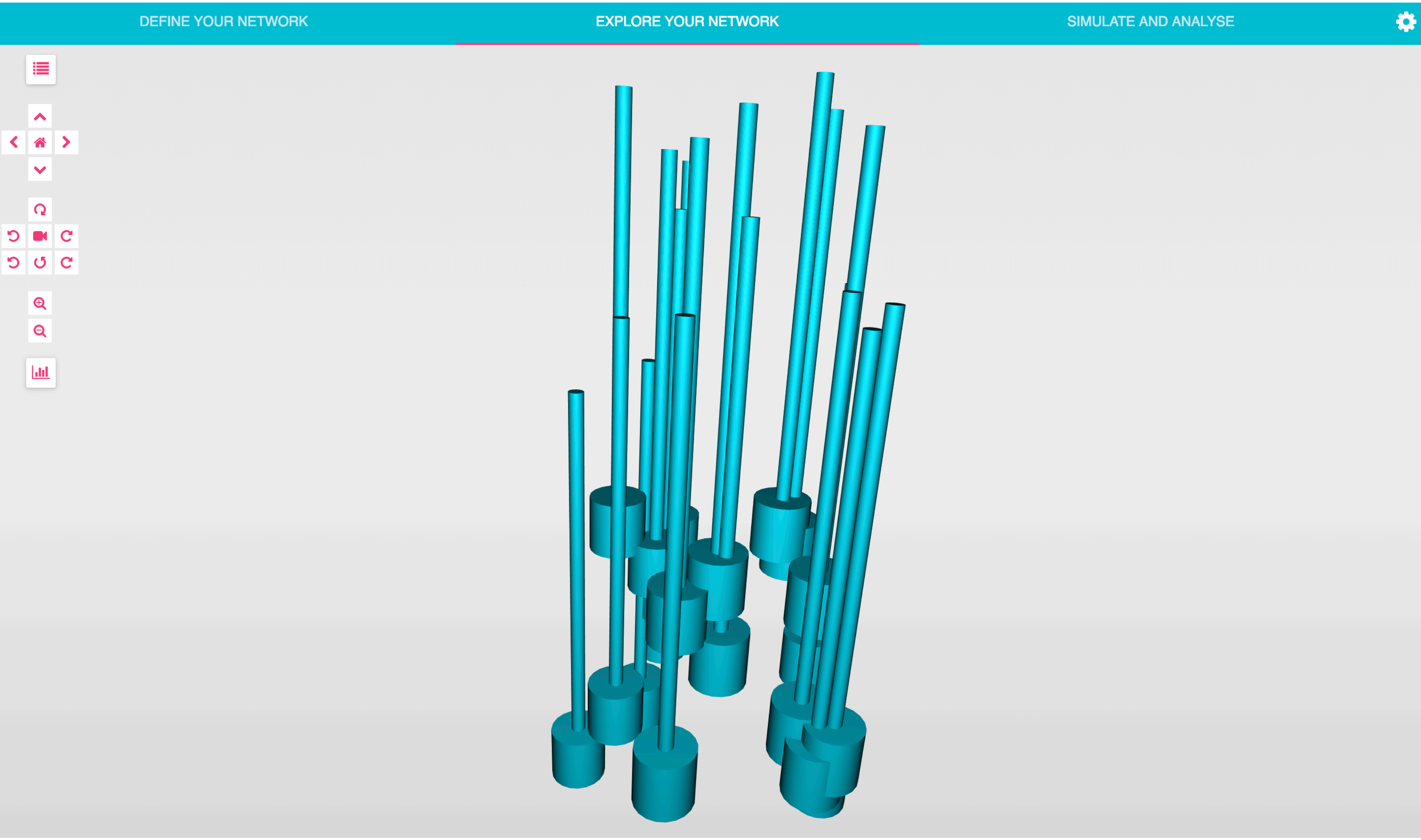
1

Child x

0

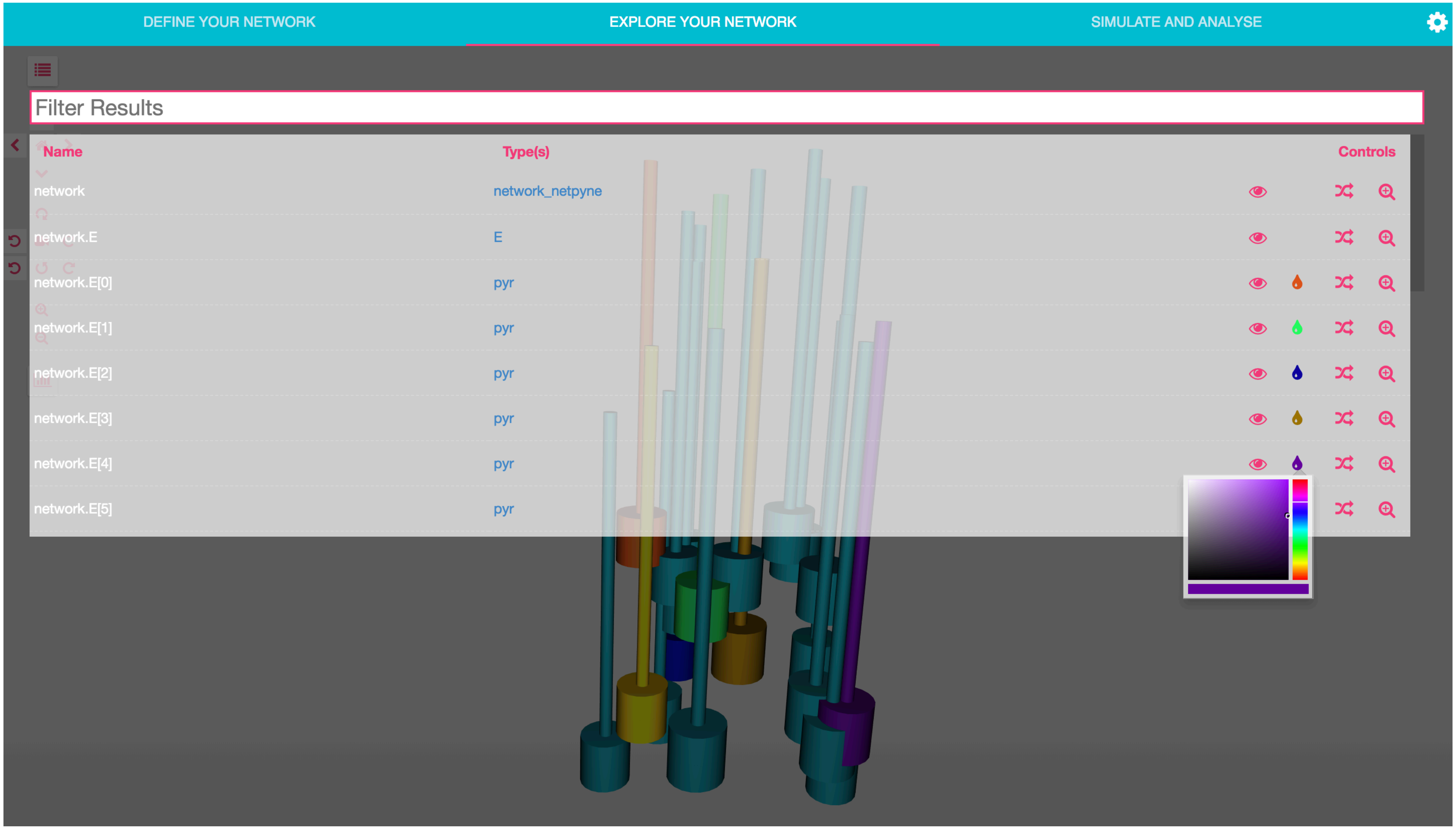
NetPyNE GUI Tutorial: Simple cell net

8) Explore your network … feel free to rotate, zoom and move around!



NetPyNE GUI Tutorial: Simple cell net

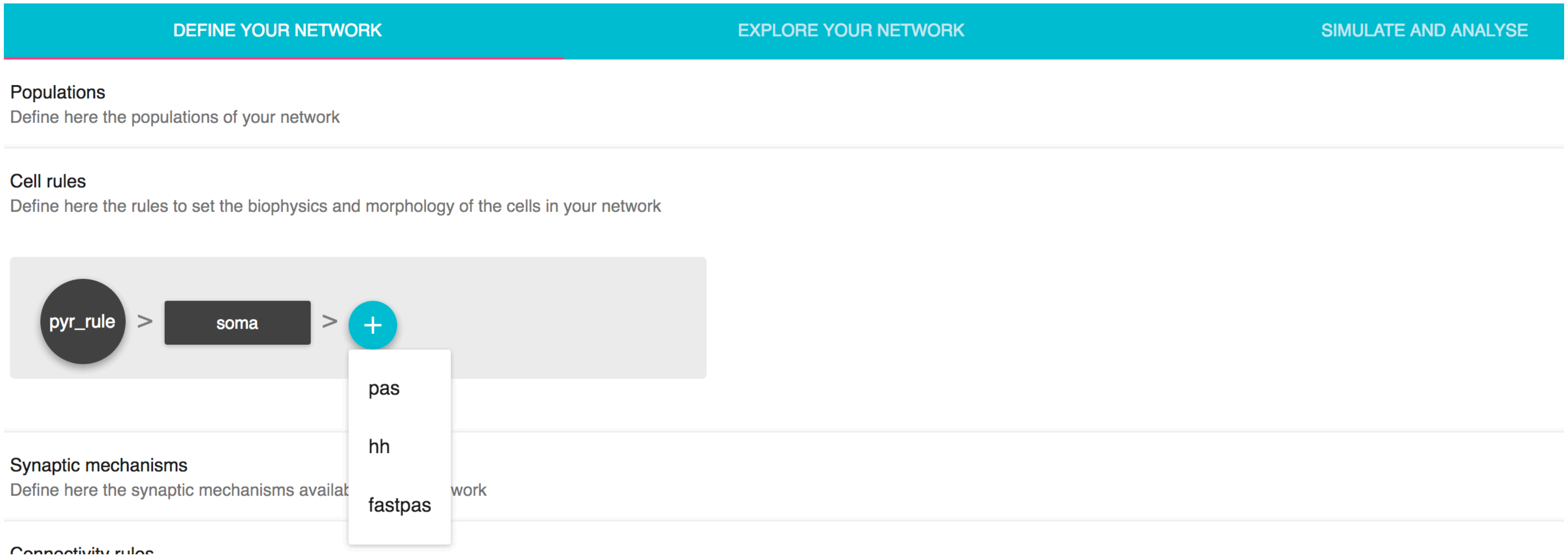
9) Customize the color of your population or cells



Name	Type(s)	Controls
network	network_netpyne	
network.E	E	
network.E[0]	pyr	
network.E[1]	pyr	
network.E[2]	pyr	
network.E[3]	pyr	
network.E[4]	pyr	
network.E[5]	pyr	

NetPyNE GUI Tutorial: Simple cell net

10) To add channels: go to 'Define your network' → 'Cell rules' → 'pyr rule' → 'soma' → 'mechanisms' → (+)



DEFINE YOUR NETWORK   EXPLORE YOUR NETWORK   SIMULATE AND ANALYSE

Populations
Define here the populations of your network

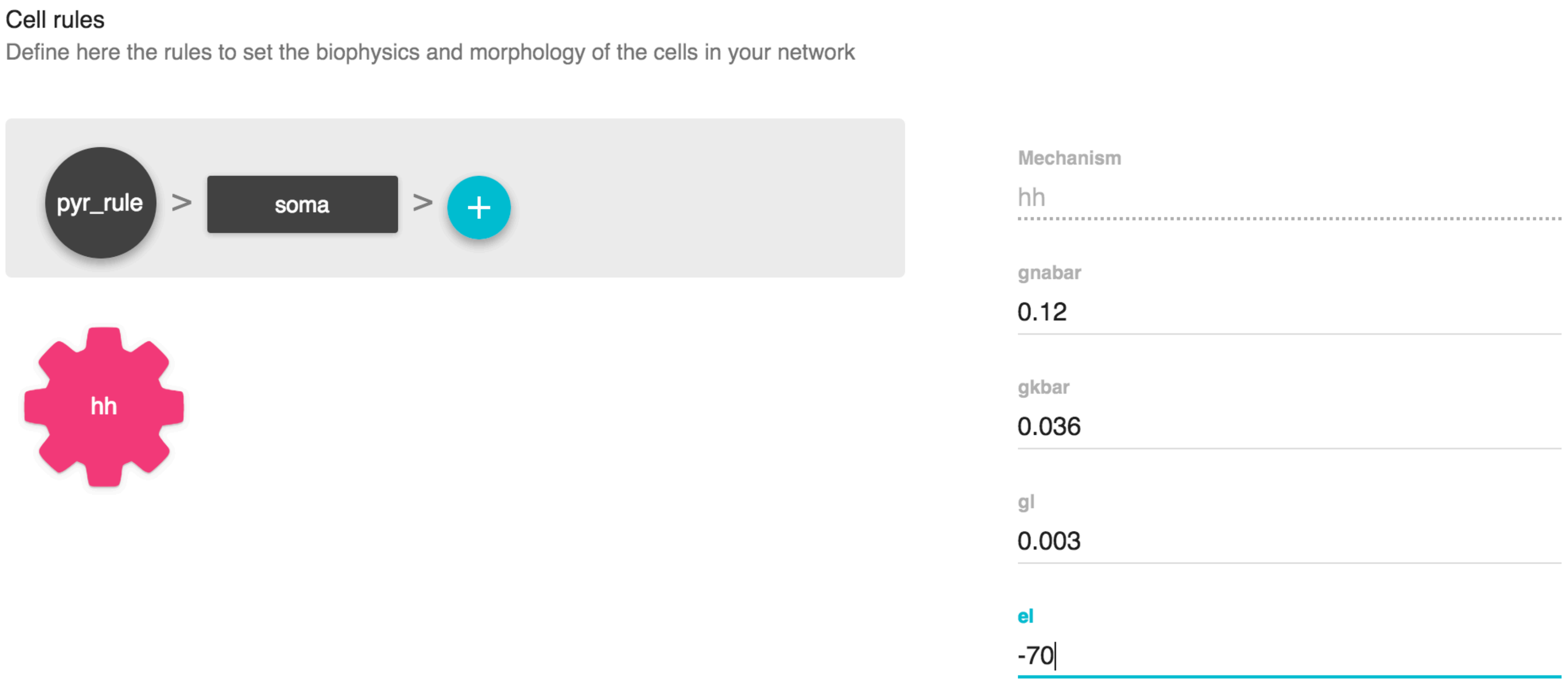
Cell rules
Define here the rules to set the biophysics and morphology of the cells in your network

Synaptic mechanisms
Define here the synaptic mechanisms availab work

Connectivity rules

NetPyNE GUI Tutorial: Simple cell net

11) Add the ‘hh’ (Hodgkin-Huxley) mechanism to the ’soma’ with the following parameters:



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

Mechanism
hh

gnabar
0.12

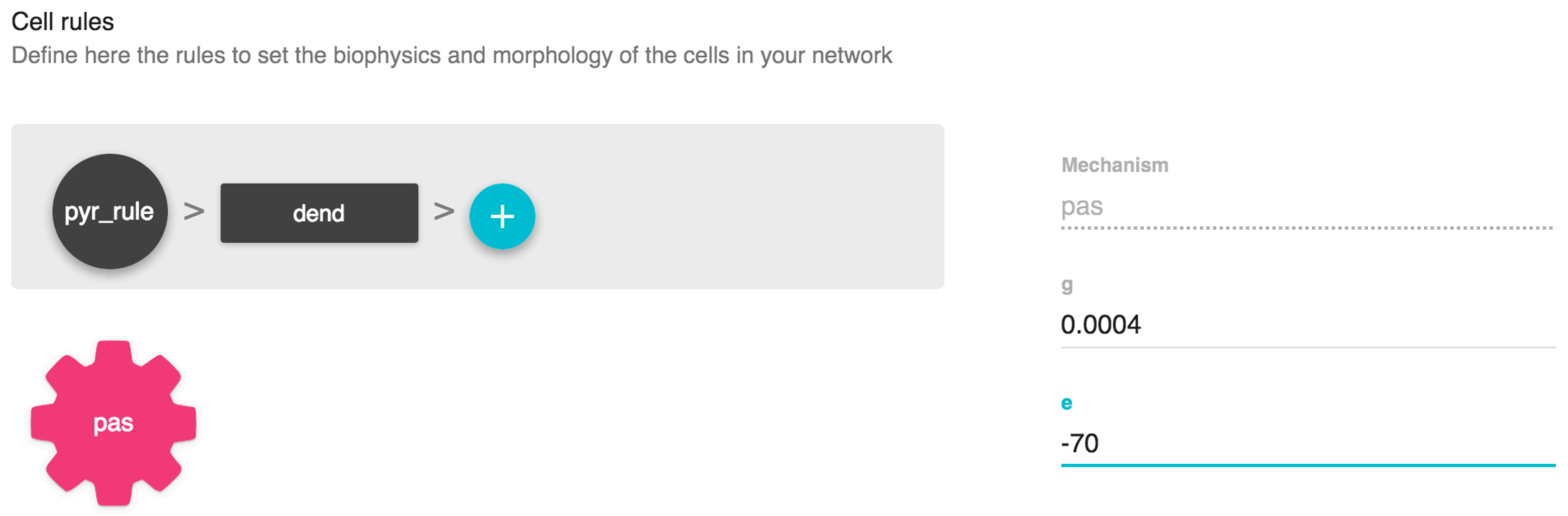
gkbar
0.036

gl
0.003

el
-70

NetPyNE GUI Tutorial: Simple cell net

12) Add the 'pas' (passive) mechanism to the 'dend' with the following parameters:



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

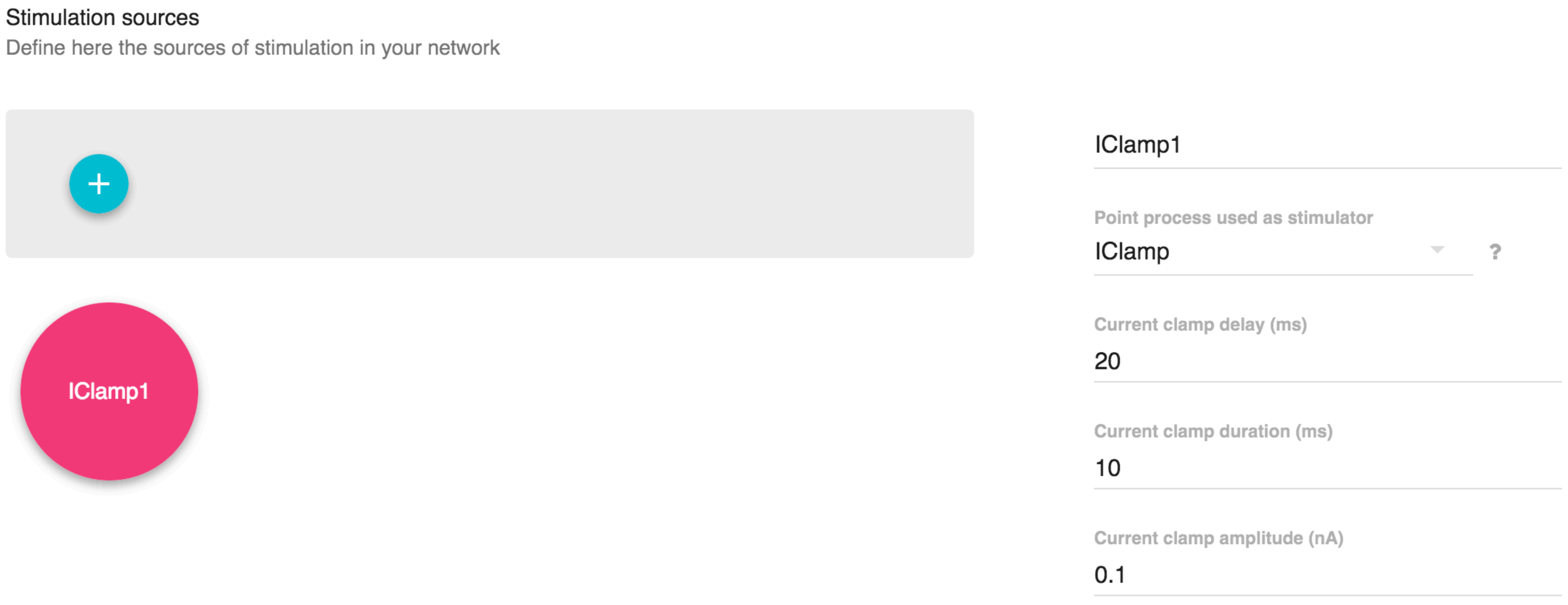
Mechanism: pas

g: 0.0004

e: -70

NetPyNE GUI Tutorial: Simple cell net

13) Add an IClamp (current clamp) source of stimulation



Stimulation sources

Define here the sources of stimulation in your network

IClamp1

Point process used as stimulator: IClamp

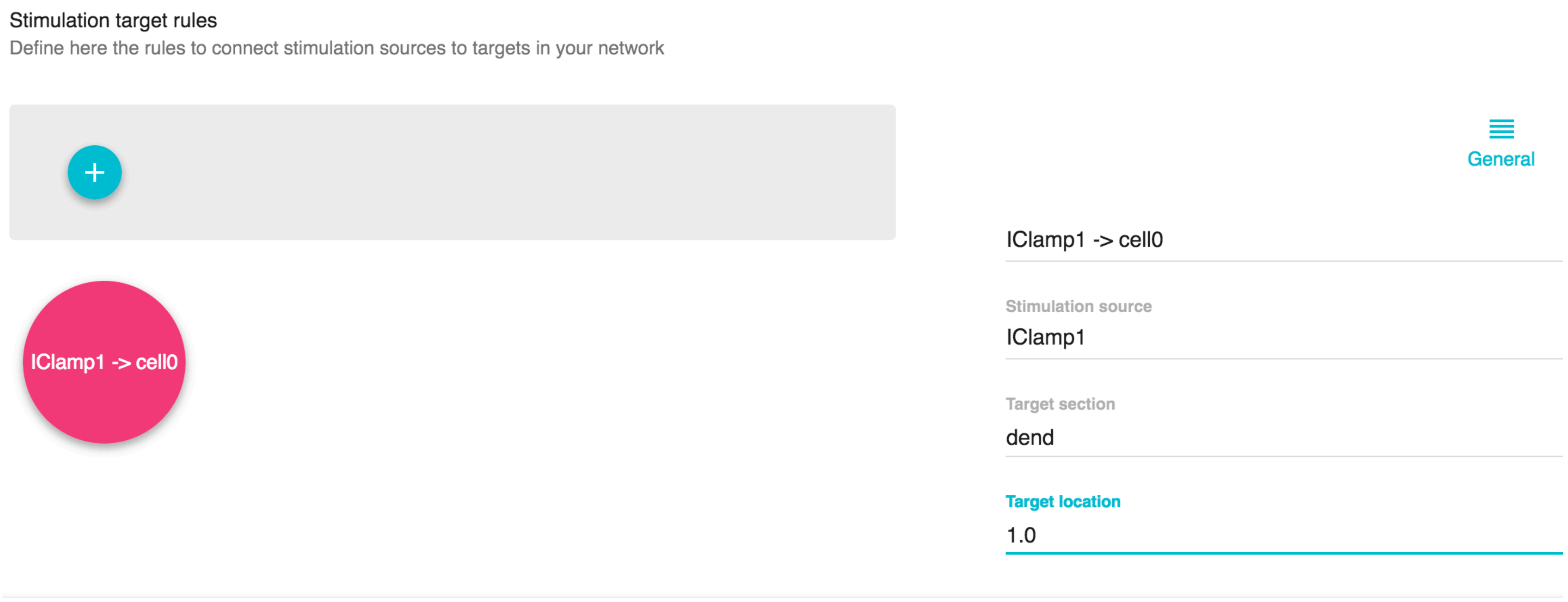
Current clamp delay (ms): 20

Current clamp duration (ms): 10

Current clamp amplitude (nA): 0.1

NetPyNE GUI Tutorial: Simple cell net

14) Create a stimulation target rule to place IClamp1 on the cell dendrite:



Stimulation target rules

Define here the rules to connect stimulation sources to targets in your network

IClamp1 -> cell0

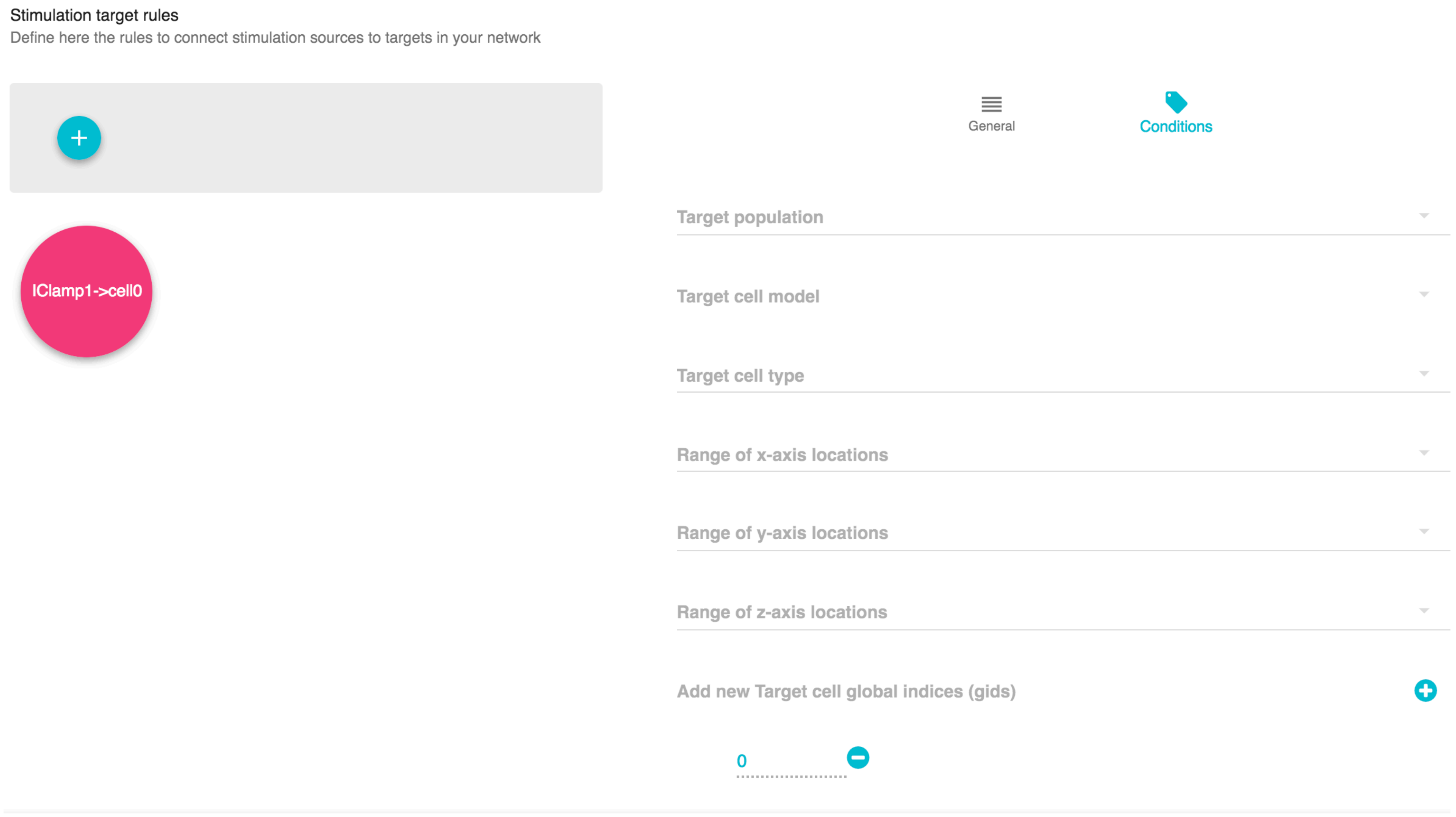
Stimulation source: IClamp1

Target section: dend

Target location: 1.0

NetPyNE GUI Tutorial: Simple cell net

15) Place the IClamp1 just on one of the cells (with global index 0) using the target rule 'conditions':



Stimulation target rules

Define here the rules to connect stimulation sources to targets in your network

Target population

Target cell model

Target cell type

Range of x-axis locations

Range of y-axis locations

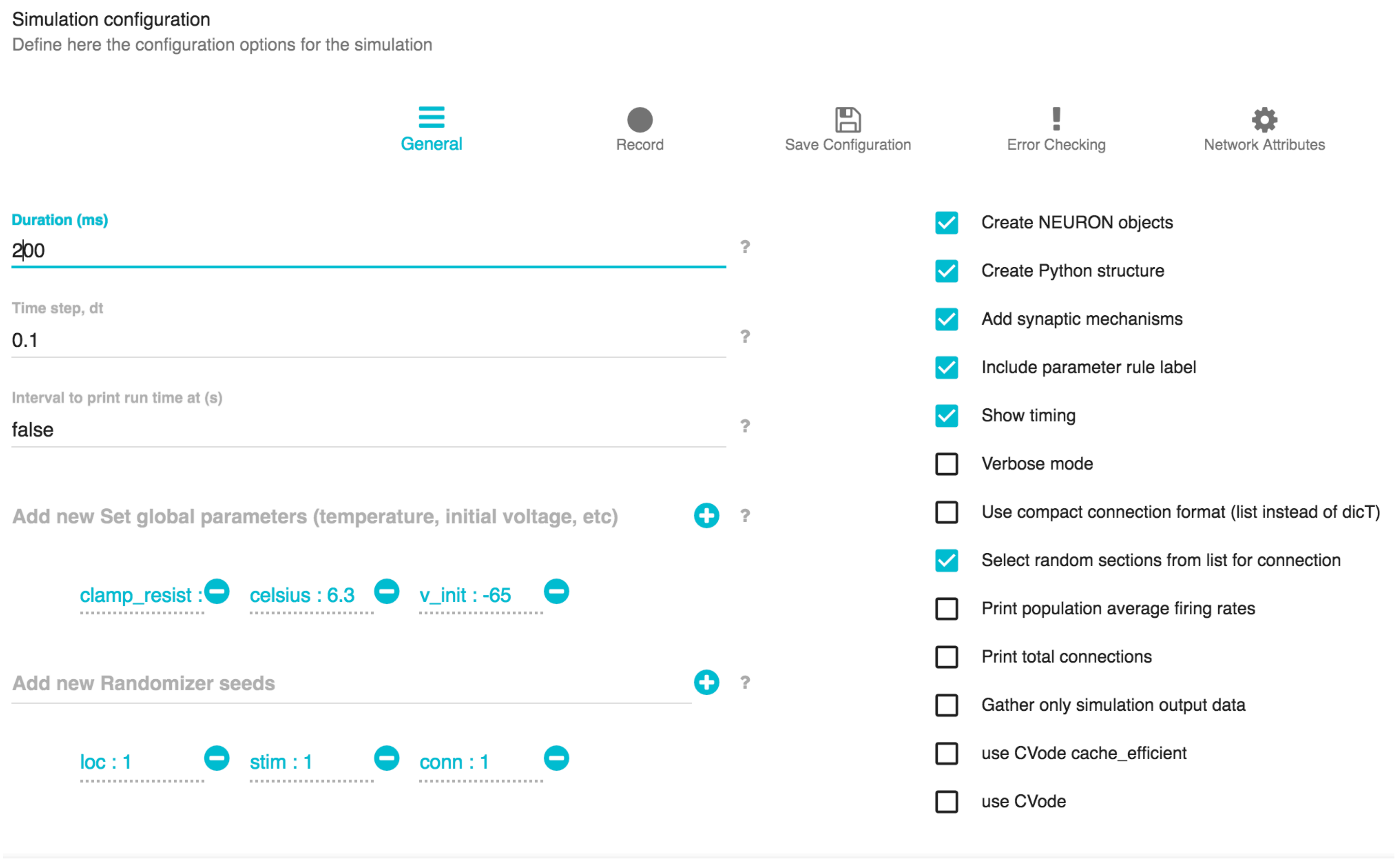
Range of z-axis locations

Add new Target cell global indices (gids)

0

NetPyNE GUI Tutorial: Simple cell net

16) Set the simulation duration to 200ms in 'Simulation configuration'



Simulation configuration
Define here the configuration options for the simulation

Duration (ms)
200

Time step, dt
0.1

Interval to print run time at (s)
false

Add new Set global parameters (temperature, initial voltage, etc)

clamp_resist : celsius : 6.3 v_init : -65

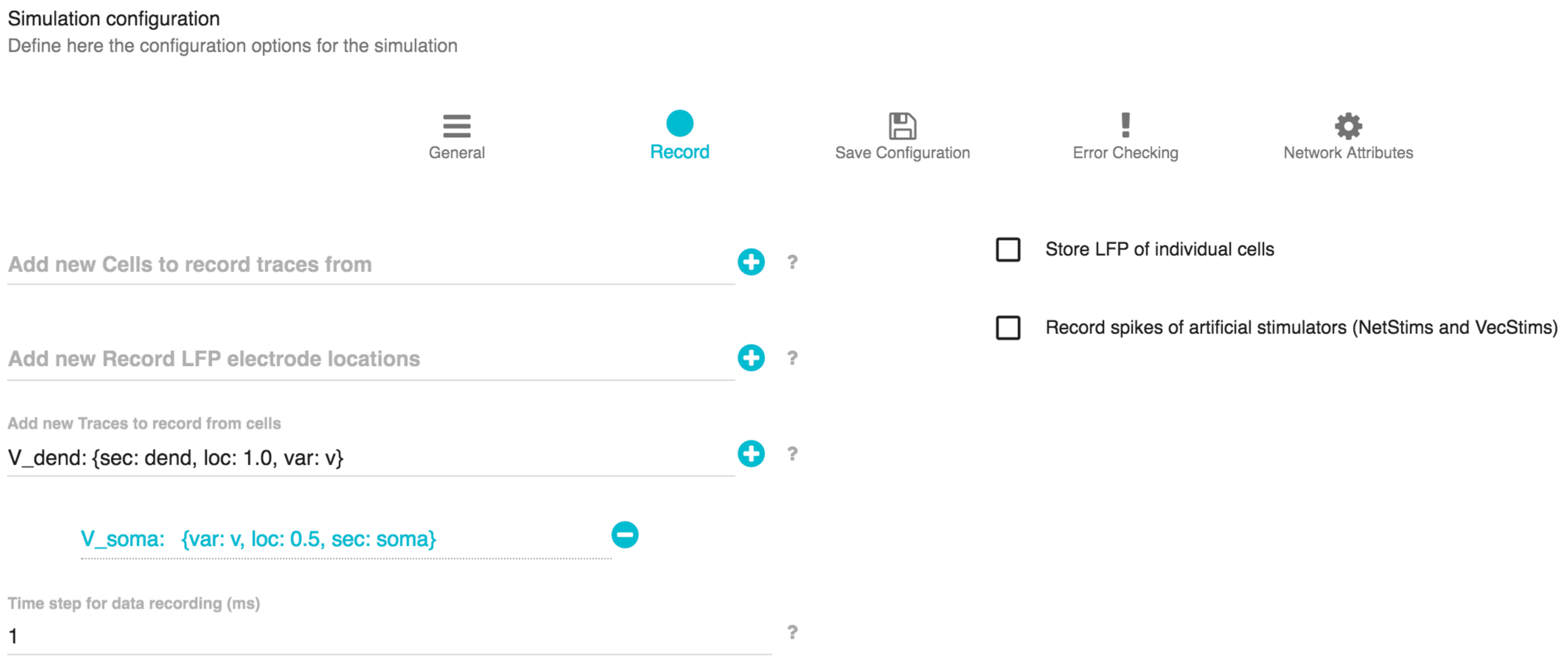
Add new Randomizer seeds

loc : 1 stim : 1 conn : 1

- [✓] Create NEURON objects
- [✓] Create Python structure
- [✓] Add synaptic mechanisms
- [✓] Include parameter rule label
- [✓] Show timing
- [ ] Verbose mode
- [ ] Use compact connection format (list instead of dicT)
- [✓] Select random sections from list for connection
- [ ] Print population average firing rates
- [ ] Print total connections
- [ ] Gather only simulation output data
- [ ] use CVode cache_efficient
- [ ] use CVode

NetPyNE GUI Tutorial: Simple cell net

17) Record soma and dendrite voltage traces from from cell with id 0:


Simulation configuration

Define here the configuration options for the simulation

Add new Cells to record traces from ?

Add new Record LFP electrode locations ?

Add new Traces to record from cells

V_dend: {sec: dend, loc: 1.0, var: v} ?

V_soma: {var: v, loc: 0.5, sec: soma}

Time step for data recording (ms)

1 ?

☐ Store LFP of individual cells

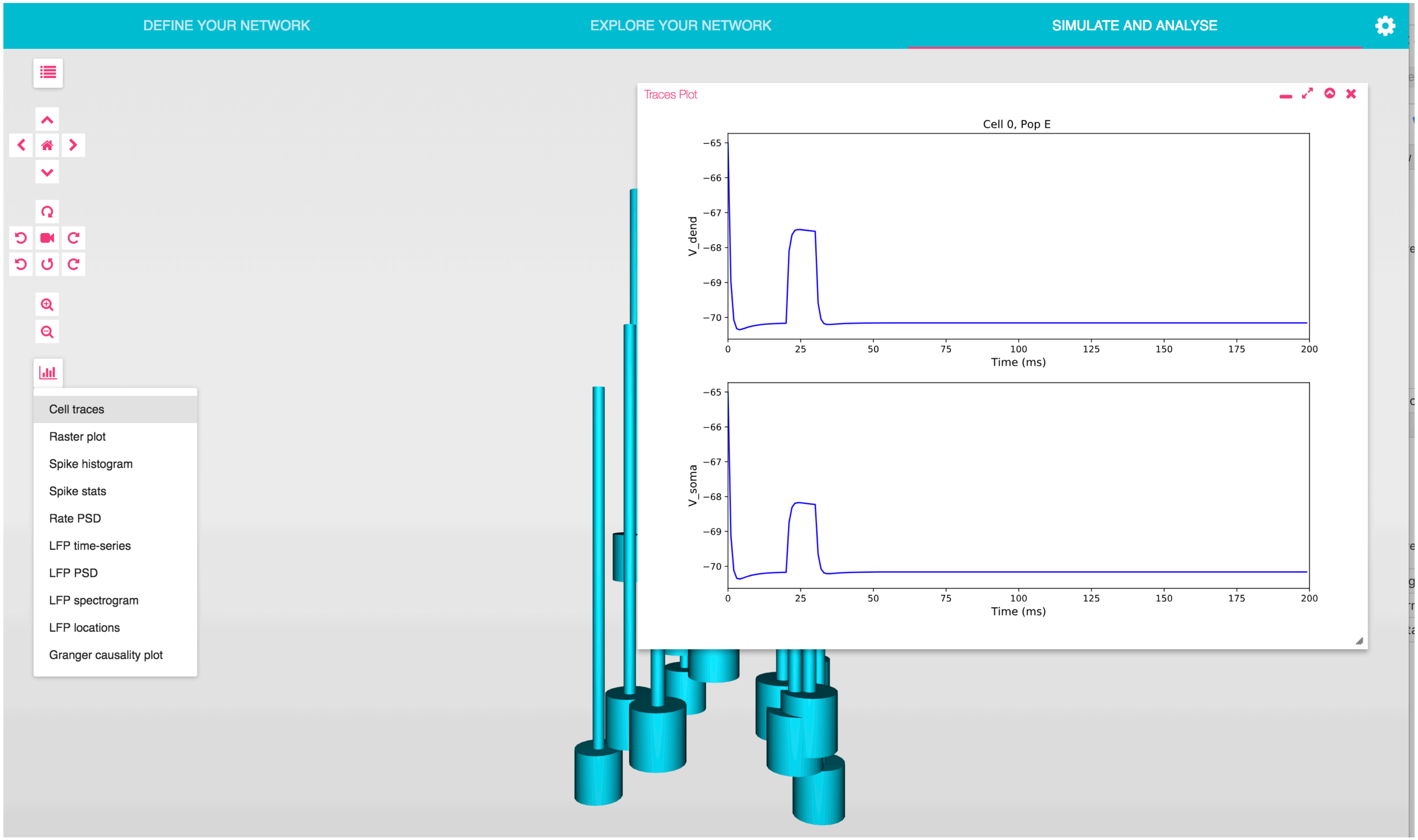
☐ Record spikes of artificial stimulators (NetStims and VecStims)

Specify traces to record using Python dictionary format (no quotes required):

```
V_soma: {var: v, sec: soma, loc: 0.5}
V_dend: {var: v, sec: dend, loc: 1.0}
```

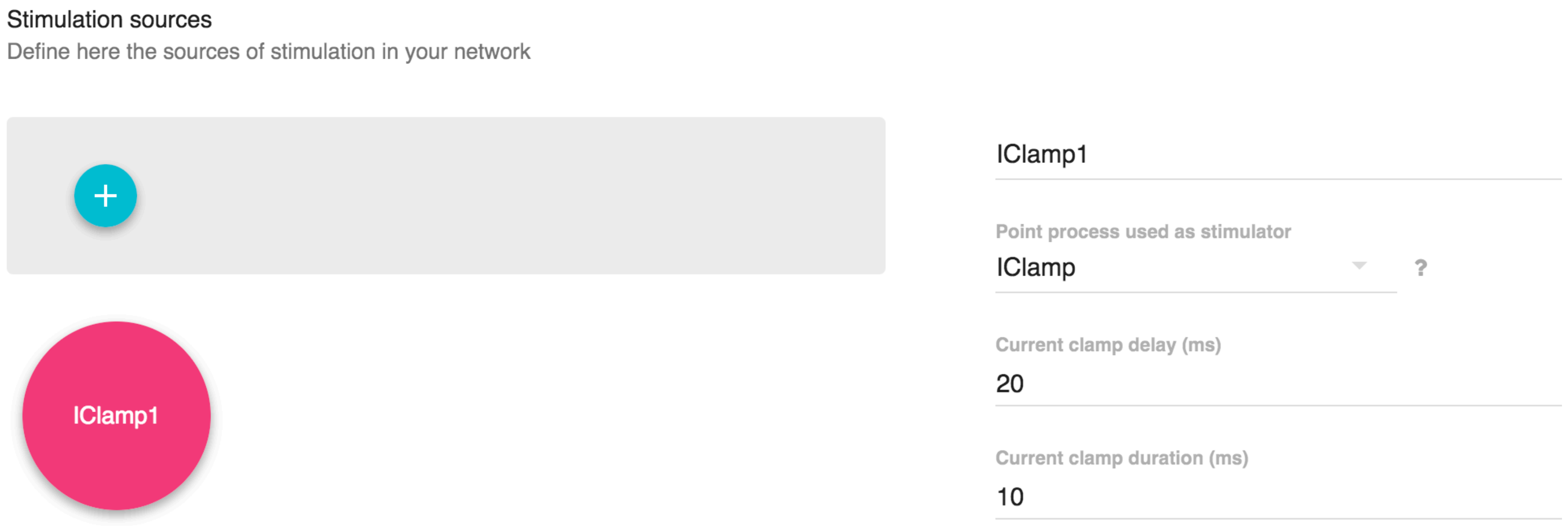
NetPyNE GUI Tutorial: Simple cell net

18) ‘Simulate and Analyze’ the network and plot ‘Cell traces’



NetPyNE GUI Tutorial: Simple cell net

19) Increase 'IClamp1' amplitude so generate a spike (set to 0.6 nA)


Stimulation sources

Define here the sources of stimulation in your network

IClamp1

Point process used as stimulator: IClamp

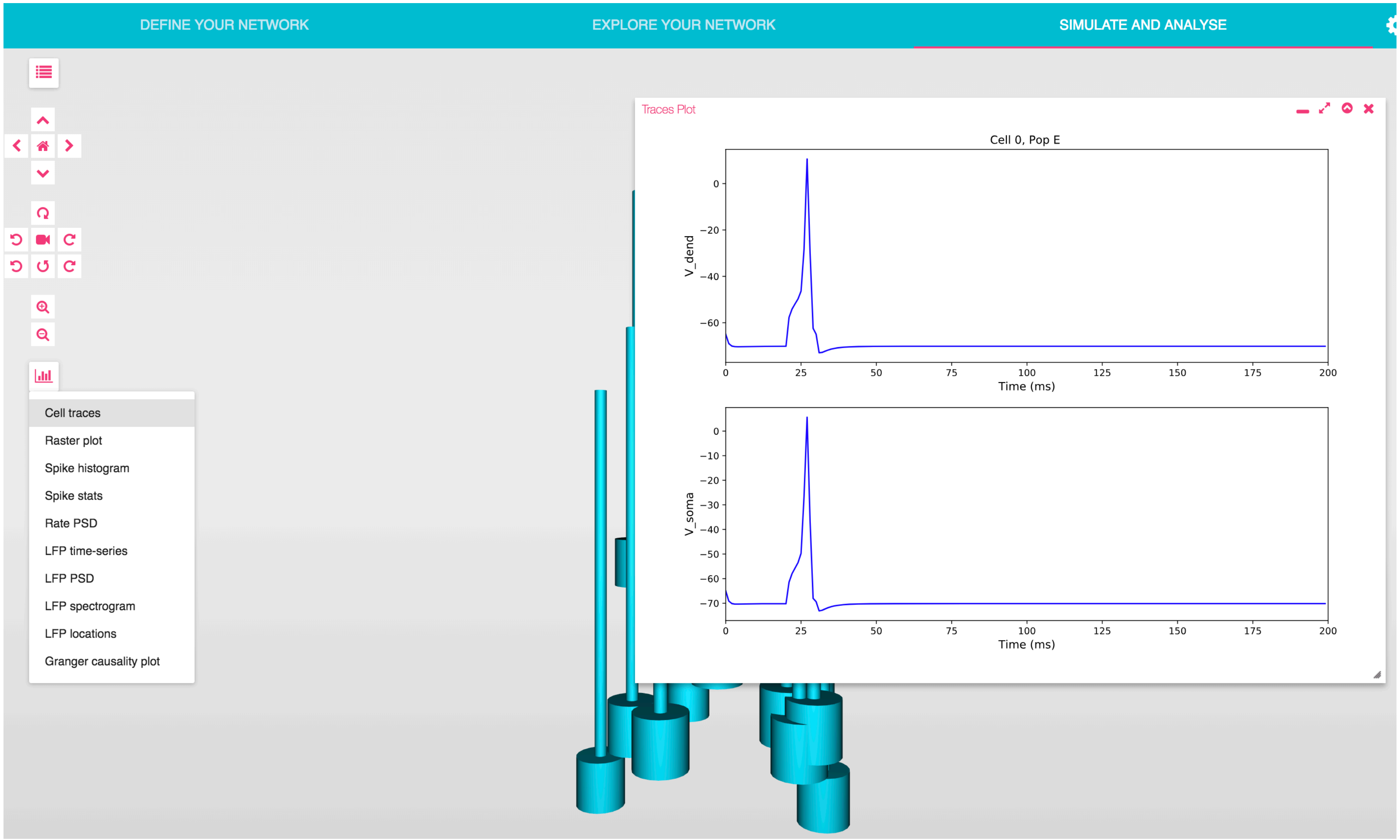
Current clamp delay (ms): 20

Current clamp duration (ms): 10

Current clamp amplitude (nA): 0.6

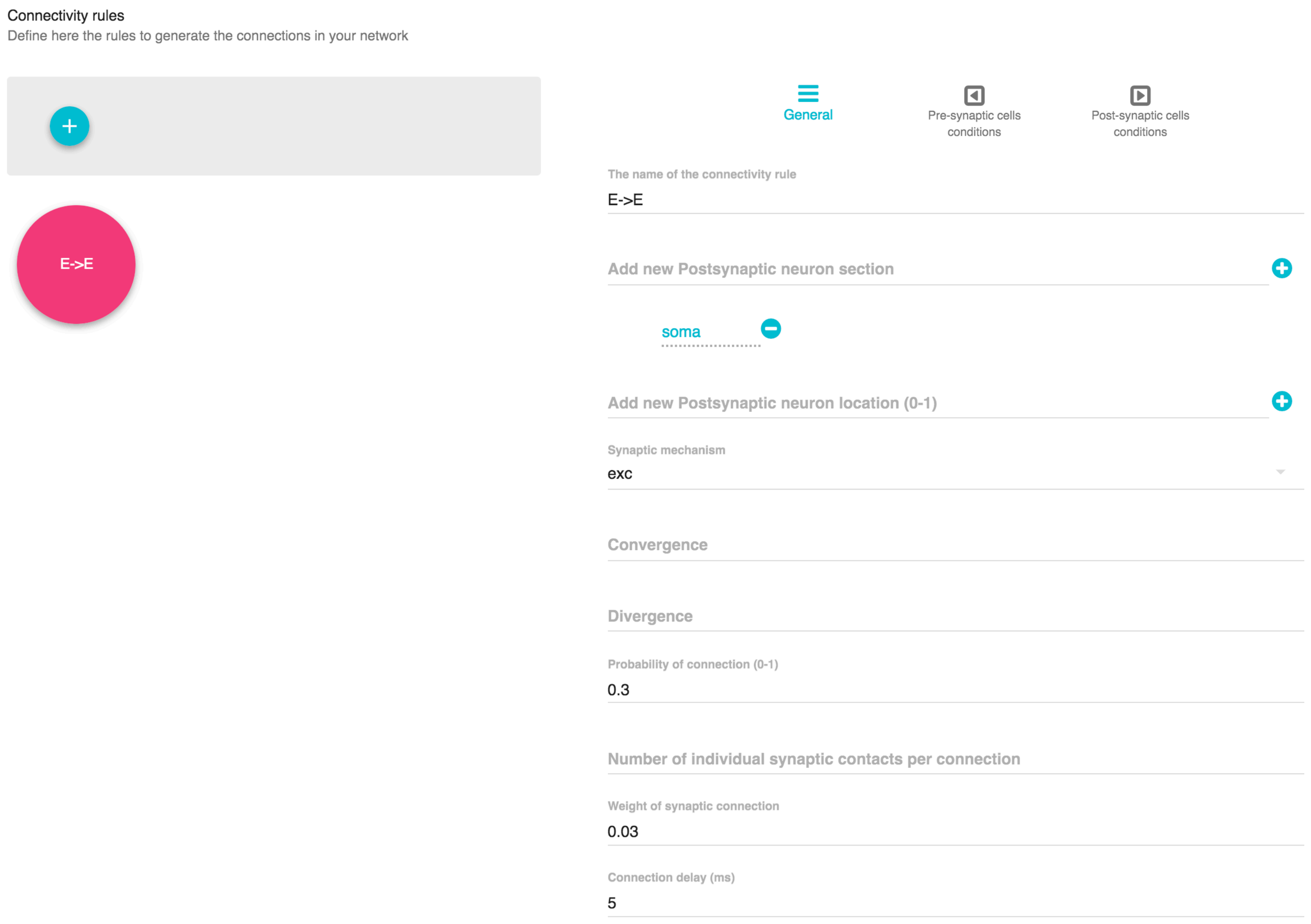
NetPyNE GUI Tutorial: Simple cell net

21) Simulate and plot traces (dendrite current clamp, soma spike and back-propagation to dendrite)



NetPyNE GUI Tutorial: Simple cell net

22a) Create recurrent connections (E->E) rule; syn=exc, probablity=0.3, weight=0.03, delay=5


Connectivity rules
Define here the rules to generate the connections in your network
+
E->E
General
Pre-synaptic cells conditions
Post-synaptic cells conditions

The name of the connectivity rule
E->E

Add new Postsynaptic neuron section

soma

Add new Postsynaptic neuron location (0-1)

Synaptic mechanism
exc

Convergence

Divergence

Probability of connection (0-1)
0.3

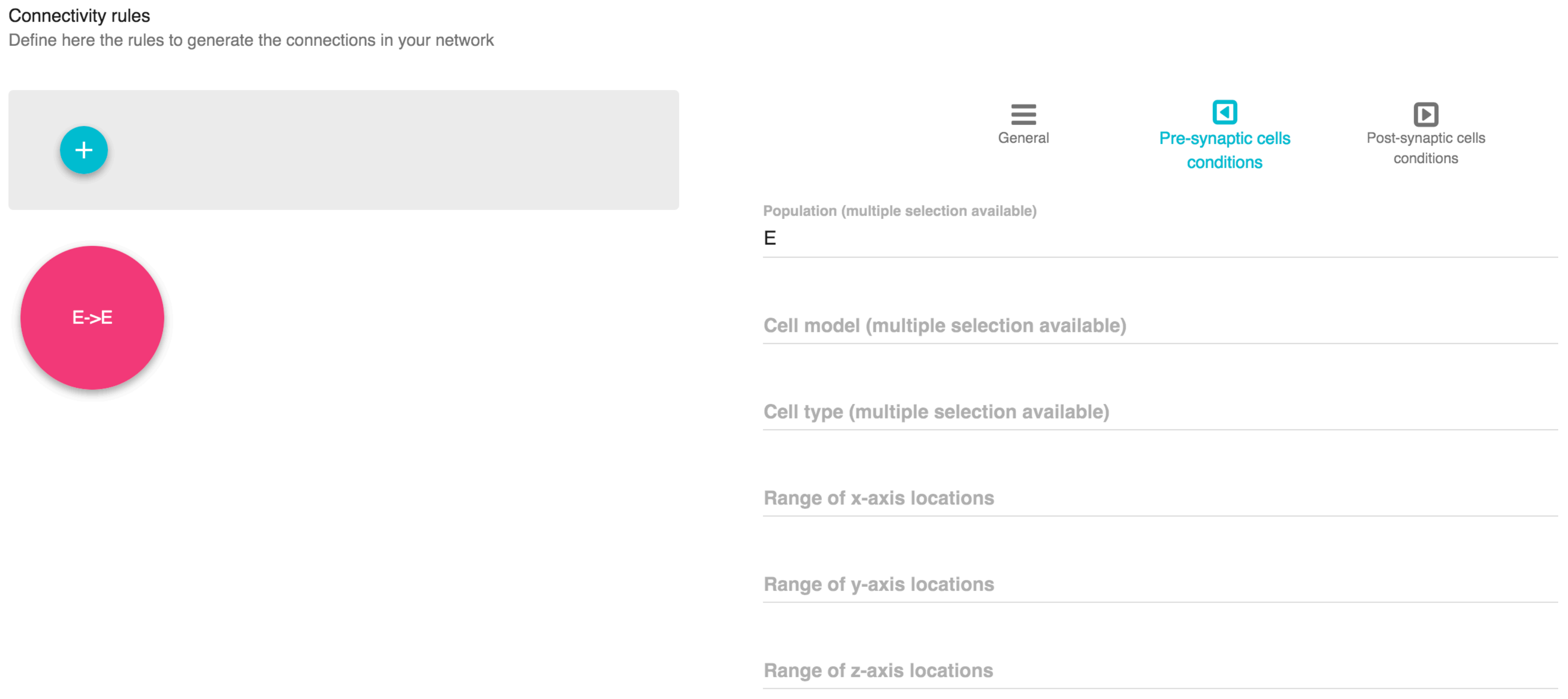
Number of individual synaptic contacts per connection

Weight of synaptic connection
0.03

Connection delay (ms)
5

NetPyNE GUI Tutorial: Simple cell net

22b) Make presynaptic cells condition be 'E' population



Connectivity rules

Define here the rules to generate the connections in your network

Population (multiple selection available)

E

Cell model (multiple selection available)

Cell type (multiple selection available)

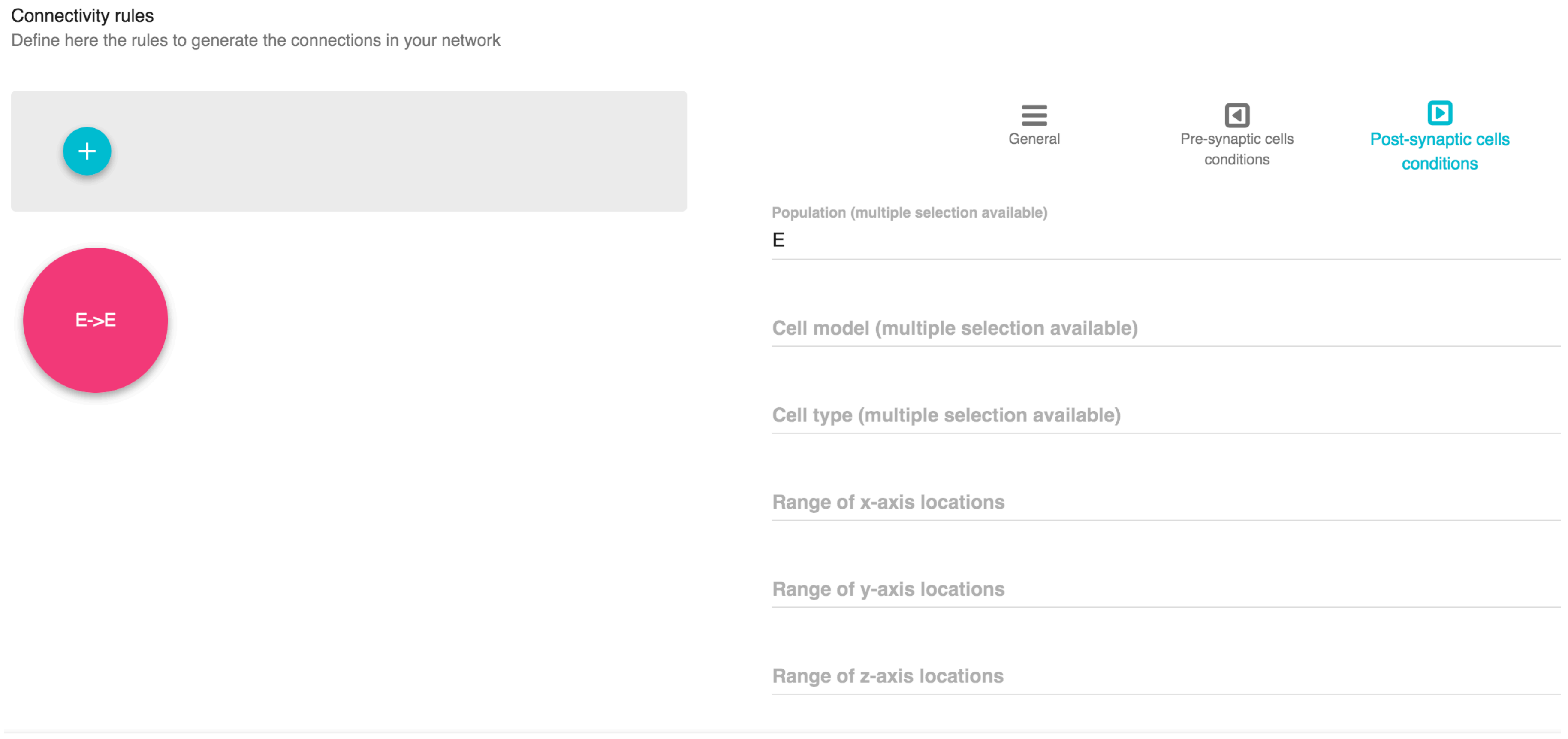
Range of x-axis locations

Range of y-axis locations

Range of z-axis locations

NetPyNE GUI Tutorial: Simple cell net

22c) Make postsynaptic cells condition be ‘E’ population


Connectivity rules

Define here the rules to generate the connections in your network

Population (multiple selection available)
E

Cell model (multiple selection available)

Cell type (multiple selection available)

Range of x-axis locations

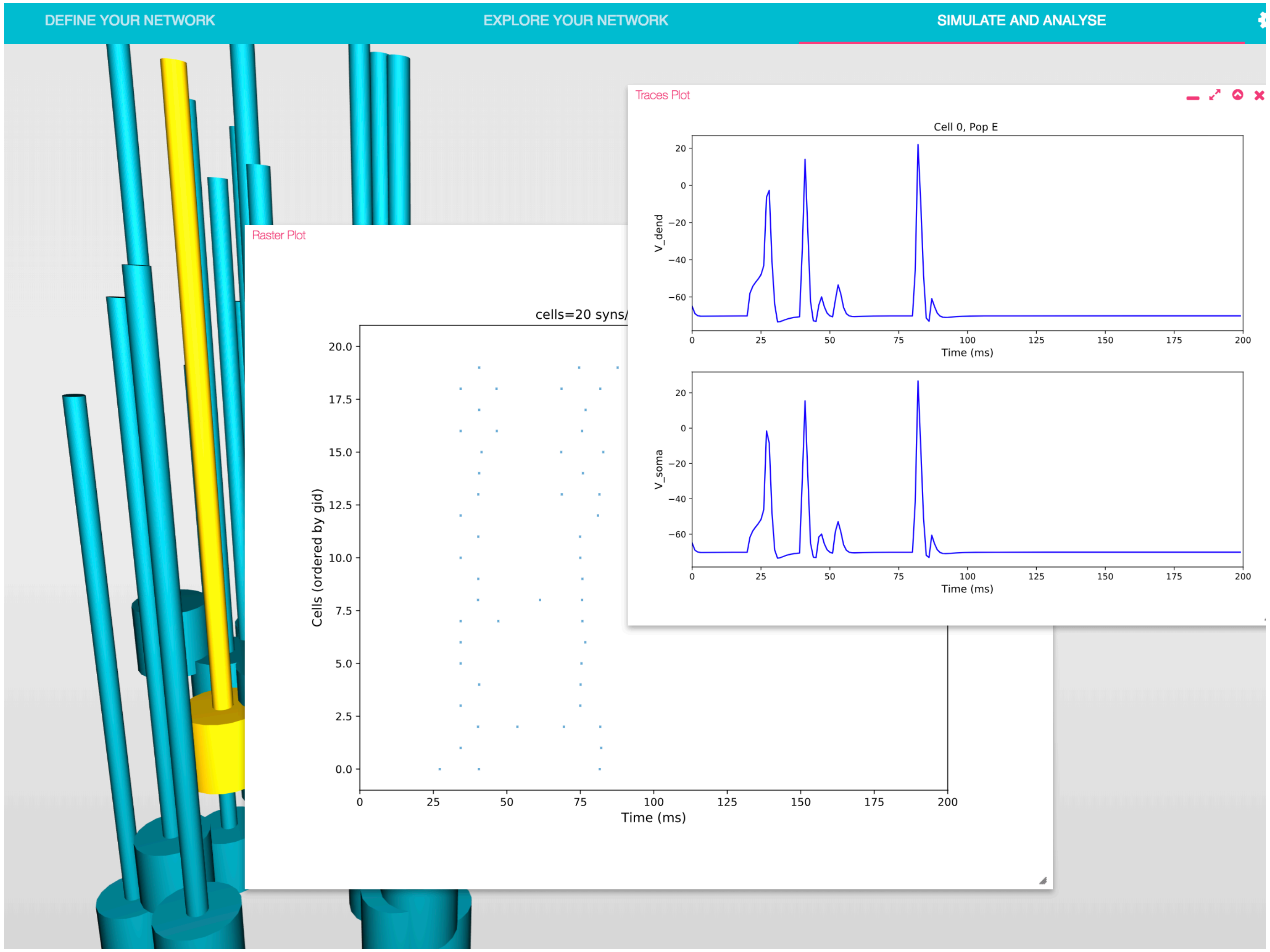
Range of y-axis locations

Range of z-axis locations

NetPyNE GUI Tutorial: Simple cell net

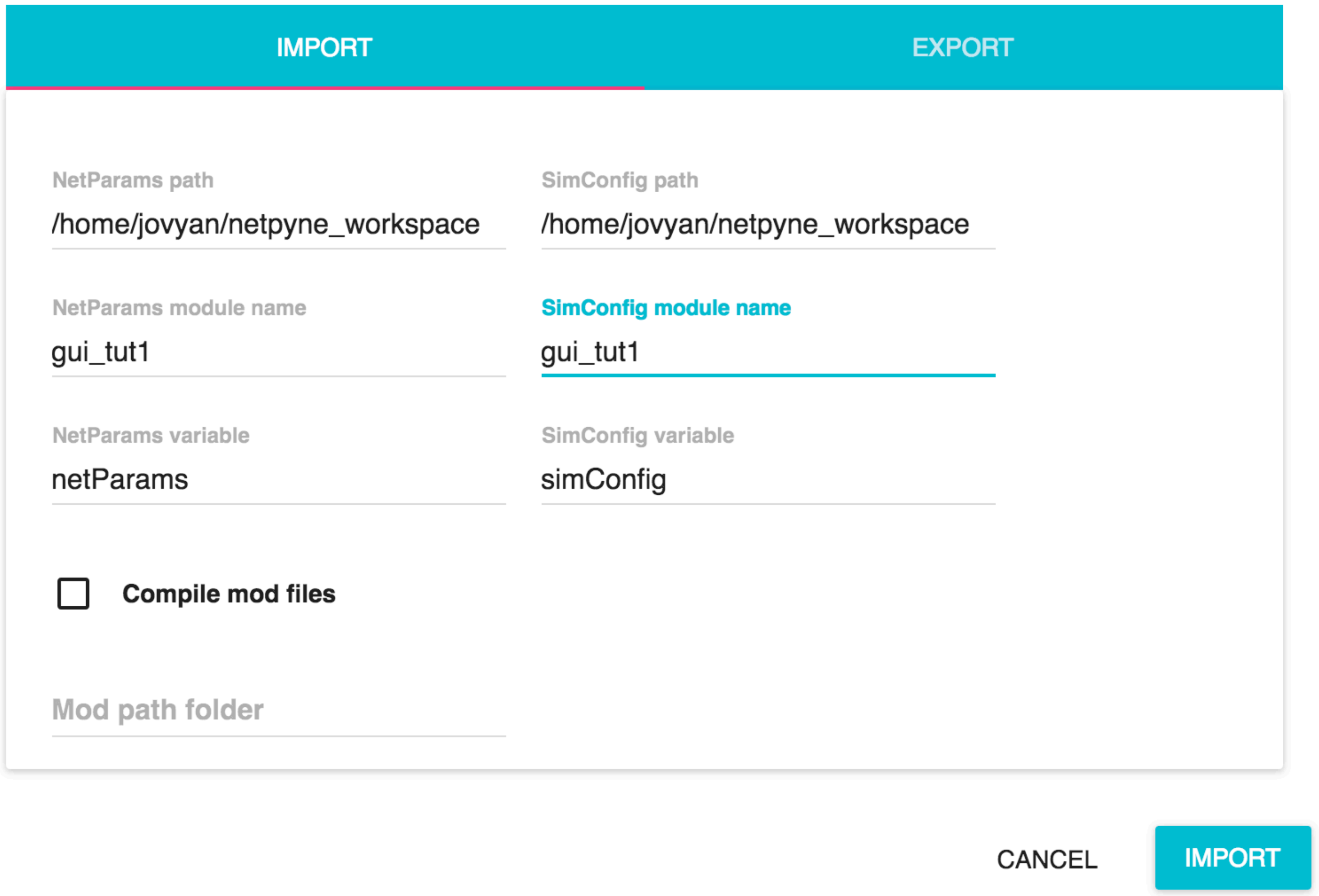
23) Simulate and plot traces and raster plot

Cell 0 spikes due to IClamp -> triggers spikes in other cells due to conn -> cell 0 spikes again



NetPyNE GUI Tutorial: Simple cell net

Note: If you have any errors with step-by-step, try loading the **“simple cell net”** tutorial directly from file



NetParams path: /home/jovyan/netpyne_workspace

SimConfig path: /home/jovyan/netpyne_workspace

NetParams module name: gui_tut1

SimConfig module name: gui_tut1

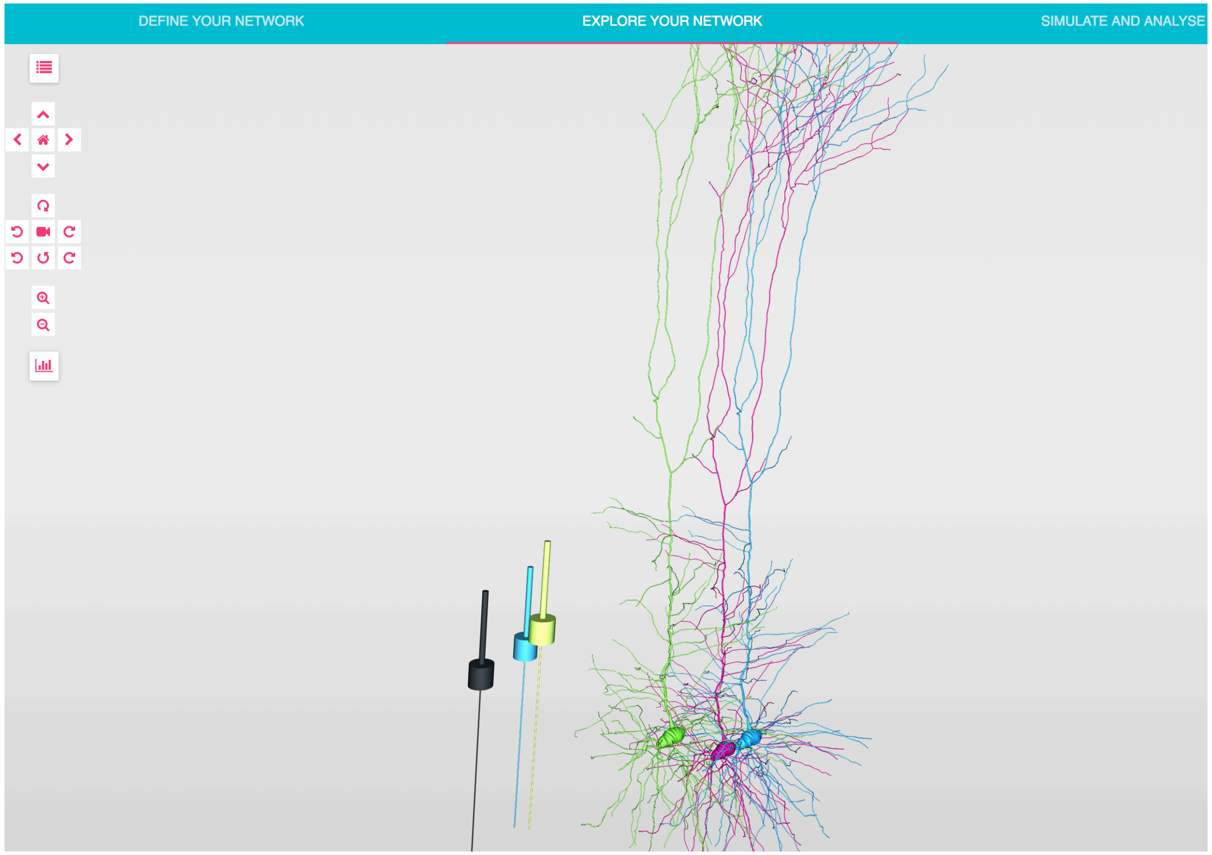
NetParams variable: netParams

SimConfig variable: simConfig

☐ Compile mod files

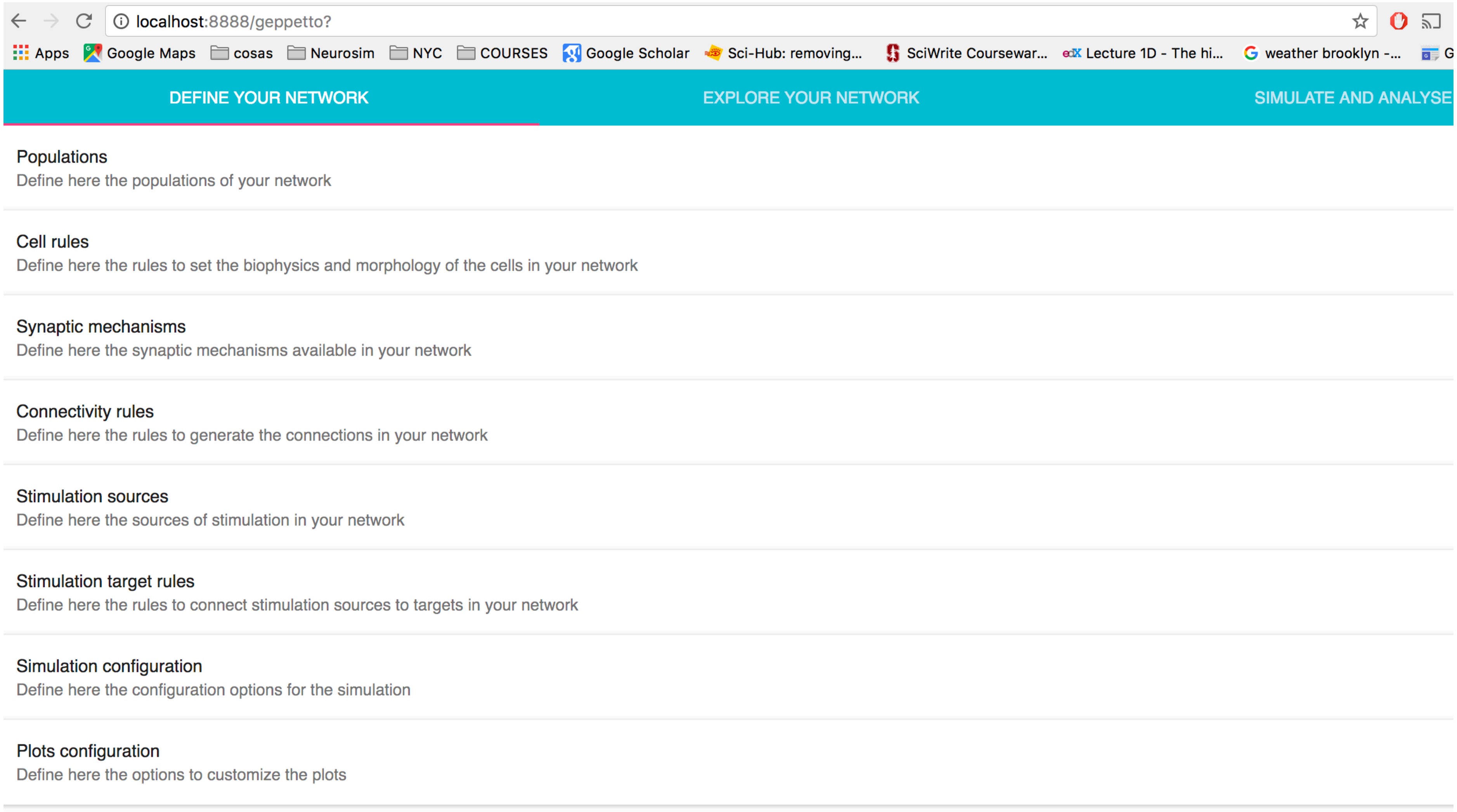
Mod path folder

NetPyNE GUI Tutorial: Complex cell net



NetPyNE GUI Tutorial: Complex cell net

1) Reload webpage to start from scratch



Populations
Define here the populations of your network

Cell rules
Define here the rules to set the biophysics and morphology of the cells in your network

Synaptic mechanisms
Define here the synaptic mechanisms available in your network

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

Stimulation sources
Define here the sources of stimulation in your network

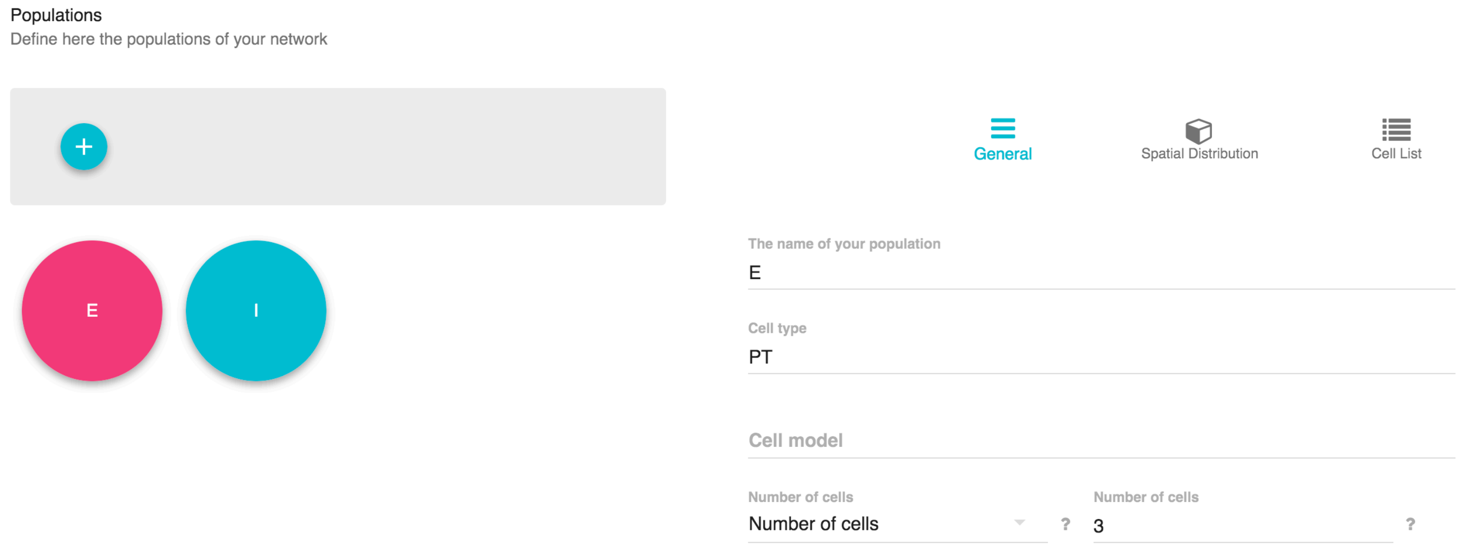
Stimulation target rules
Define here the rules to connect stimulation sources to targets in your network

Simulation configuration
Define here the configuration options for the simulation

Plots configuration
Define here the options to customize the plots

NetPyNE GUI Tutorial: Complex cell net

2) Add 2 populations of 3 cells :
- 'E' (excitatory) of cell type 'PT' (pyramidal-tract corticospinal)
- 'I' (inhibitory) of cell type 'FS' (fast-spiking interneuron)


Populations
Define here the populations of your network

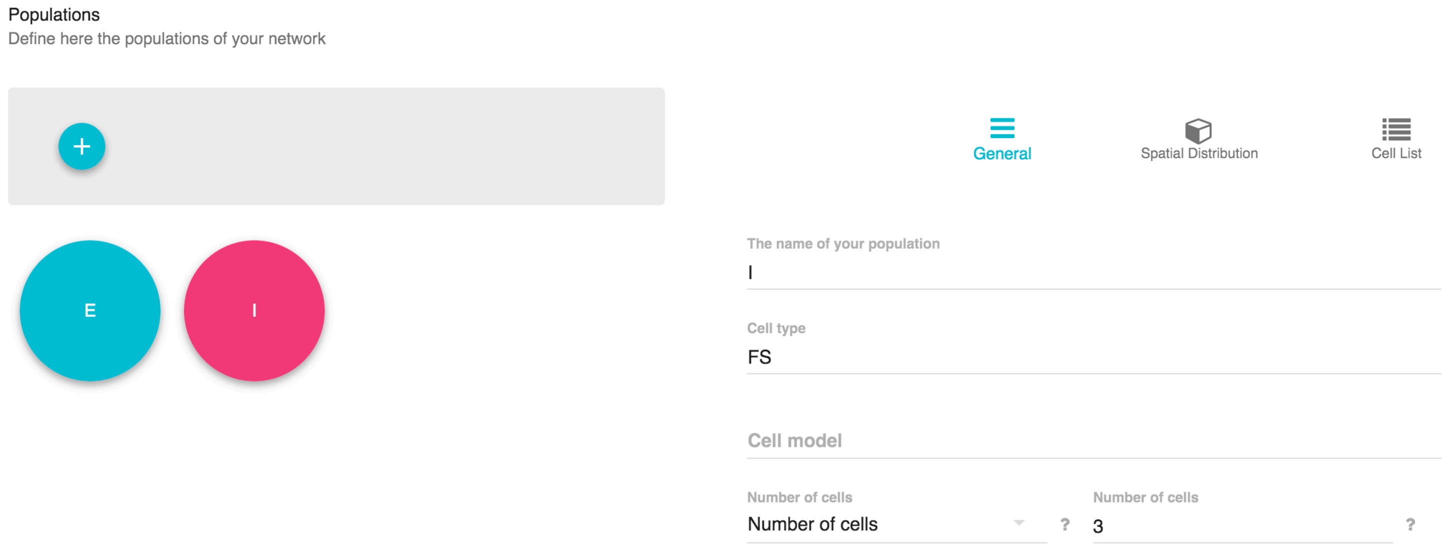
The name of your population
E

Cell type
PT

Cell model

Number of cells
Number of cells ?

Number of cells
3 ?


Populations
Define here the populations of your network

The name of your population
I

Cell type
FS

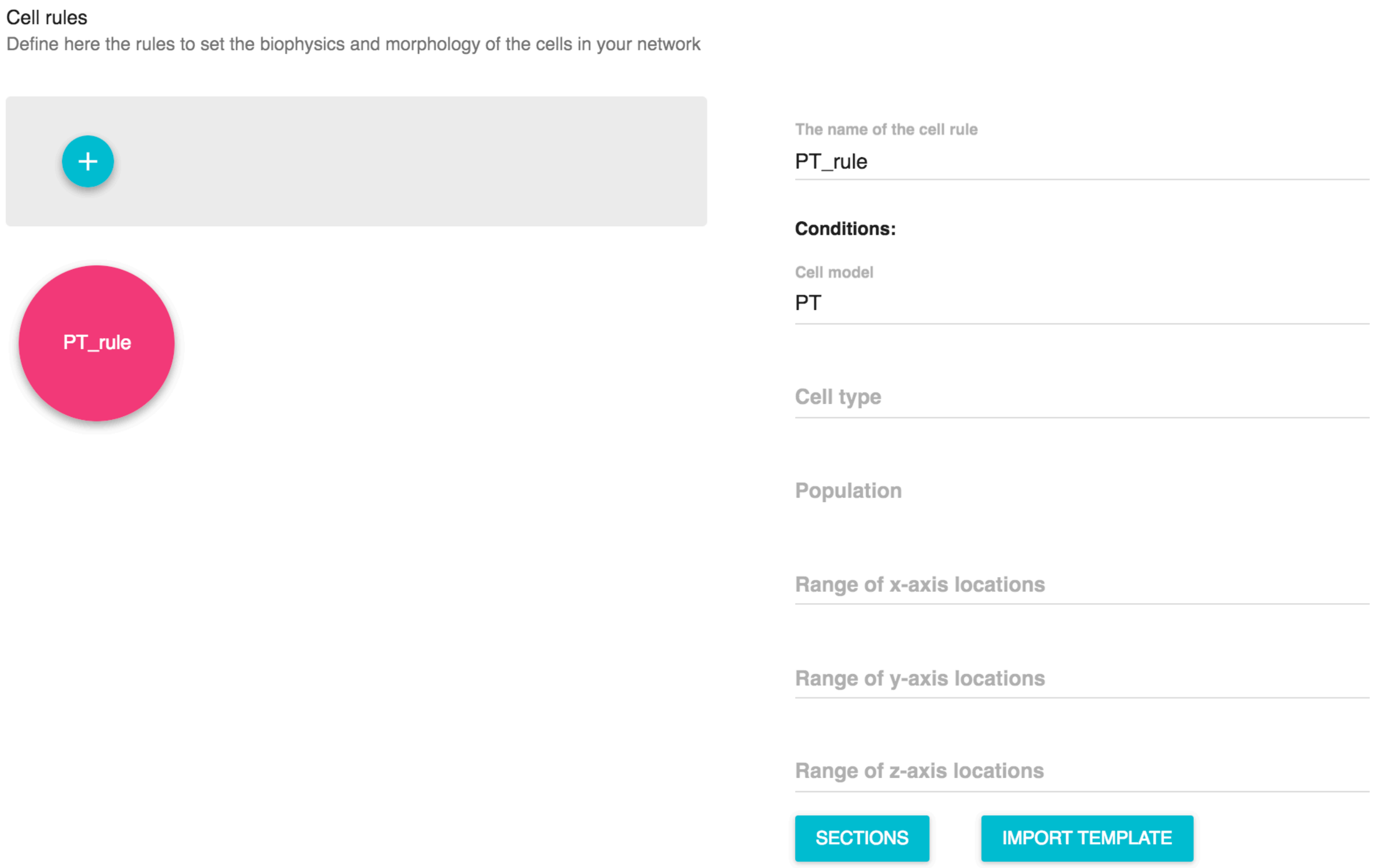
Cell model

Number of cells
Number of cells ?

Number of cells
3 ?

NetPyNE GUI Tutorial: Complex cell net

3a) Create PT cell rule



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

The name of the cell rule

PT_rule

Conditions:

Cell model

PT

Cell type

Population

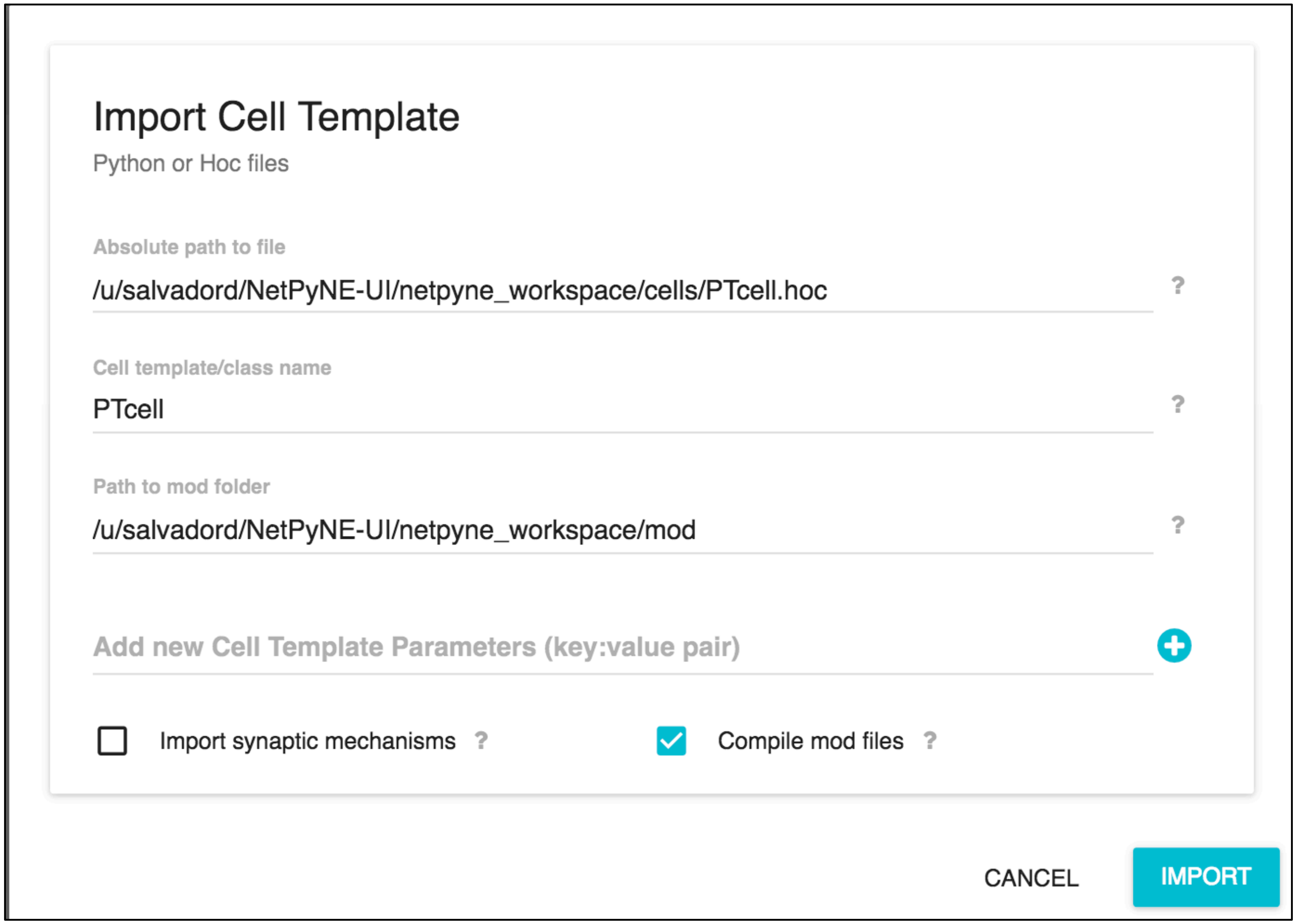
Range of x-axis locations

Range of y-axis locations

Range of z-axis locations

NetPyNE GUI Tutorial: Complex cell net

3b) Import PT cell from template (PTcell.hoc)



Import Cell Template

Python or Hoc files

Absolute path to file: /u/salvadord/NetPyNE-UI/netpyne_workspace/cells/PTcell.hoc

Cell template/class name: PTcell

Path to mod folder: /u/salvadord/NetPyNE-UI/netpyne_workspace/mod

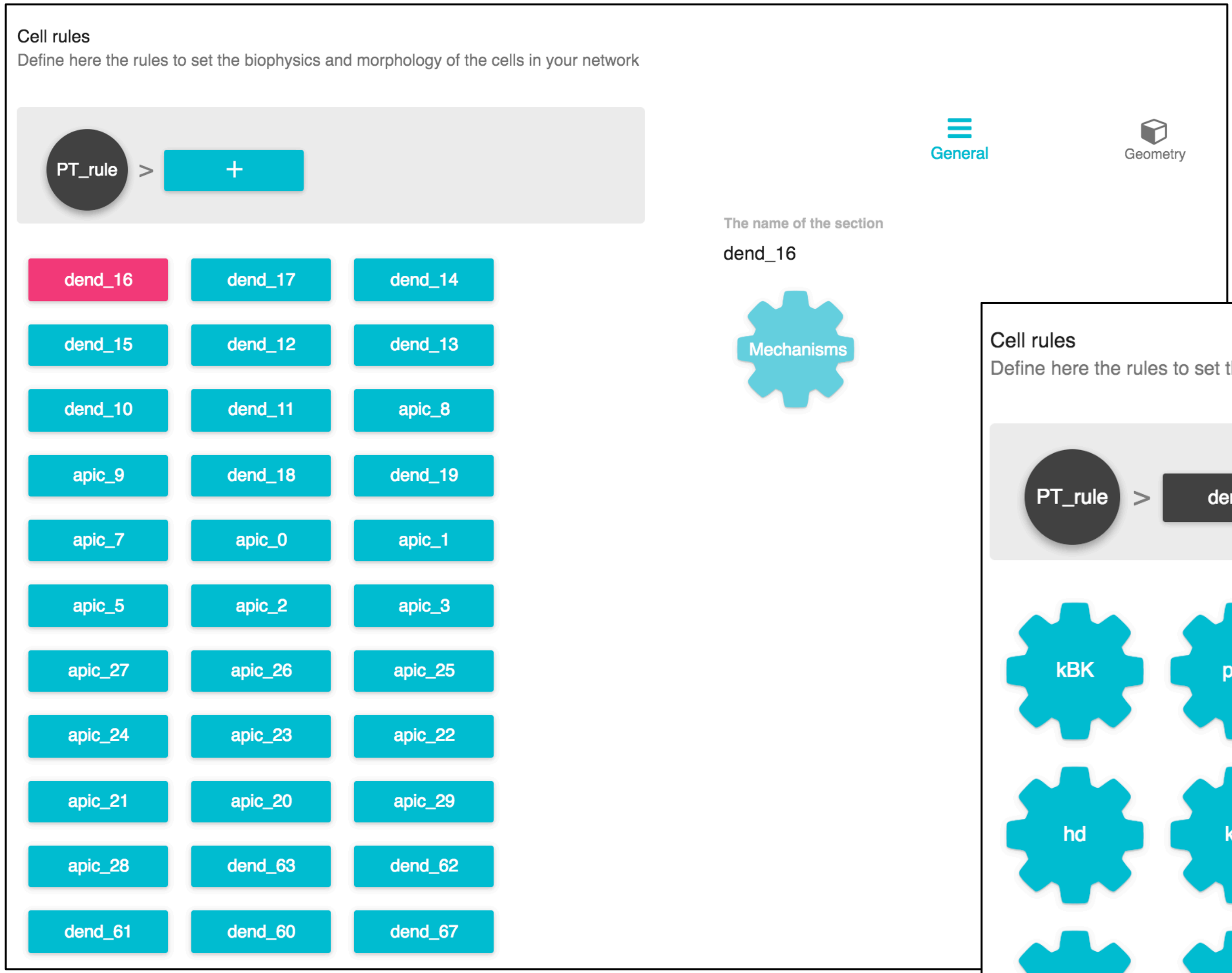
Add new Cell Template Parameters (key:value pair)

☐ Import synaptic mechanisms ?   ☑ Compile mod files ?

CANCEL   IMPORT

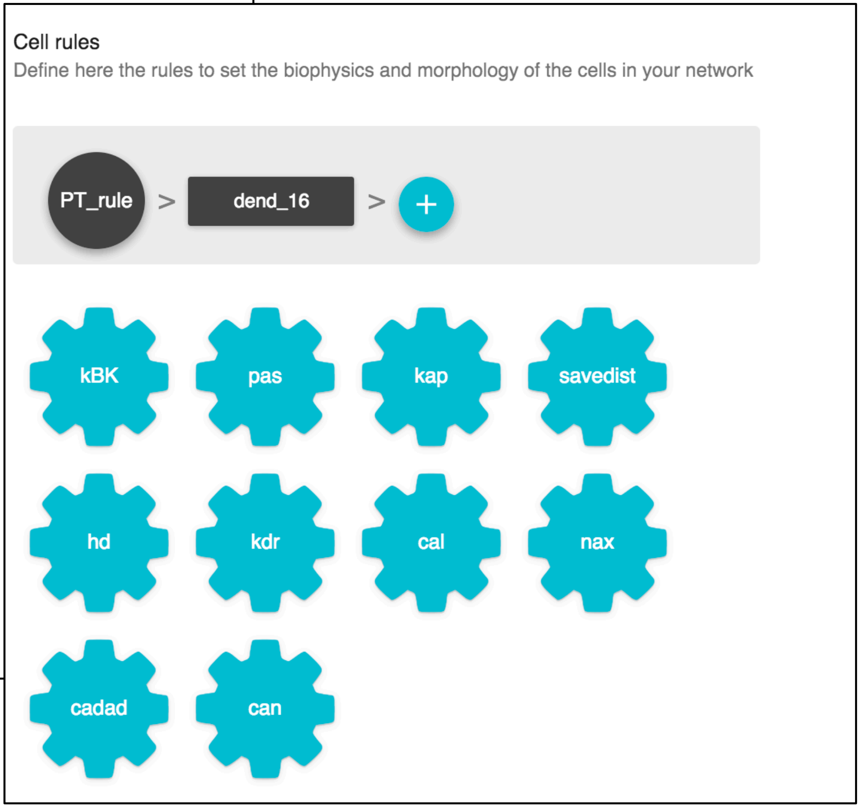
NetPyNE GUI Tutorial: Complex cell net

3c) Check sections and mechanisms imported



Cell rules
Define here the rules to set the biophysics and morphology of the cells in your network

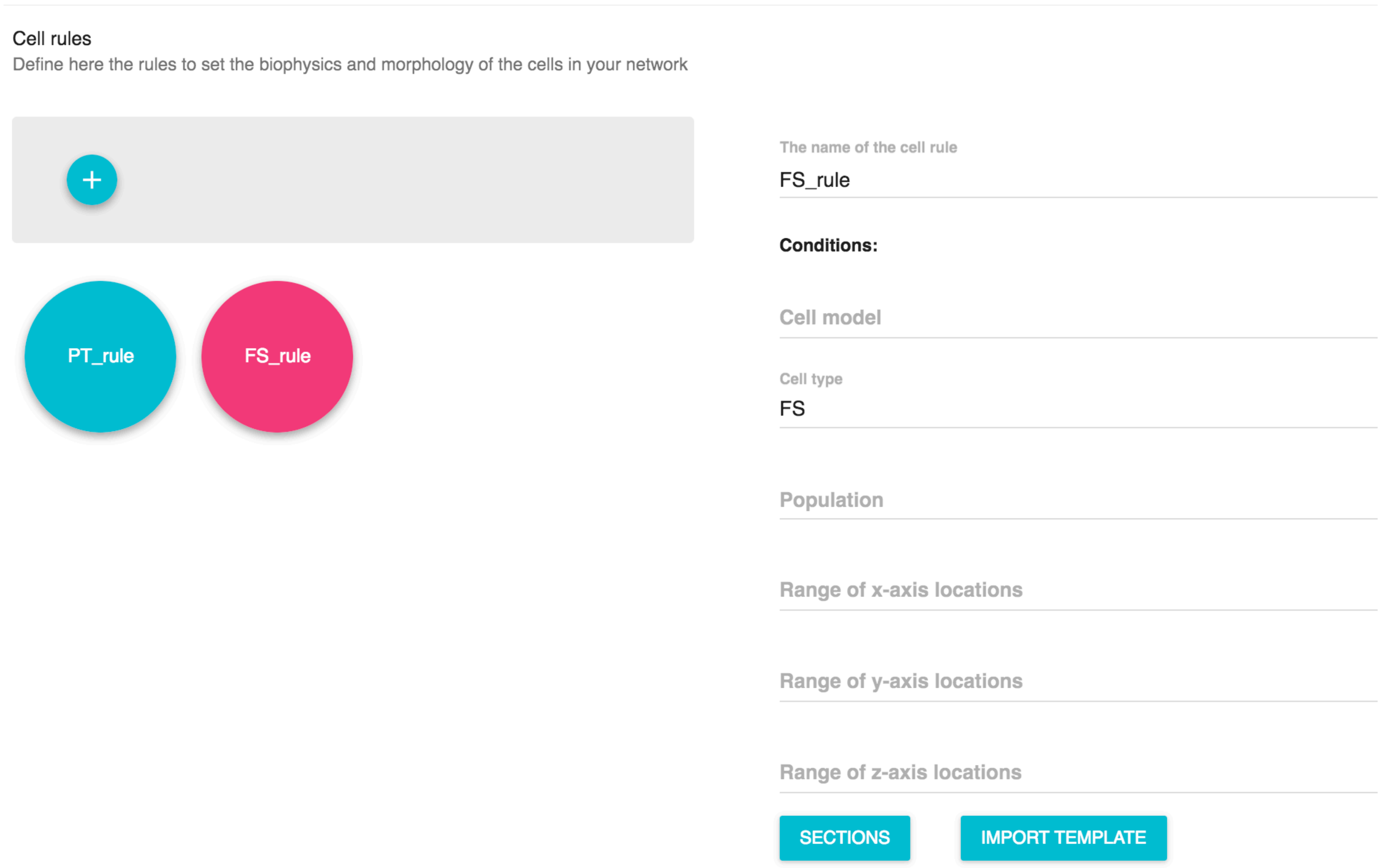
The name of the section
dend_16



Cell rules
Define here the rules to set the biophysics and morphology of the cells in your network

NetPyNE GUI Tutorial: Complex cell net

4a) Create FS rule



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

The name of the cell rule: FS_rule

Conditions:

Cell model

Cell type: FS

Population

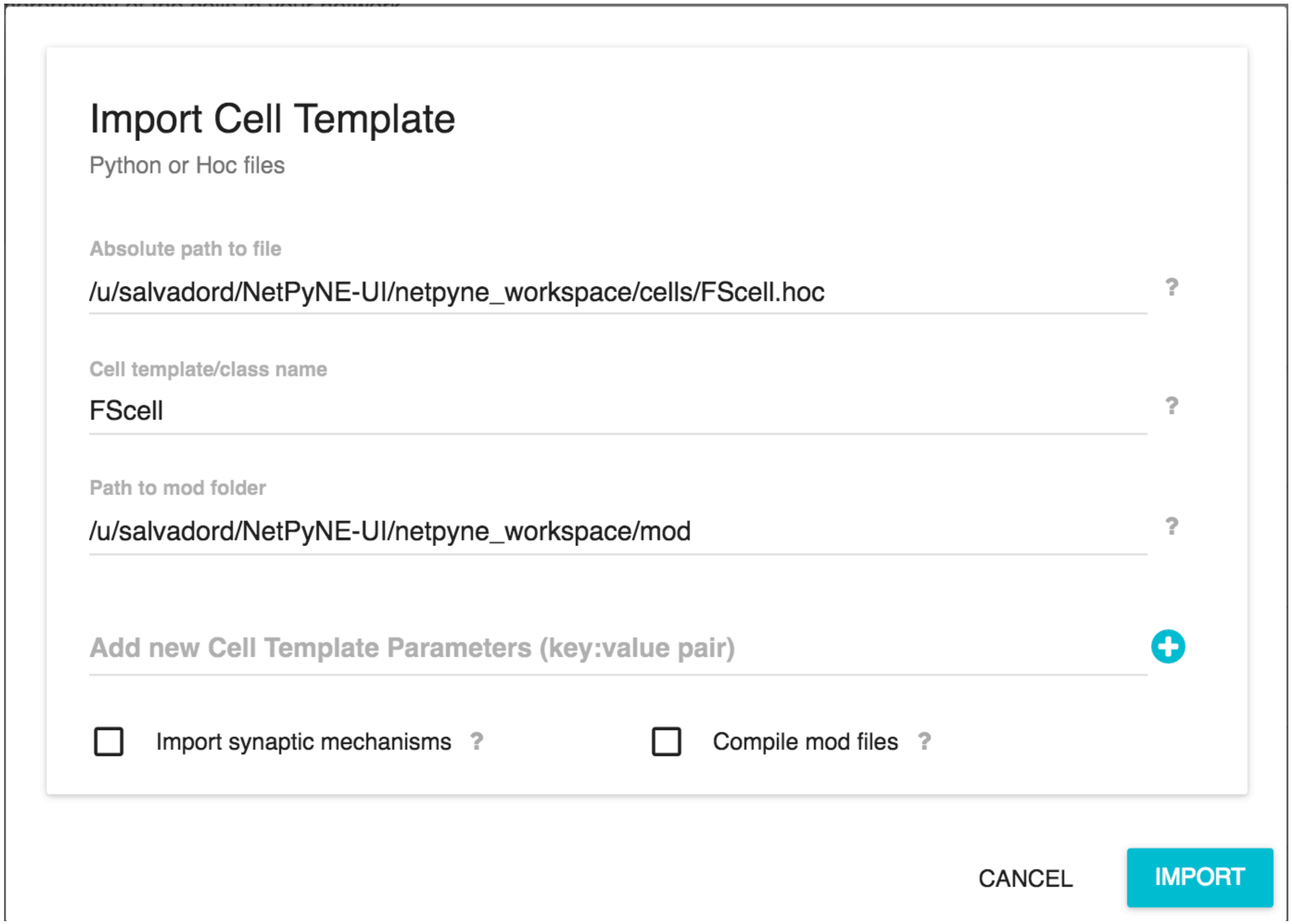
Range of x-axis locations

Range of y-axis locations

Range of z-axis locations

NetPyNE GUI Tutorial: Complex cell net

4b) Import FS cell from template (FScell.hoc)



Import Cell Template

Python or Hoc files

Absolute path to file: /u/salvadord/NetPyNE-UI/netpyne_workspace/cells/FScell.hoc

Cell template/class name: FScell

Path to mod folder: /u/salvadord/NetPyNE-UI/netpyne_workspace/mod

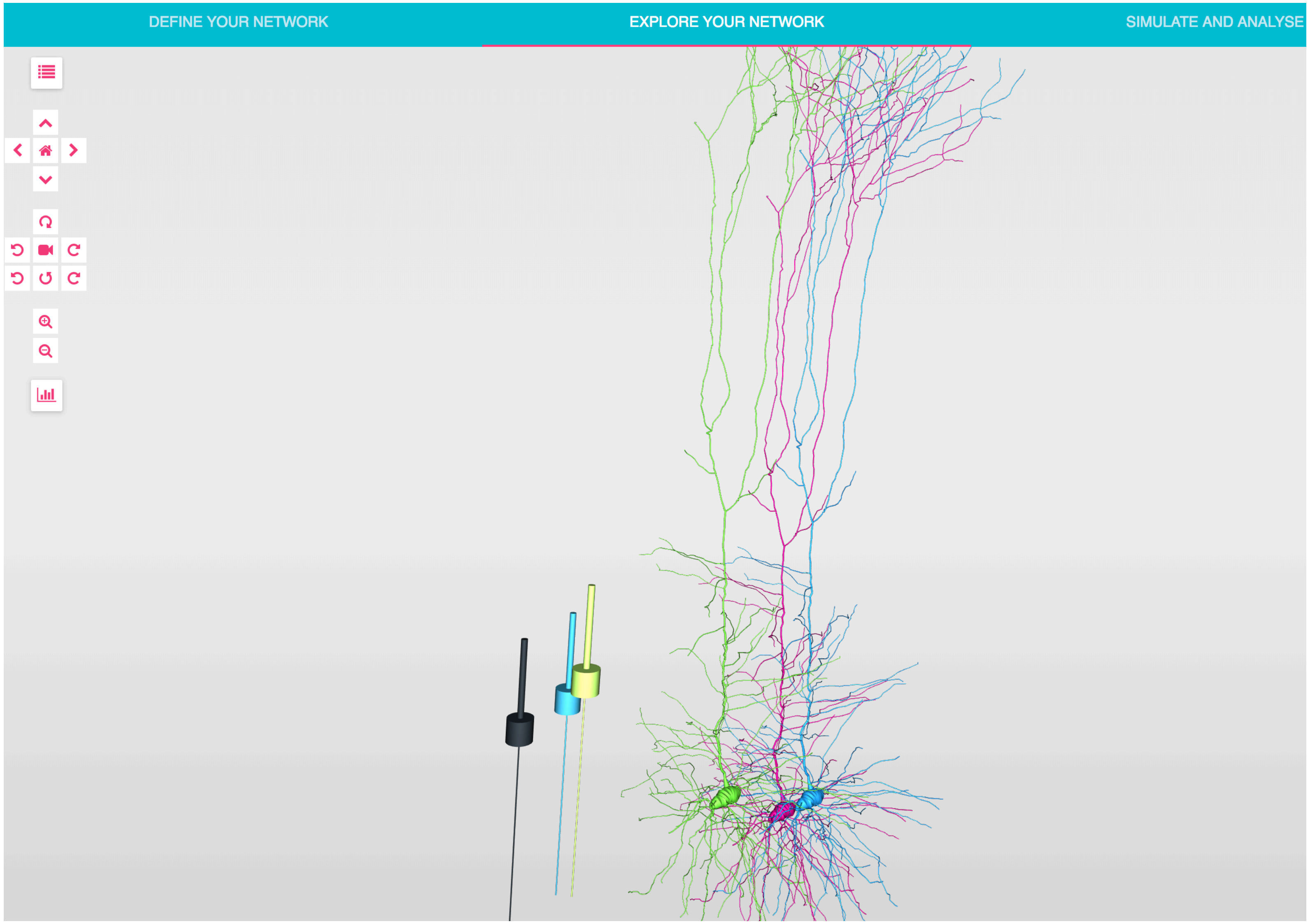
Add new Cell Template Parameters (key:value pair)

☐ Import synaptic mechanisms ?

☐ Compile mod files ?

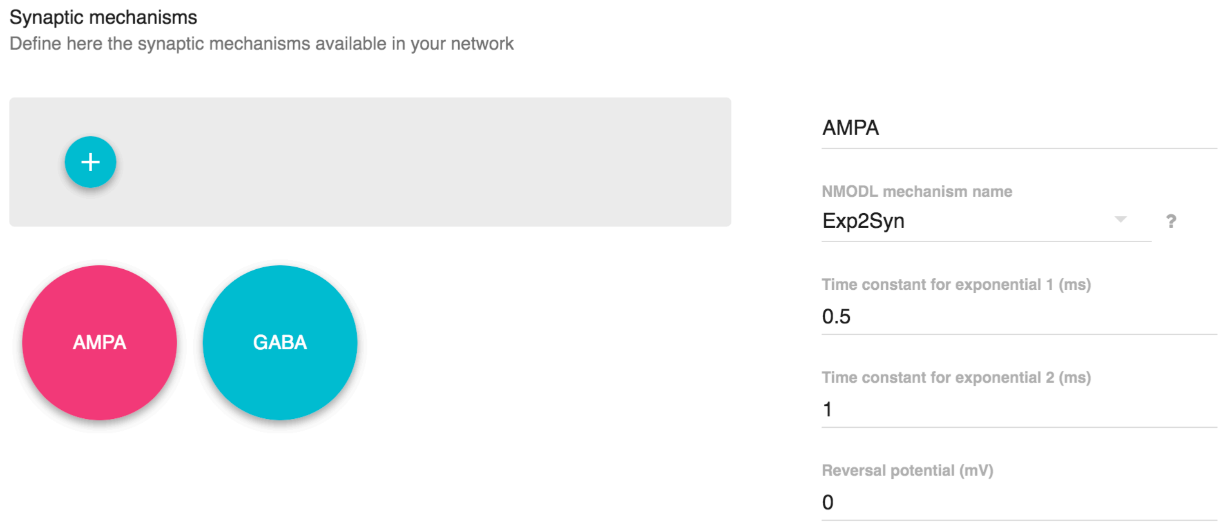
NetPyNE GUI Tutorial: Complex cell net

5) Visualize network



NetPyNE GUI Tutorial: Complex cell net

6) Add AMPA and GABA synapses



Synaptic mechanisms
Define here the synaptic mechanisms available in your network

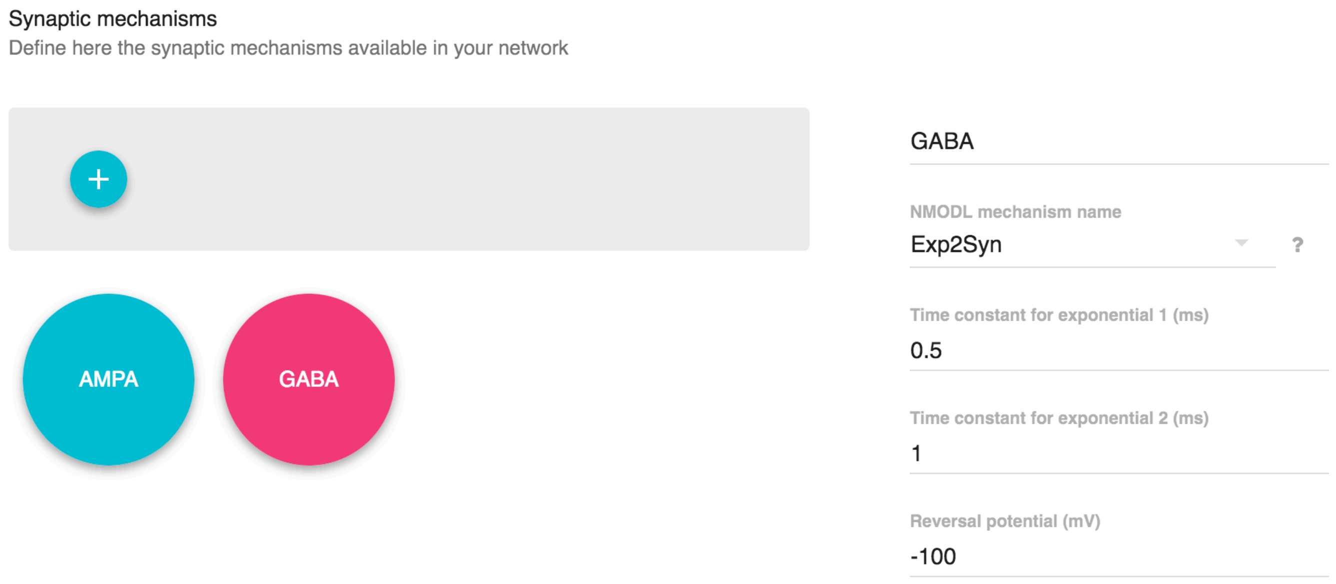
AMPA

NMODL mechanism name
Exp2Syn

Time constant for exponential 1 (ms)
0.5

Time constant for exponential 2 (ms)
1

Reversal potential (mV)
0



Synaptic mechanisms
Define here the synaptic mechanisms available in your network

GABA

NMODL mechanism name
Exp2Syn

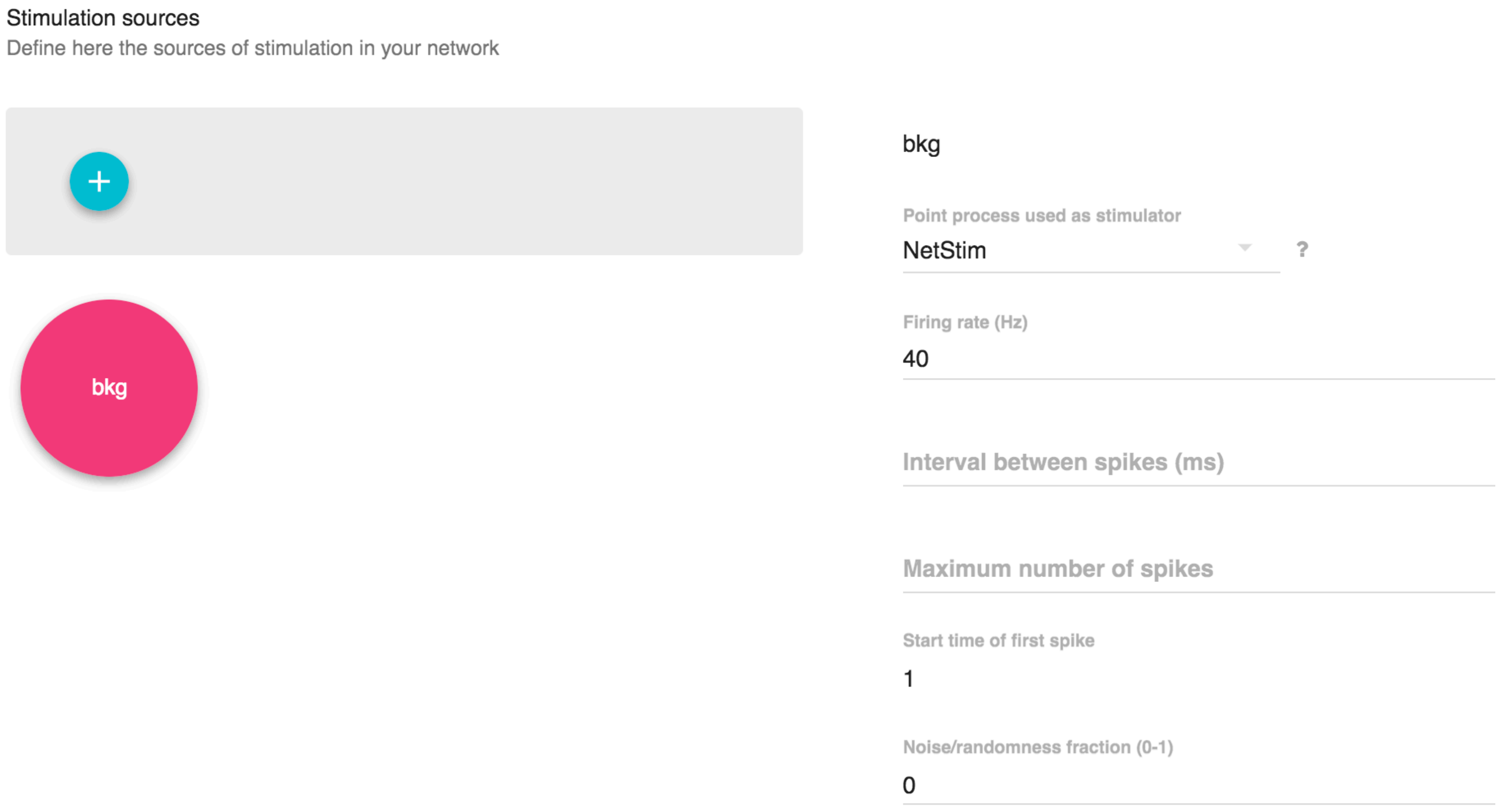
Time constant for exponential 1 (ms)
0.5

Time constant for exponential 2 (ms)
1

Reversal potential (mV)
-100

NetPyNE GUI Tutorial: Complex cell net

7a) Add background stimulation Netstim (spike generator) to PT cells



Stimulation sources

Define here the sources of stimulation in your network

bkg

Point process used as stimulator
NetStim

Firing rate (Hz)
40

Interval between spikes (ms)

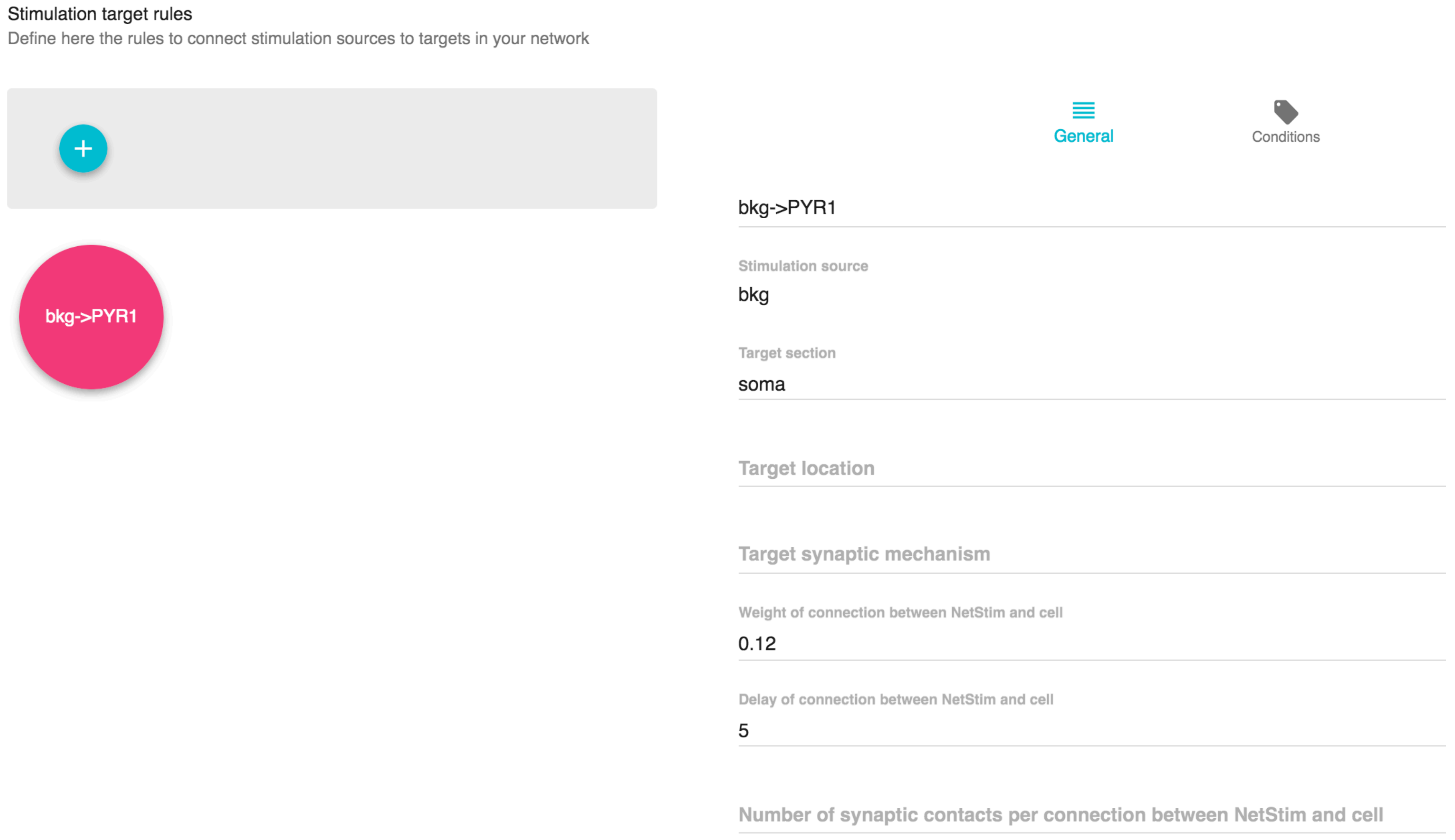
Maximum number of spikes

Start time of first spike
1

Noise/randomness fraction (0-1)
0

NetPyNE GUI Tutorial: Complex cell net

7b) Add background stimulation Netstim (spike generator) to PT cells



Stimulation target rules

Define here the rules to connect stimulation sources to targets in your network

bkg->PYR1

Stimulation source
bkg

Target section
soma

Target location

Target synaptic mechanism

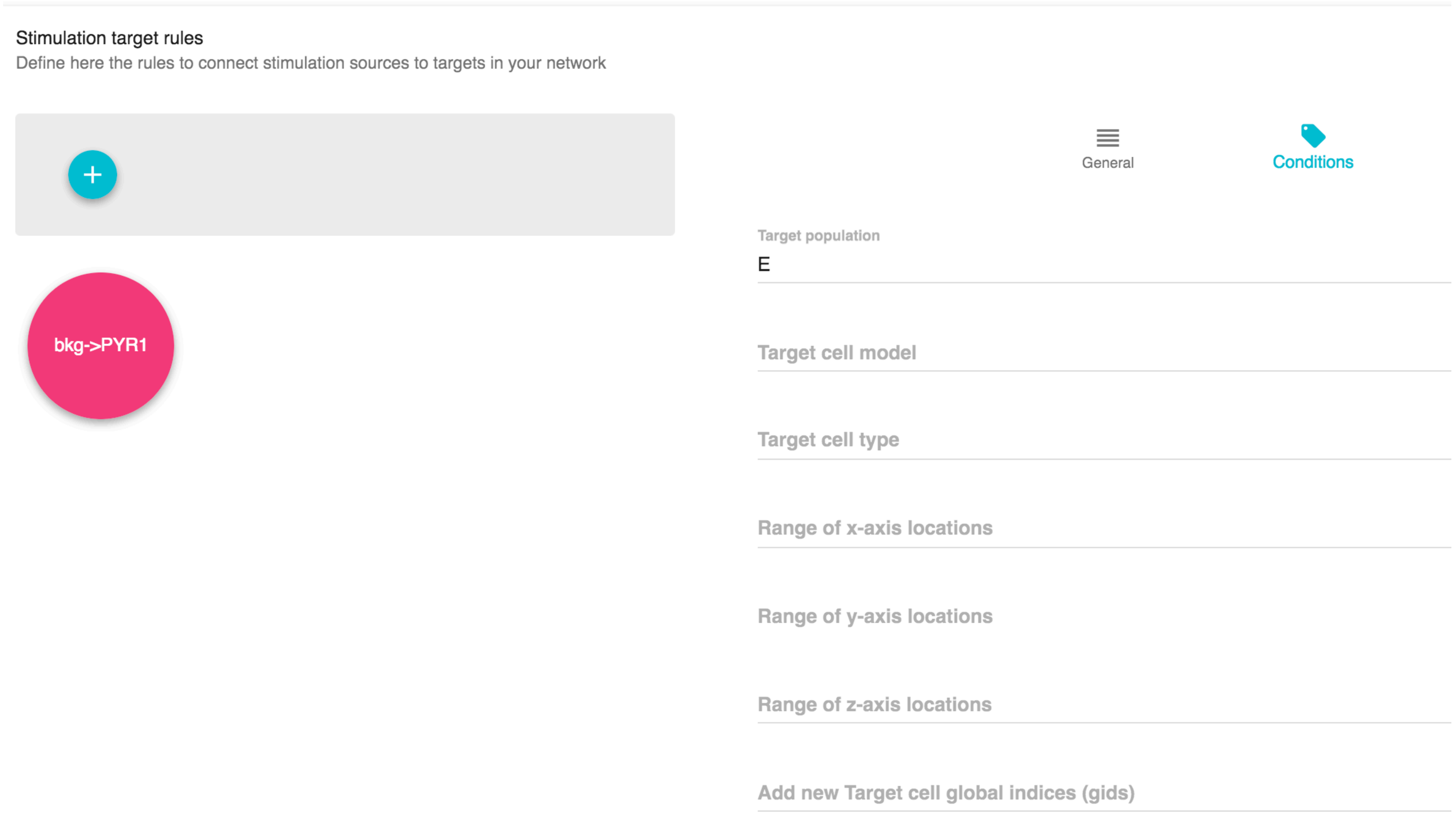
Weight of connection between NetStim and cell
0.12

Delay of connection between NetStim and cell
5

Number of synaptic contacts per connection between NetStim and cell

NetPyNE GUI Tutorial: Complex cell net

7c) Add background stimulation Netstim (spike generator) to PT cells (E population)



Stimulation target rules

Define here the rules to connect stimulation sources to targets in your network

Target population

E

Target cell model

Target cell type

Range of x-axis locations

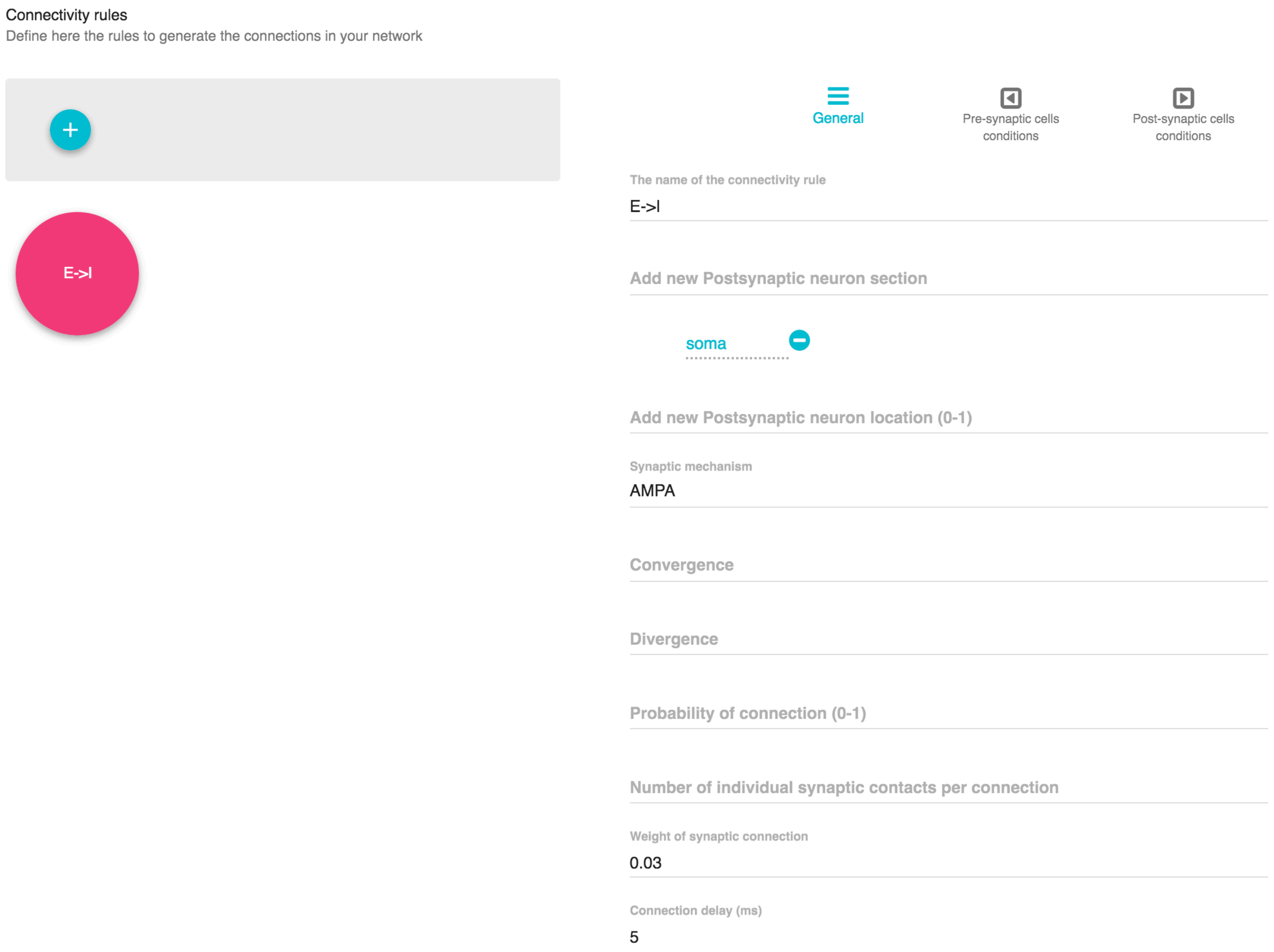
Range of y-axis locations

Range of z-axis locations

Add new Target cell global indices (gids)

NetPyNE GUI Tutorial: Complex cell net

8a) Connect E->I



Connectivity rules

Define here the rules to generate the connections in your network

The name of the connectivity rule: E->I

Add new Postsynaptic neuron section

soma

Add new Postsynaptic neuron location (0-1)

Synaptic mechanism: AMPA

Convergence

Divergence

Probability of connection (0-1)

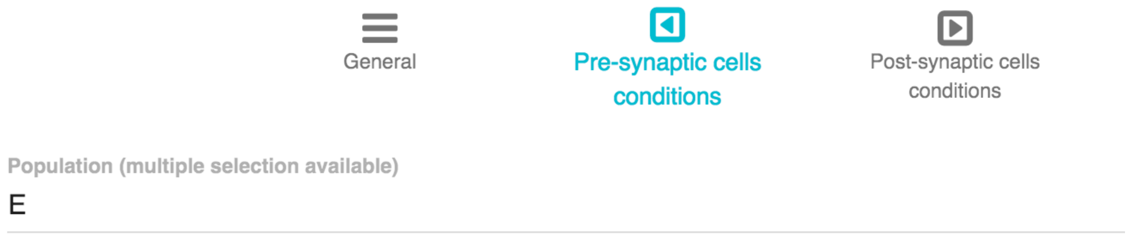
Number of individual synaptic contacts per connection

Weight of synaptic connection: 0.03

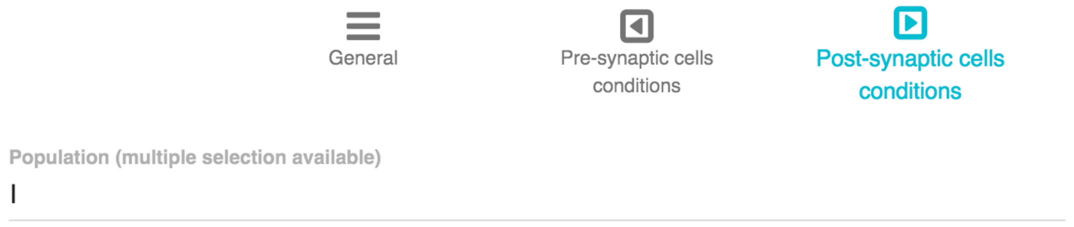
Connection delay (ms): 5

NetPyNE GUI Tutorial: Complex cell net

8b) Connect E->I



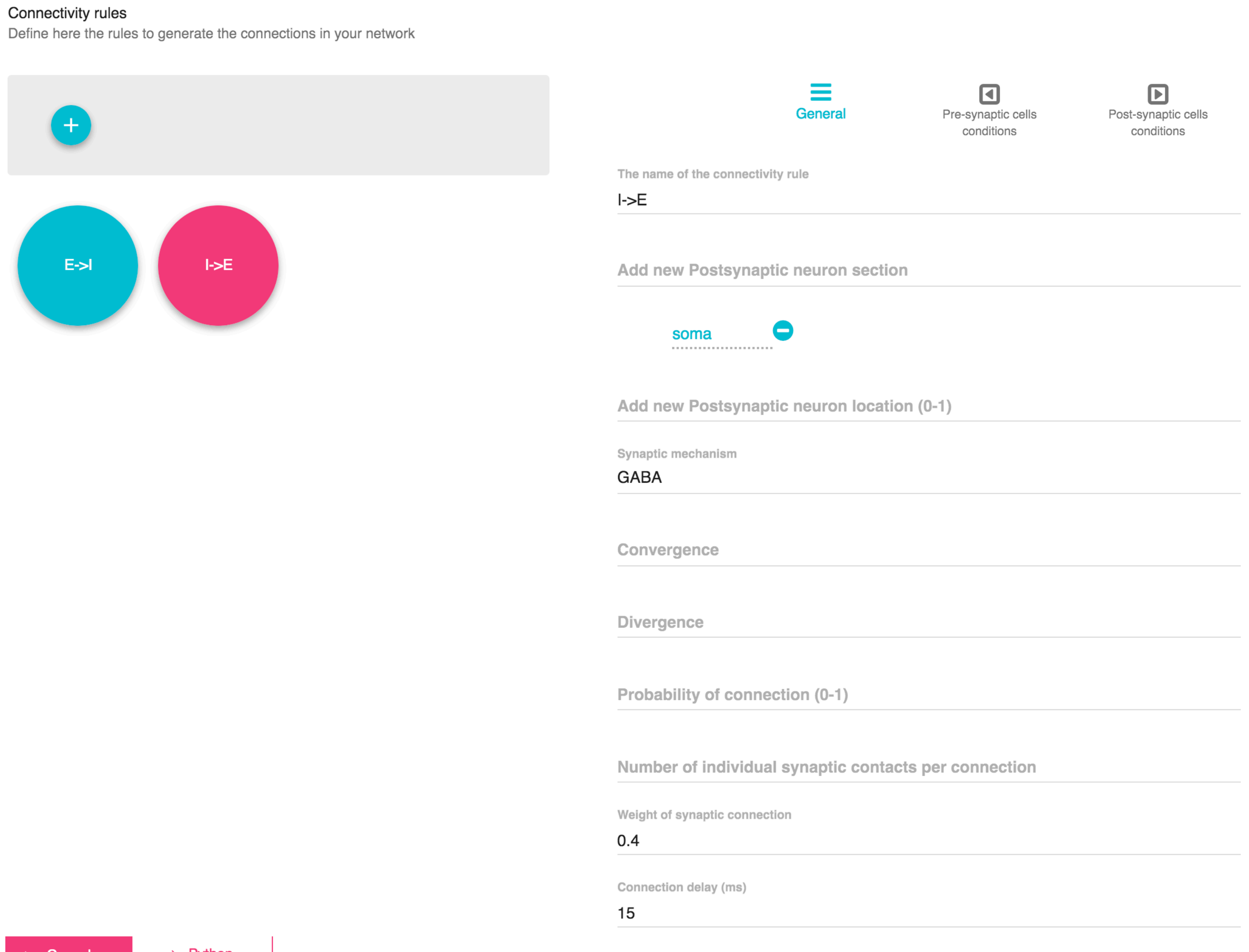
Population (multiple selection available)
E



Population (multiple selection available)
I

NetPyNE GUI Tutorial: Complex cell net

9a) Connect I->E



Connectivity rules

Define here the rules to generate the connections in your network

The name of the connectivity rule: I->E

Add new Postsynaptic neuron section

soma

Add new Postsynaptic neuron location (0-1)

Synaptic mechanism: GABA

Convergence

Divergence

Probability of connection (0-1)

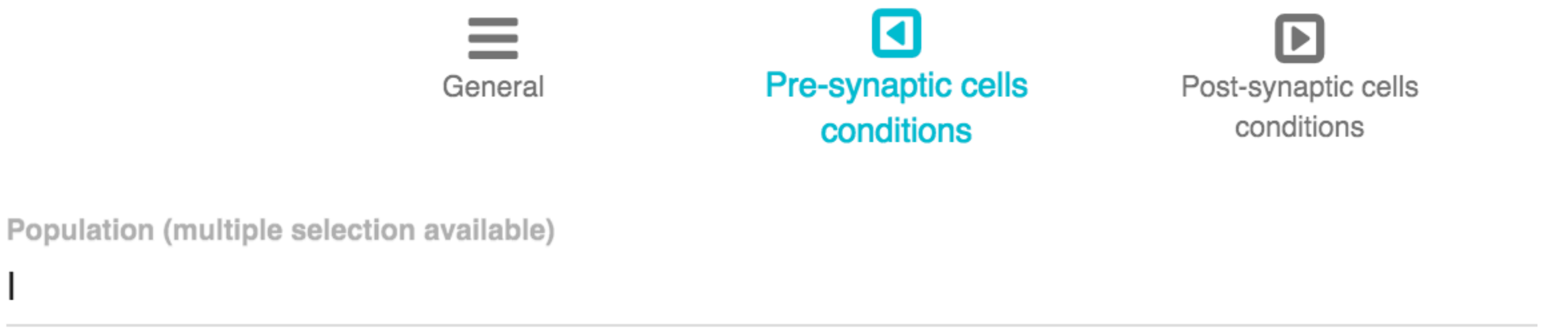
Number of individual synaptic contacts per connection

Weight of synaptic connection: 0.4

Connection delay (ms): 15

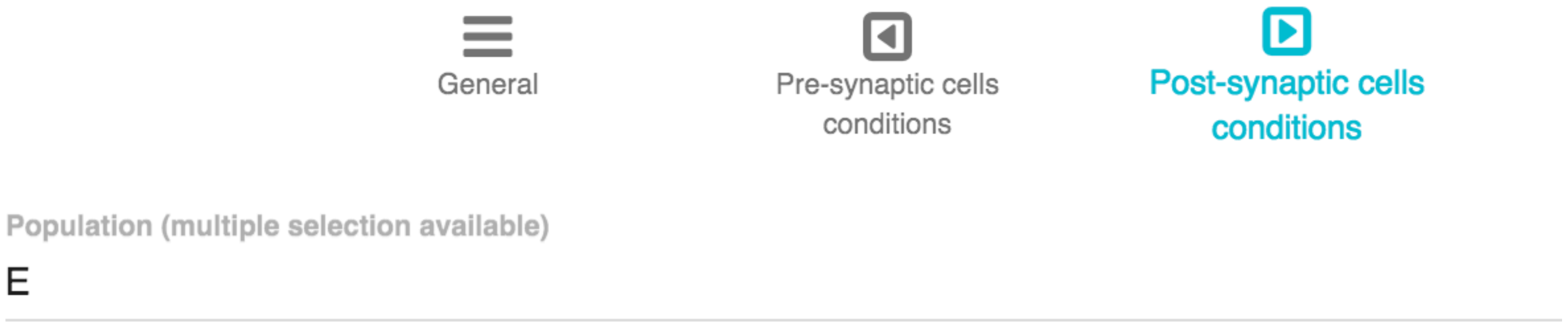
NetPyNE GUI Tutorial: Complex cell net

9b) Connect I->E



Population (multiple selection available)

I

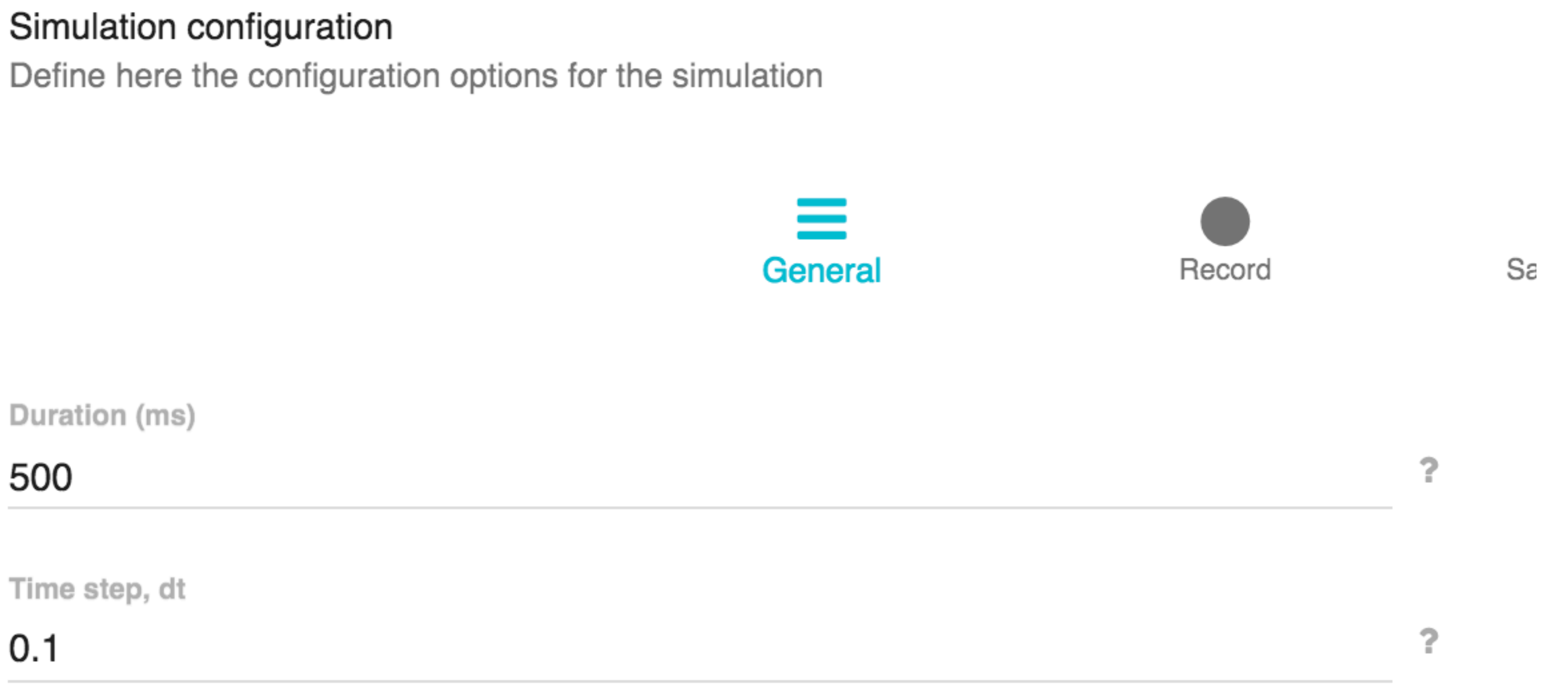


Population (multiple selection available)

E

NetPyNE GUI Tutorial: Complex cell net

10) Set duration to 500 ms and time step to 0.1 (if too slow can decrease duration)



Simulation configuration

Define here the configuration options for the simulation

Duration (ms)

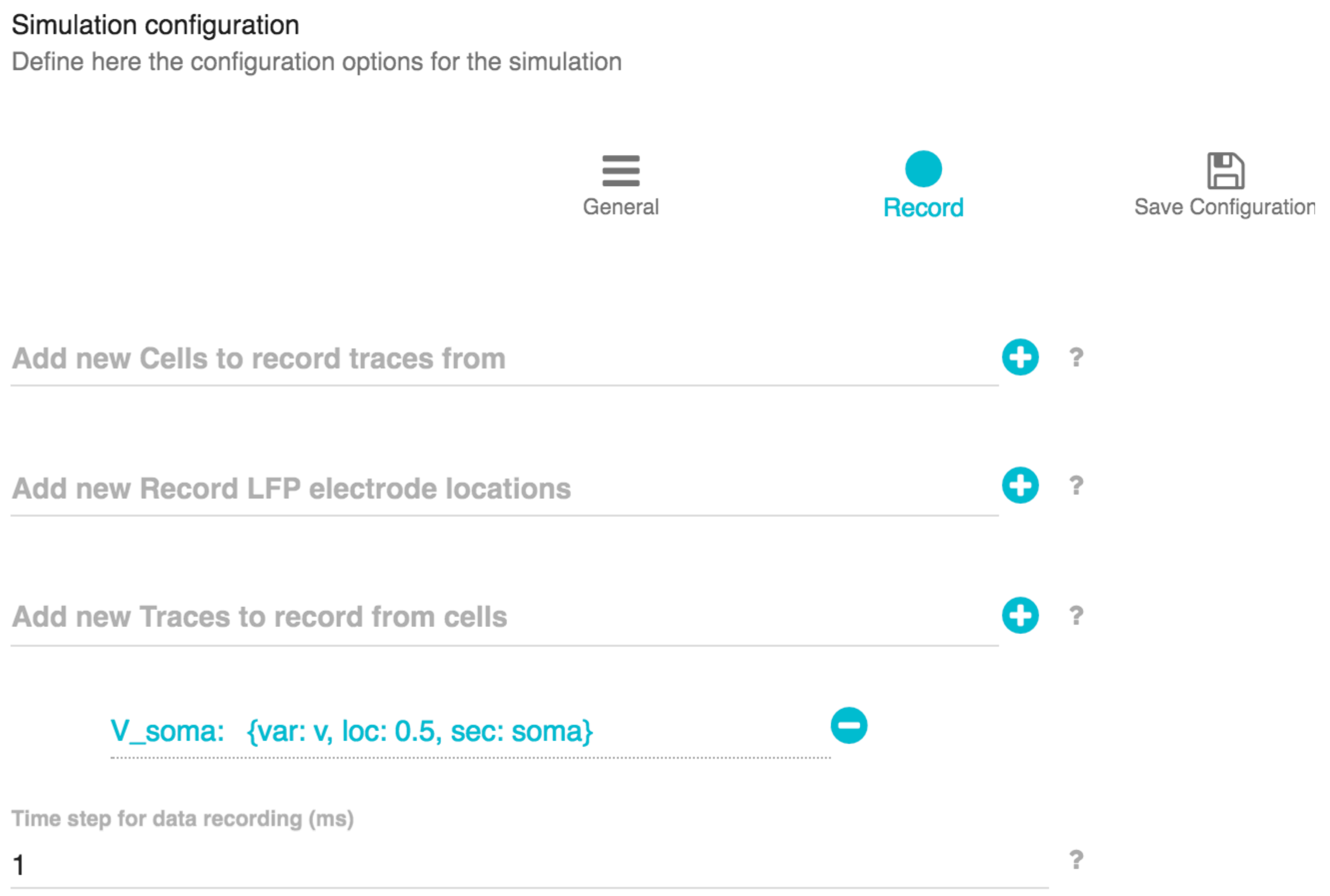
500

Time step, dt

0.1

NetPyNE GUI Tutorial: Complex cell net

11) Record voltate trace from soma


Simulation configuration
Define here the configuration options for the simulation

Add new Cells to record traces from

Add new Record LFP electrode locations

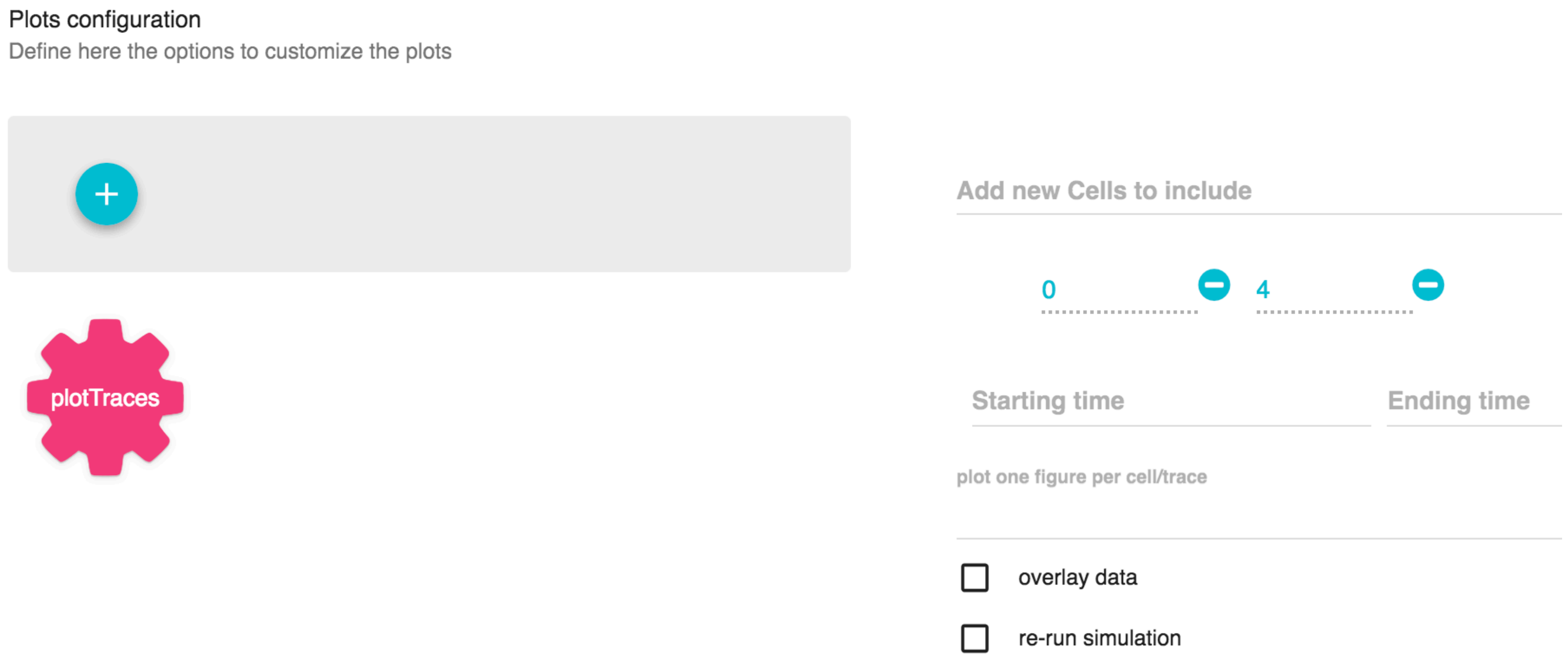
Add new Traces to record from cells

V_soma: {var: v, loc: 0.5, sec: soma}

Time step for data recording (ms)
1

NetPyNE GUI Tutorial: Complex cell net

12) Configure traces plot to include cells 0 (PT) and 4 (FS)


Plots configuration

Define here the options to customize the plots

Add new Cells to include

0

4

Starting time

Ending time

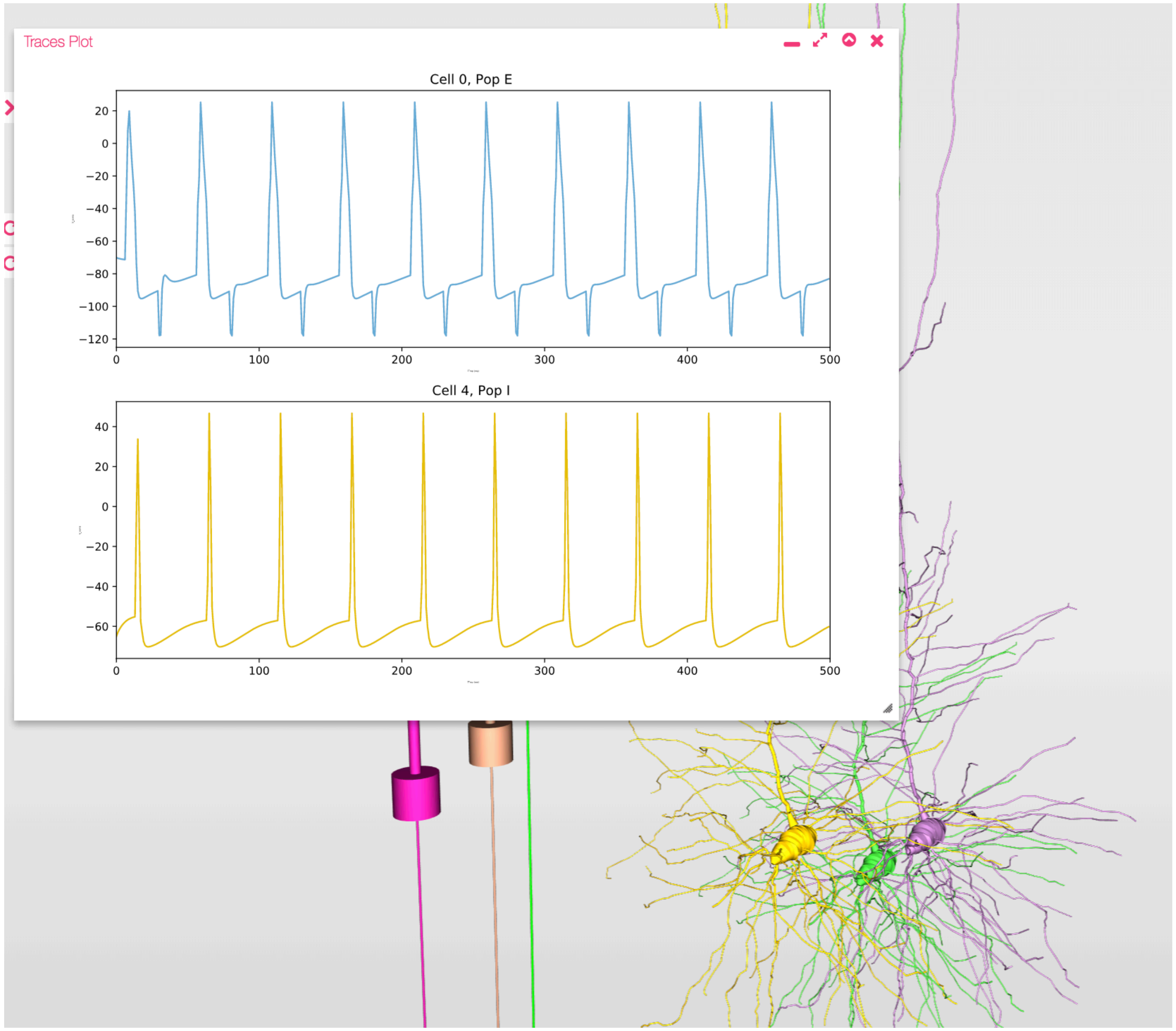
plot one figure per cell/trace

☐ overlay data

☐ re-run simulation

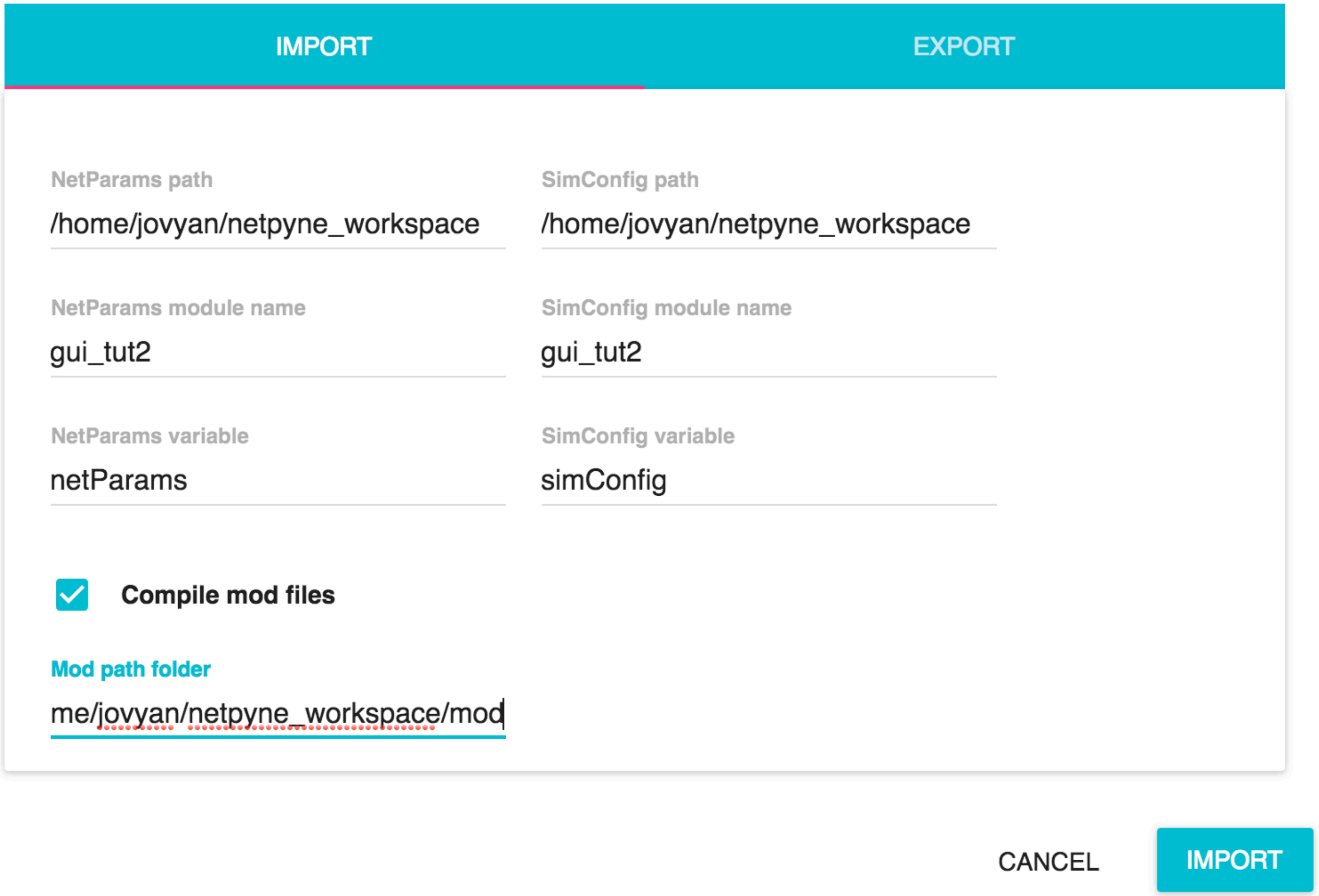
NetPyNE GUI Tutorial: Complex cell net

13) Simulate and visualize traces (synchrony due to recurrent conns and exaggerated IPSPs)



NetPyNE GUI Tutorial: Complex cell net

Note: If you have any errors with step-by-step, try loading the **“complex cell net”** tutorial directly from file



NetParams path: /home/jovyan/netpyne_workspace

SimConfig path: /home/jovyan/netpyne_workspace

NetParams module name: gui_tut2

SimConfig module name: gui_tut2

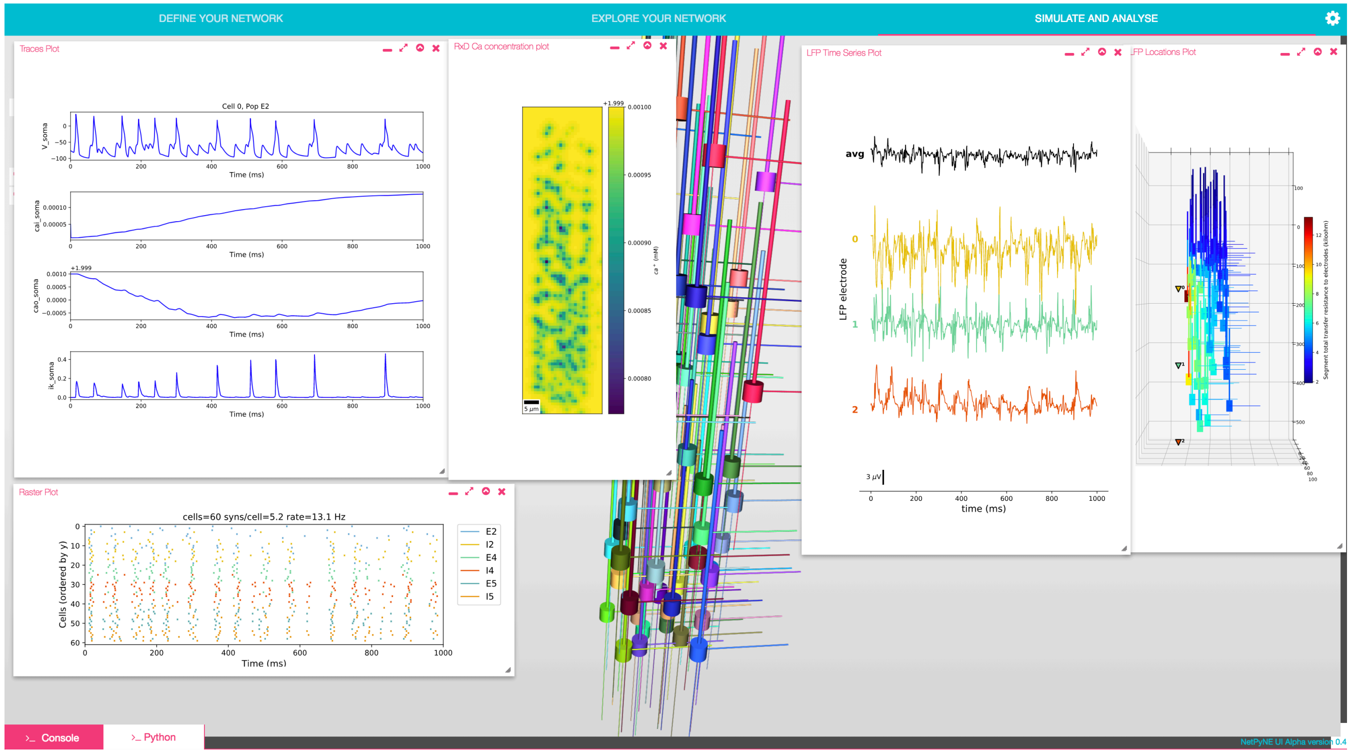
NetParams variable: netParams

SimConfig variable: simConfig

✓ Compile mod files

Mod path folder: me/jovyan/netpyne_workspace/mod

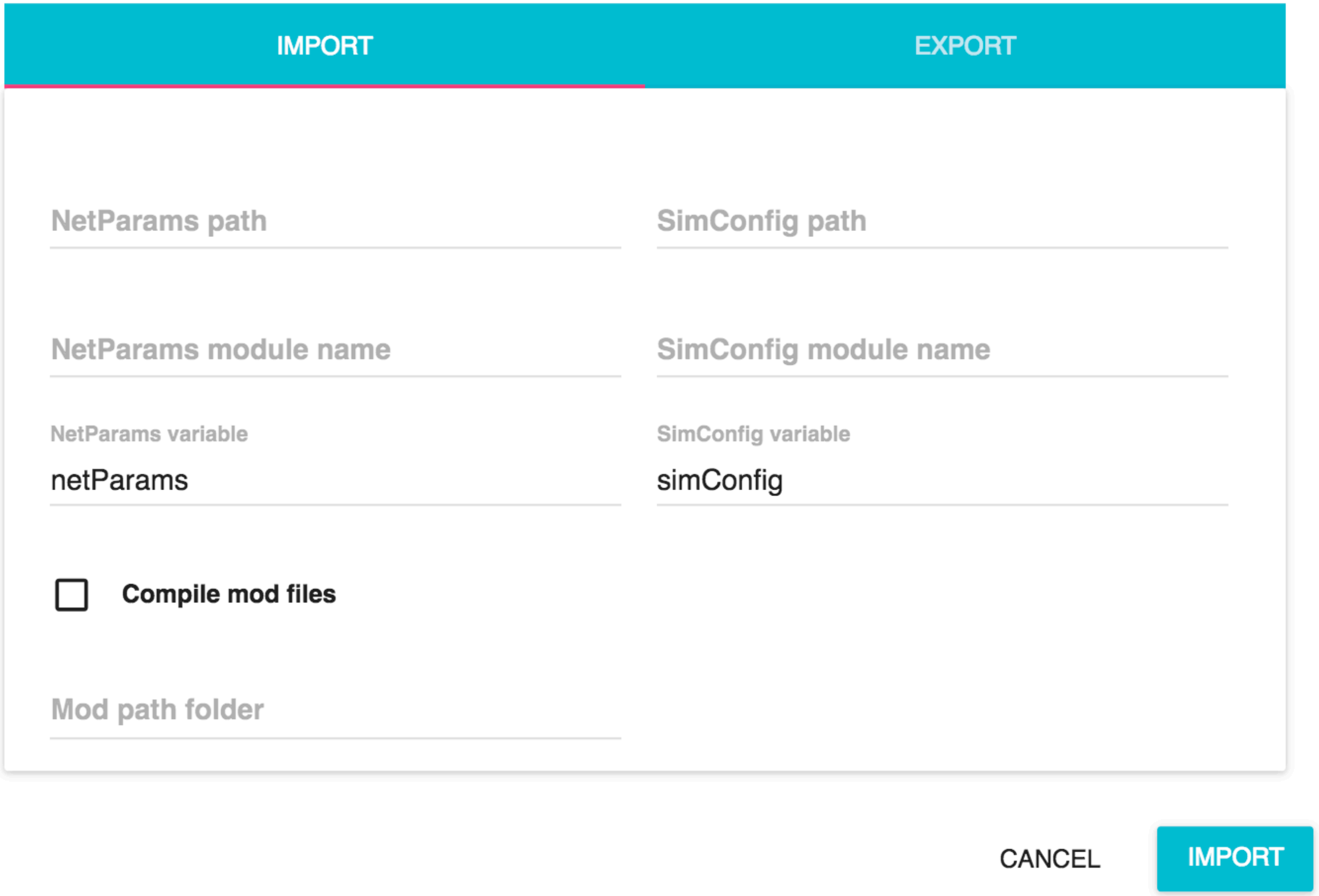
NetPyNE GUI Tutorial: Multiscale net



NetPyNE GUI Tutorial: Multiscale net

1) Load the **“multiscale net”** tutorial (gui_tut3.py) directly from file **via GUI “Import model”**:

a) Click on utilities icon in top-right of GUI to open “Import” window



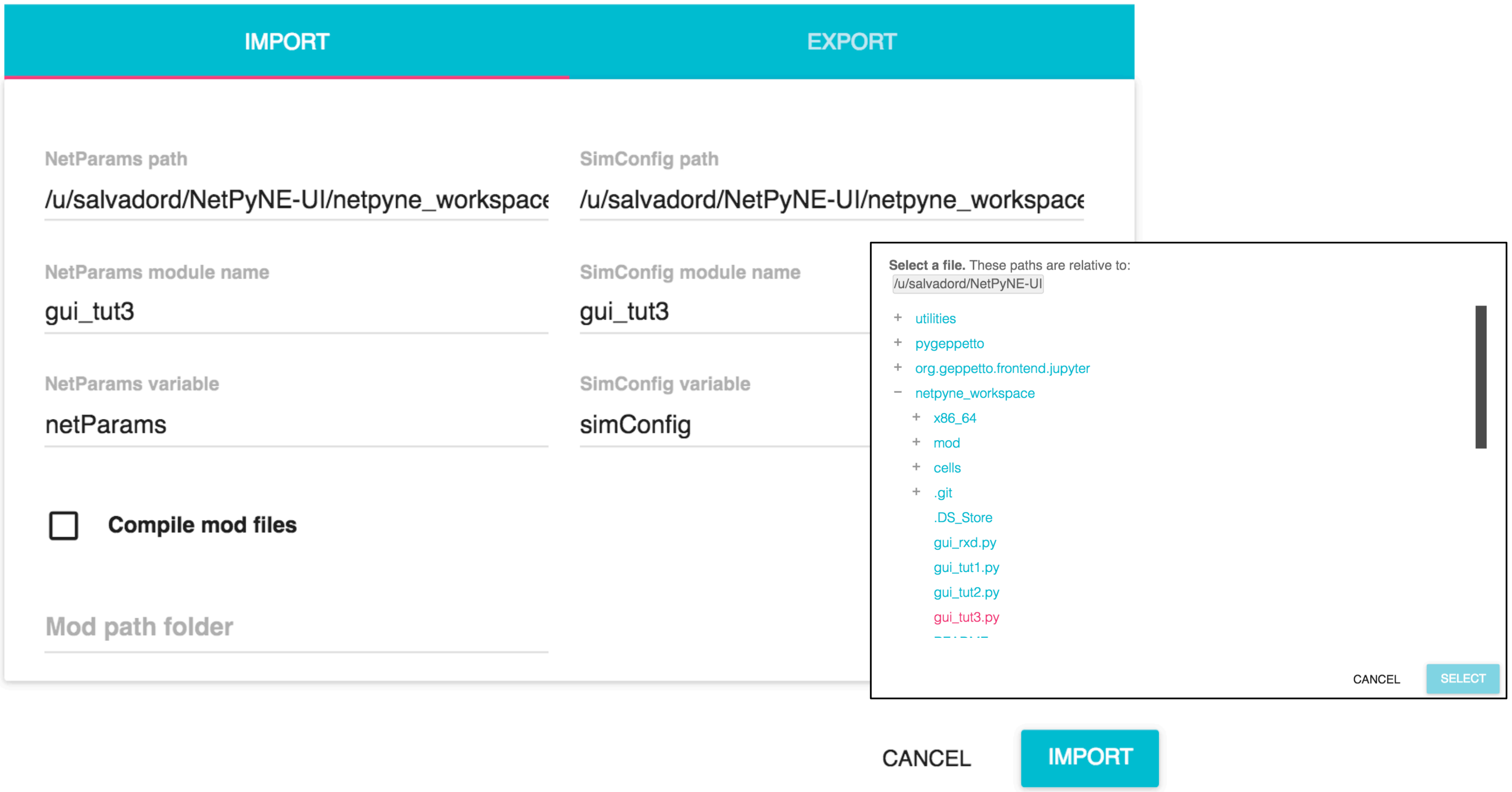
NetParams variable
netParams

SimConfig variable
simConfig

NetPyNE GUI Tutorial: Multiscale net

1) Load the **"multiscale net"** tutorial (gui_tut3.py) directly from file **via GUI "Import model"**:

b) Select gui_tut3.py in "NetParams path" and "SimConfig path"



NetParams path: /u/salvadord/NetPyNE-UI/netpyne_workspace

SimConfig path: /u/salvadord/NetPyNE-UI/netpyne_workspace

NetParams module name: gui_tut3

SimConfig module name: gui_tut3

NetParams variable: netParams

SimConfig variable: simConfig

Compile mod files

Mod path folder

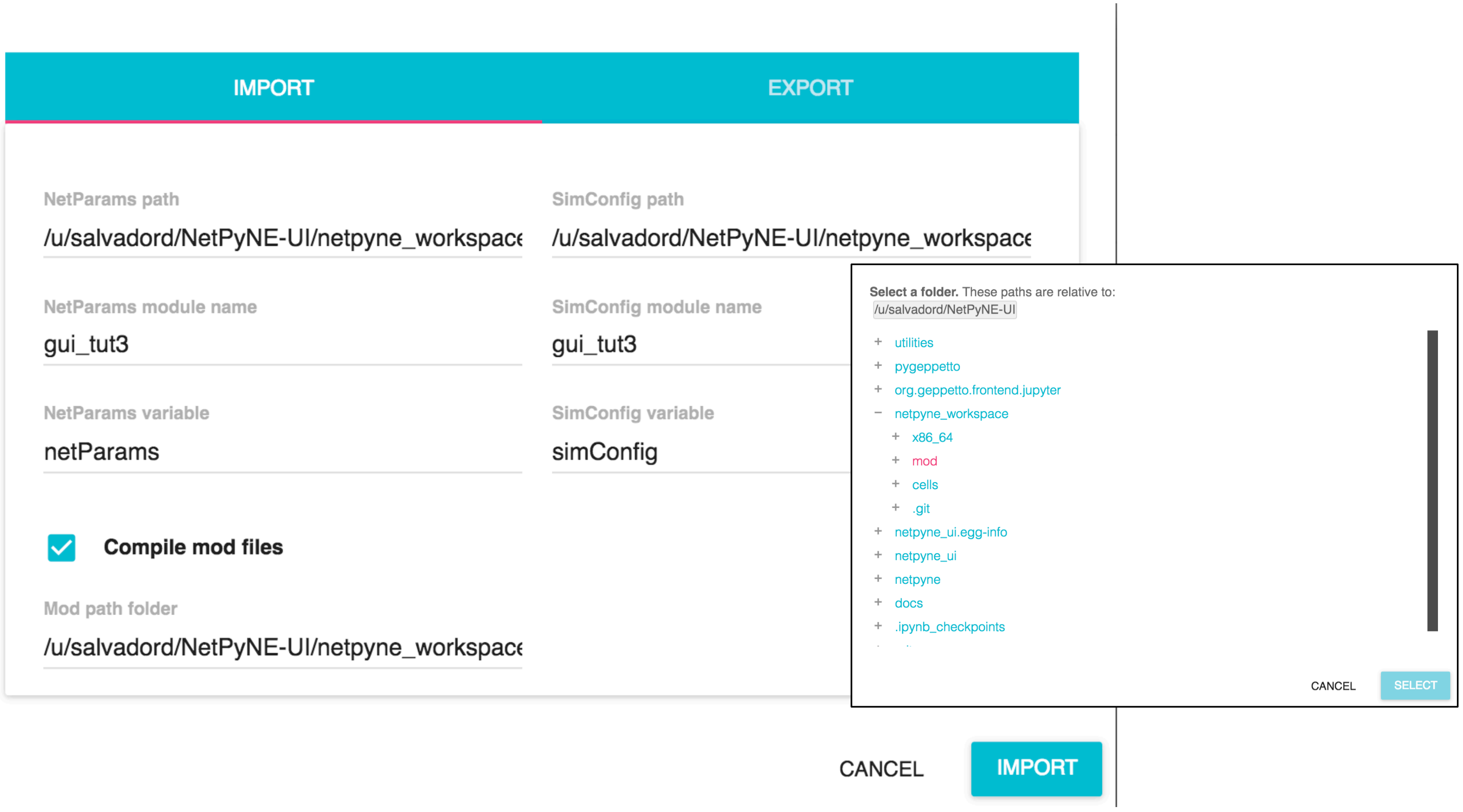
Select a file. These paths are relative to: /u/salvadord/NetPyNE-UI

```
+ utilities
+ pygeppetto
+ org.geppetto.frontend.jupyter
− netpyne_workspace
  + x86_64
  + mod
  + cells
  + .git
    .DS_Store
    gui_rxd.py
    gui_tut1.py
    gui_tut2.py
    gui_tut3.py
```

NetPyNE GUI Tutorial: Multiscale net

1) Load the **“multiscale net”** tutorial (gui_tut3.py) directly from file **via GUI “Import model”**:

c) Select “Compile mod files” and “mod” in “Mod path folder”



NetParams path
/u/salvadord/NetPyNE-UI/netpyne_workspace

SimConfig path
/u/salvadord/NetPyNE-UI/netpyne_workspace

NetParams module name
gui_tut3

SimConfig module name
gui_tut3

NetParams variable
netParams

SimConfig variable
simConfig

✓ Compile mod files

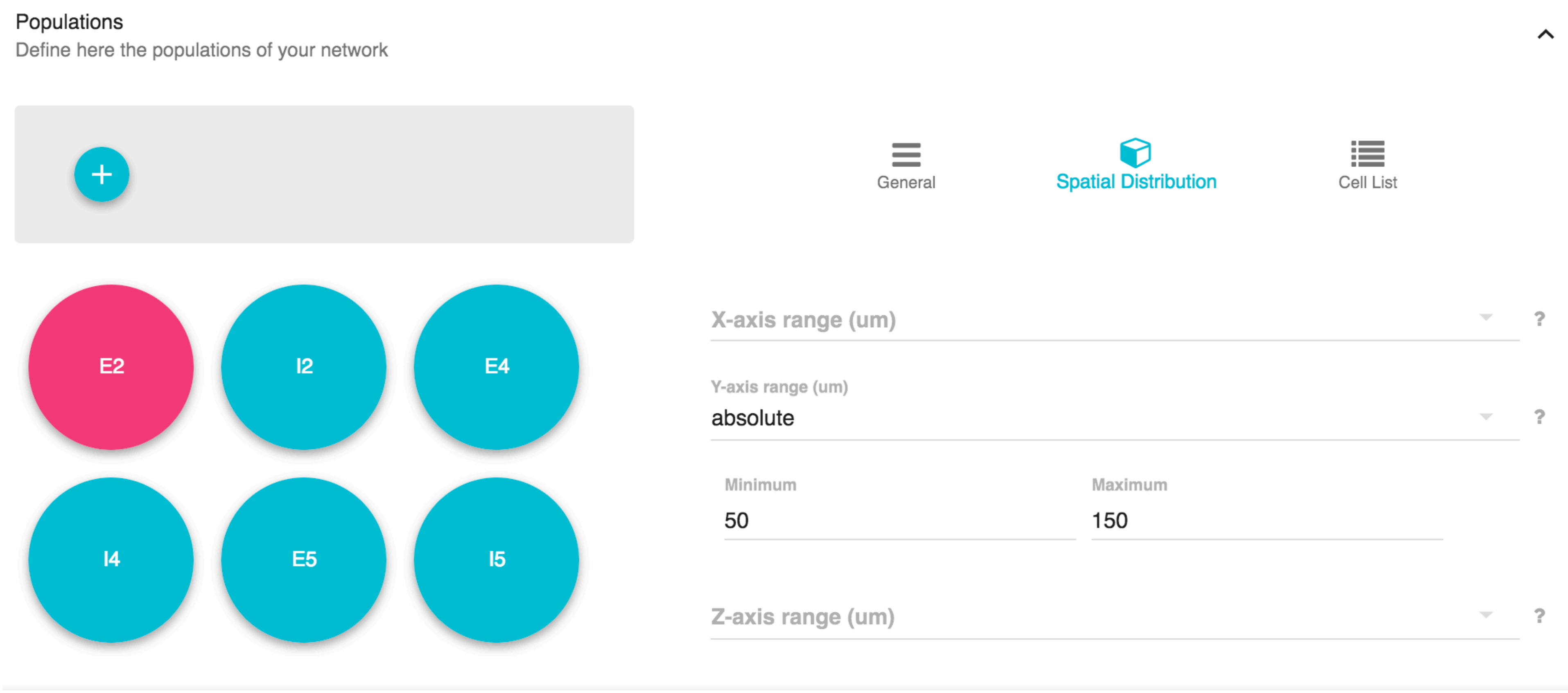
Mod path folder
/u/salvadord/NetPyNE-UI/netpyne_workspace

Select a folder. These paths are relative to:
/u/salvadord/NetPyNE-UI

+ utilities
+ pygeppetto
+ org.geppetto.frontend.jupyter
− netpyne_workspace
  + x86_64
  + mod
  + cells
  + .git
+ netpyne_ui.egg-info
+ netpyne_ui
+ netpyne
+ docs
+ .ipynb_checkpoints

NetPyNE GUI Tutorial: Multiscale net

2) Check the populations and its spatial distribution (3 layers, each with E and I pops)



Populations
Define here the populations of your network

X-axis range (um)

Y-axis range (um): absolute

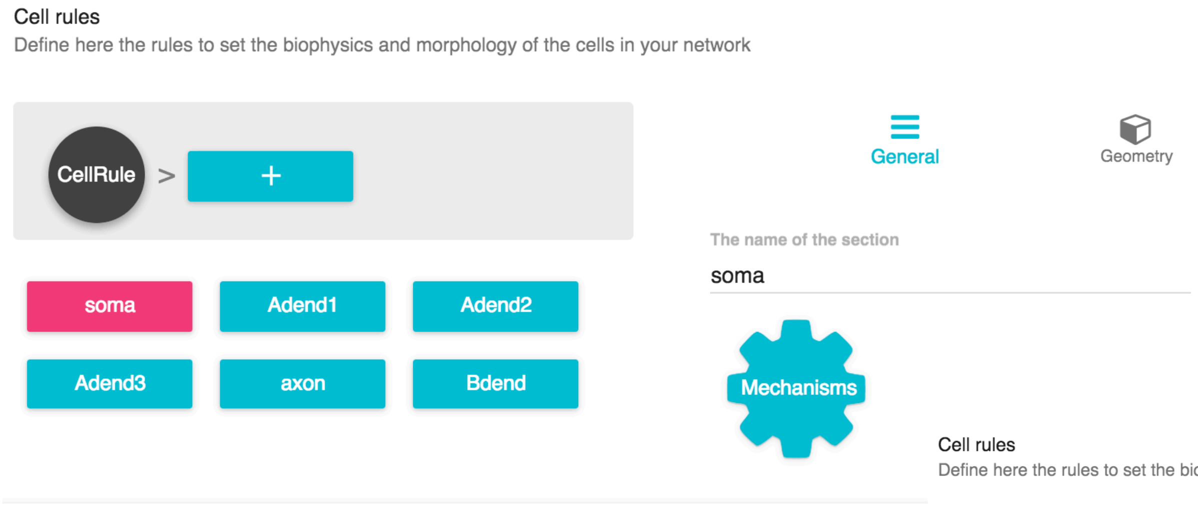
Minimum: 50

Maximum: 150

Z-axis range (um)

NetPyNE GUI Tutorial: Multiscale net

3) Check the Cell Rule, with its sections and mechanisms (6-comp cell; detailed biphysics / 9 ionic channels)

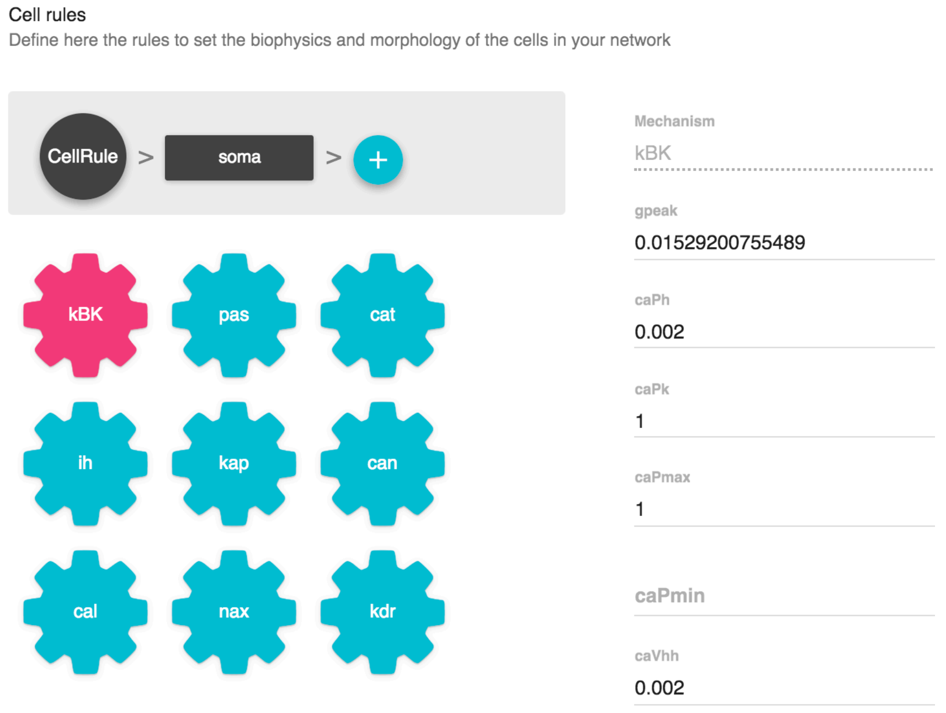


Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

The name of the section

soma



Cell rules

Define here the rules to set the biophysics and morphology of the cells in your network

Mechanism: kBK

gpeak: 0.01529200755489

caPh: 0.002

caPk: 1

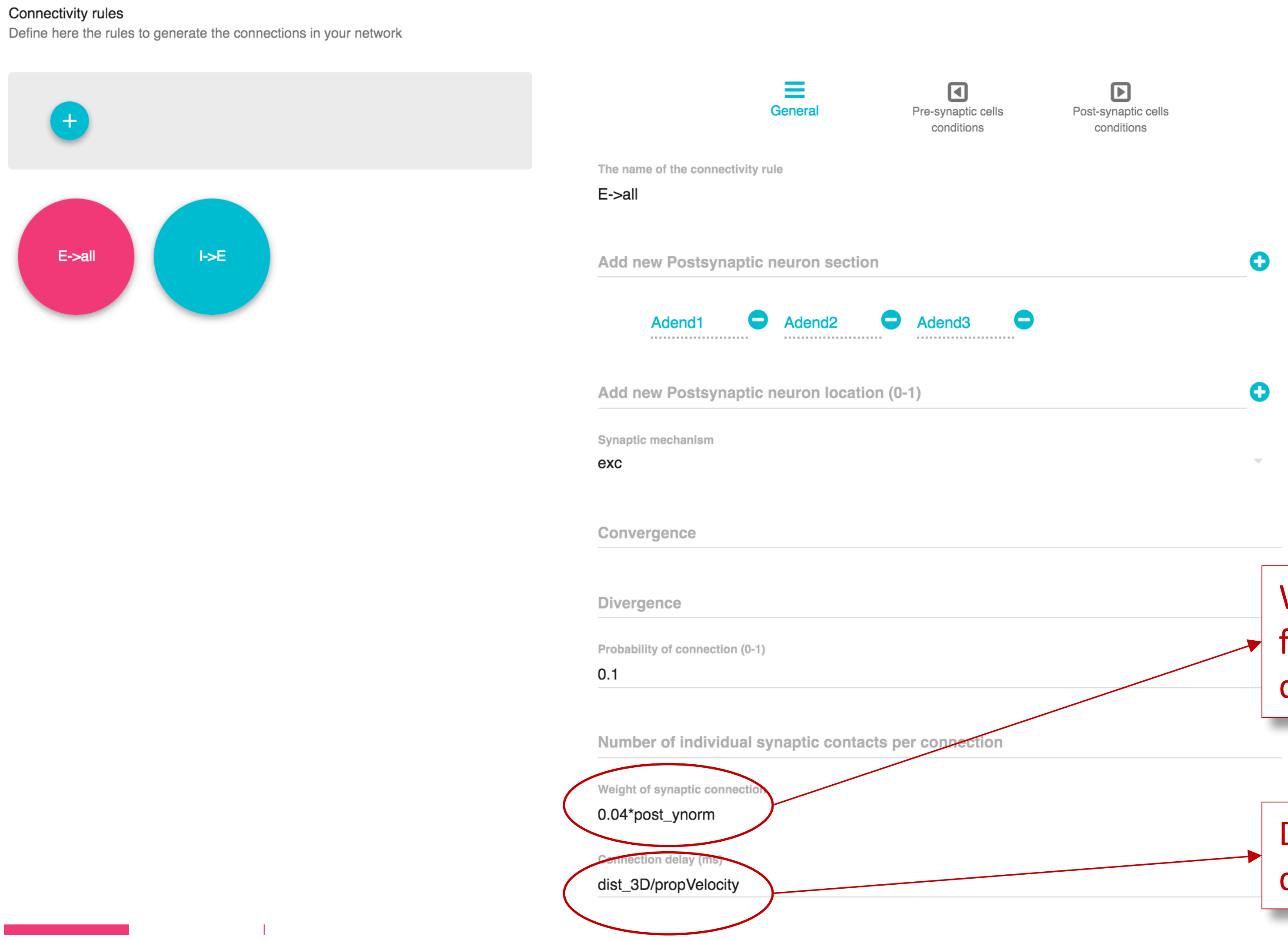
caPmax: 1

caPmin

caVhh: 0.002

NetPyNE GUI Tutorial: Multiscale net

4) Check Connectivity rules (some parameters defined using functions)



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

The name of the connectivity rule
E->all

Add new Postsynaptic neuron section
Adend1 Adend2 Adend3

Add new Postsynaptic neuron location (0-1)

Synaptic mechanism
exc

Convergence

Divergence

Probability of connection (0-1)
0.1

Number of individual synaptic contacts per connection

Weight of synaptic connection
0.04*post_ynorm

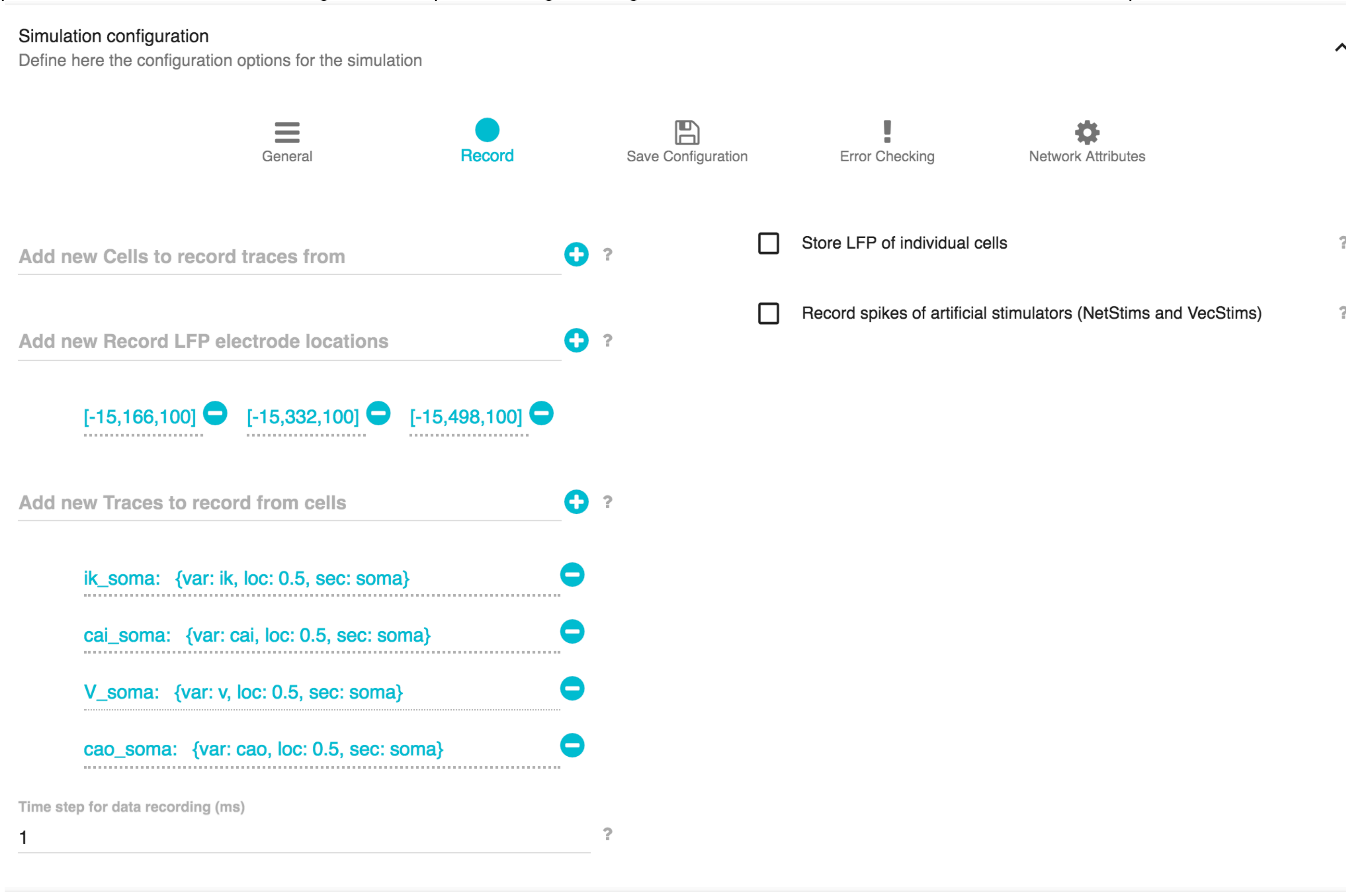
Connection delay (ms)
dist_3D/propVelocity

Weight increases as function of cortical depth

Delay is function of distance between cells

NetPyNE GUI Tutorial: Multiscale net

5) Check Simulation Configuration (recording voltage, current, and calcium concentrations)



Simulation configuration
Define here the configuration options for the simulation

General | Record | Save Configuration | Error Checking | Network Attributes

Add new Cells to record traces from ?

Add new Record LFP electrode locations ?

[-15,166,100] [-15,332,100] [-15,498,100]

Add new Traces to record from cells ?

ik_soma: {var: ik, loc: 0.5, sec: soma}

cai_soma: {var: cai, loc: 0.5, sec: soma}

V_soma: {var: v, loc: 0.5, sec: soma}

cao_soma: {var: cao, loc: 0.5, sec: soma}

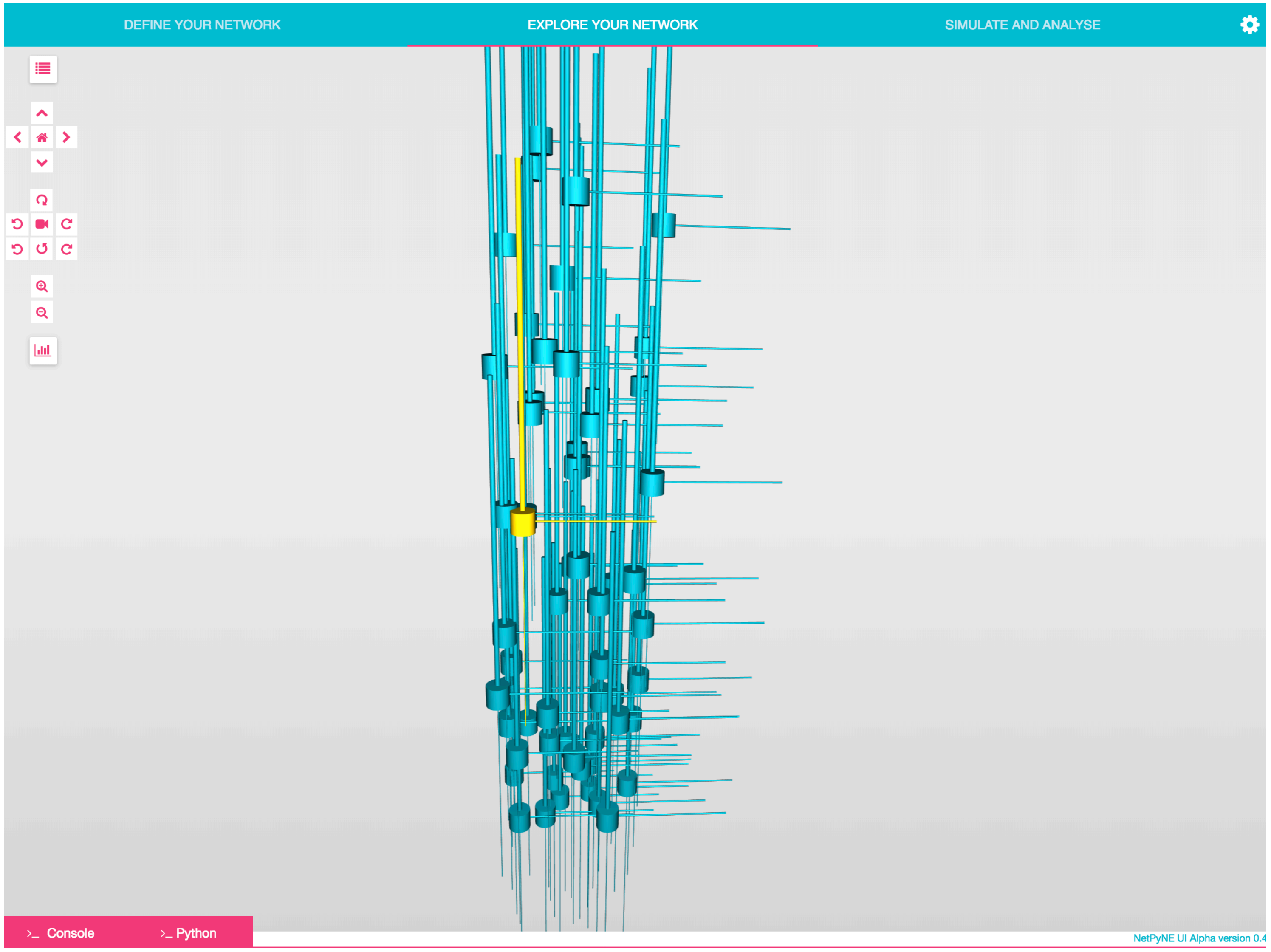
Time step for data recording (ms)
1 ?

☐ Store LFP of individual cells ?

☐ Record spikes of artificial stimulators (NetStims and VecStims) ?

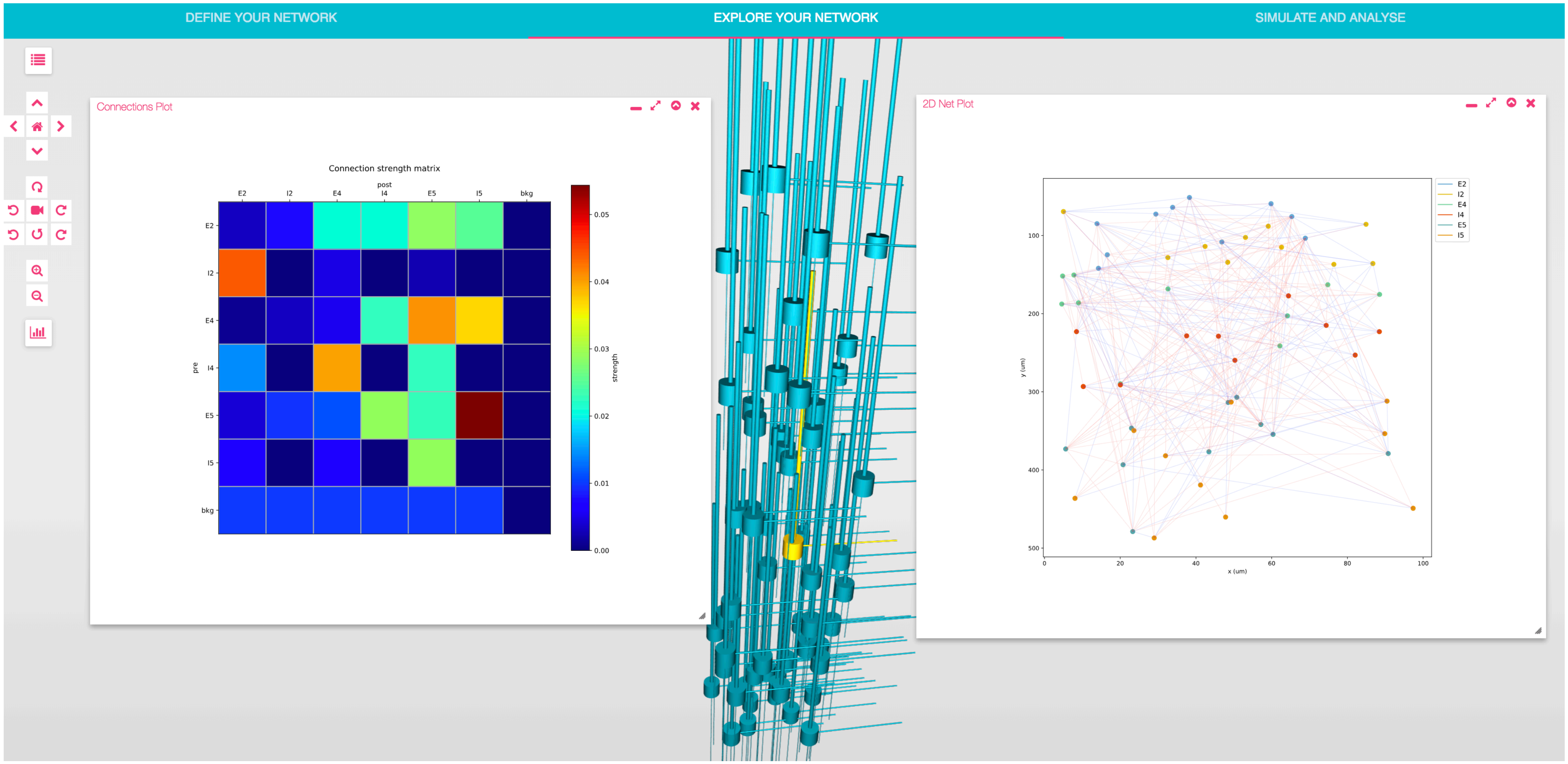
NetPyNE GUI Tutorial: Multiscale net

6) Click on “Explore your Network” to instantiate the network



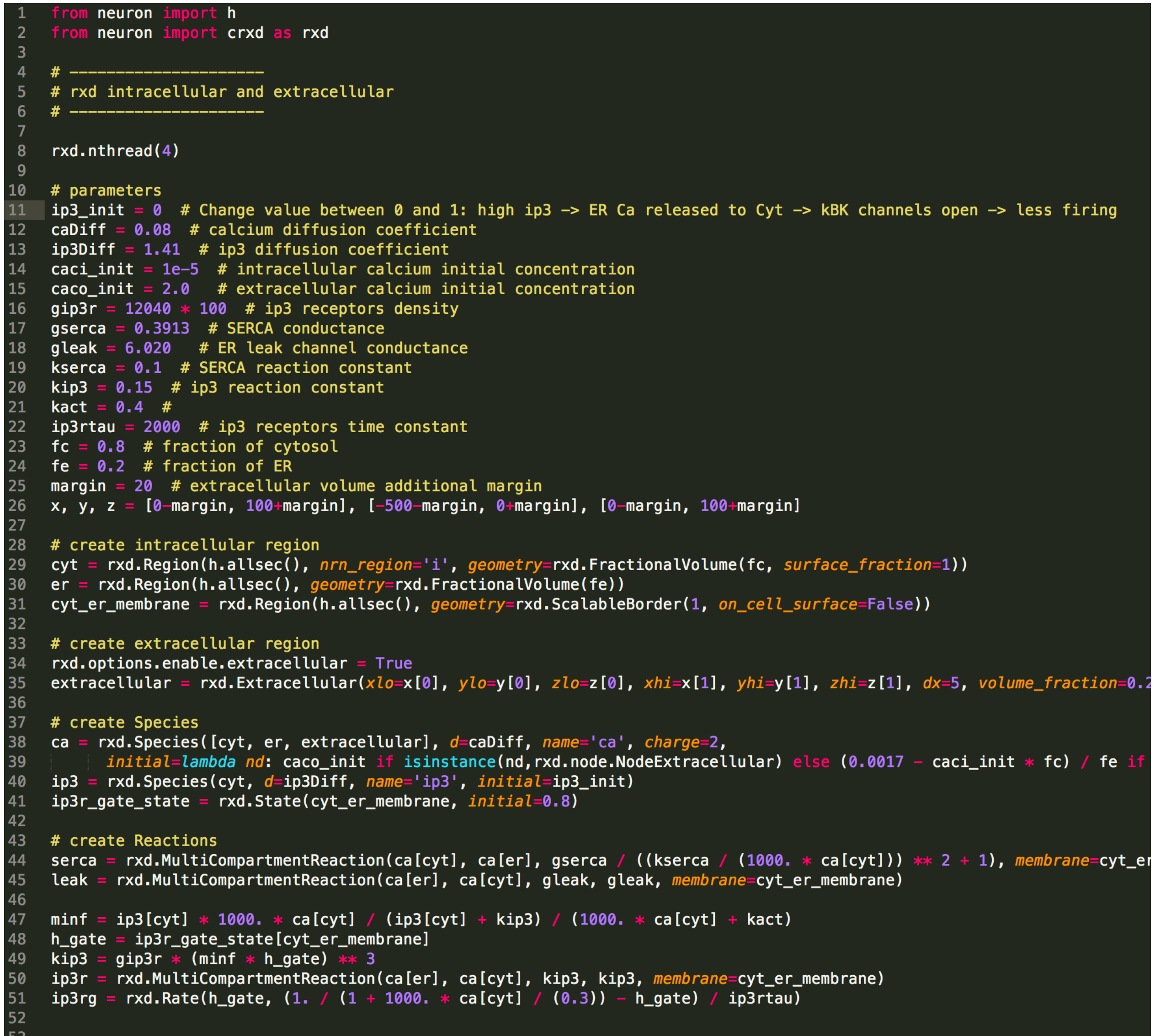
NetPyNE GUI Tutorial: Multiscale net

7) Show connectivity plots



NetPyNE GUI Tutorial: Multiscale net

8) Check reaction-diffusion (RxD) code gui_tut3_rxd.py (similar to morning tutorial)



```python
from neuron import h
from neuron import crxd as rxd

# ---------------------
# rxd intracellular and extracellular
# ---------------------

rxd.nthread(4)

# parameters
ip3_init = 0  # Change value between 0 and 1: high ip3 -> ER Ca released to Cyt -> kBK channels open -> less firing
caDiff = 0.08  # calcium diffusion coefficient
ip3Diff = 1.41  # ip3 diffusion coefficient
caci_init = 1e-5  # intracellular calcium initial concentration
caco_init = 2.0   # extracellular calcium initial concentration
gip3r = 12040 * 100  # ip3 receptors density
gserca = 0.3913  # SERCA conductance
gleak = 6.020   # ER leak channel conductance
kserca = 0.1  # SERCA reaction constant
kip3 = 0.15  # ip3 reaction constant
kact = 0.4  #
ip3rtau = 2000  # ip3 receptors time constant
fc = 0.8  # fraction of cytosol
fe = 0.2  # fraction of ER
margin = 20  # extracellular volume additional margin
x, y, z = [0-margin, 100+margin], [-500-margin, 0+margin], [0-margin, 100+margin]

# create intracellular region
cyt = rxd.Region(h.allsec(), nrn_region='i', geometry=rxd.FractionalVolume(fc, surface_fraction=1))
er = rxd.Region(h.allsec(), geometry=rxd.FractionalVolume(fe))
cyt_er_membrane = rxd.Region(h.allsec(), geometry=rxd.ScalableBorder(1, on_cell_surface=False))

# create extracellular region
rxd.options.enable.extracellular = True
extracellular = rxd.Extracellular(xlo=x[0], ylo=y[0], zlo=z[0], xhi=x[1], yhi=y[1], zhi=z[1], dx=5, volume_fraction=0.2

# create Species
ca = rxd.Species([cyt, er, extracellular], d=caDiff, name='ca', charge=2,
      initial=lambda nd: caco_init if isinstance(nd,rxd.node.NodeExtracellular) else (0.0017 - caci_init * fc) / fe if
ip3 = rxd.Species(cyt, d=ip3Diff, name='ip3', initial=ip3_init)
ip3r_gate_state = rxd.State(cyt_er_membrane, initial=0.8)

# create Reactions
serca = rxd.MultiCompartmentReaction(ca[cyt], ca[er], gserca / ((kserca / (1000. * ca[cyt])) ** 2 + 1), membrane=cyt_er
leak = rxd.MultiCompartmentReaction(ca[er], ca[cyt], gleak, gleak, membrane=cyt_er_membrane)

minf = ip3[cyt] * 1000. * ca[cyt] / (ip3[cyt] + kip3) / (1000. * ca[cyt] + kact)
h_gate = ip3r_gate_state[cyt_er_membrane]
kip3 = gip3r * (minf * h_gate) ** 3
ip3r = rxd.MultiCompartmentReaction(ca[er], ca[cyt], kip3, kip3, membrane=cyt_er_membrane)
ip3rg = rxd.Rate(h_gate, (1. / (1 + 1000. * ca[cyt] / (0.3)) - h_gate) / ip3rtau)
```

NetPyNE GUI Tutorial: Multiscale net

9) Remind yourself what's going on:

1) Metabotropic glutamate receptors (mGluR) activate

2) increase IP3 in cytosol

3) ER IP3R channels open

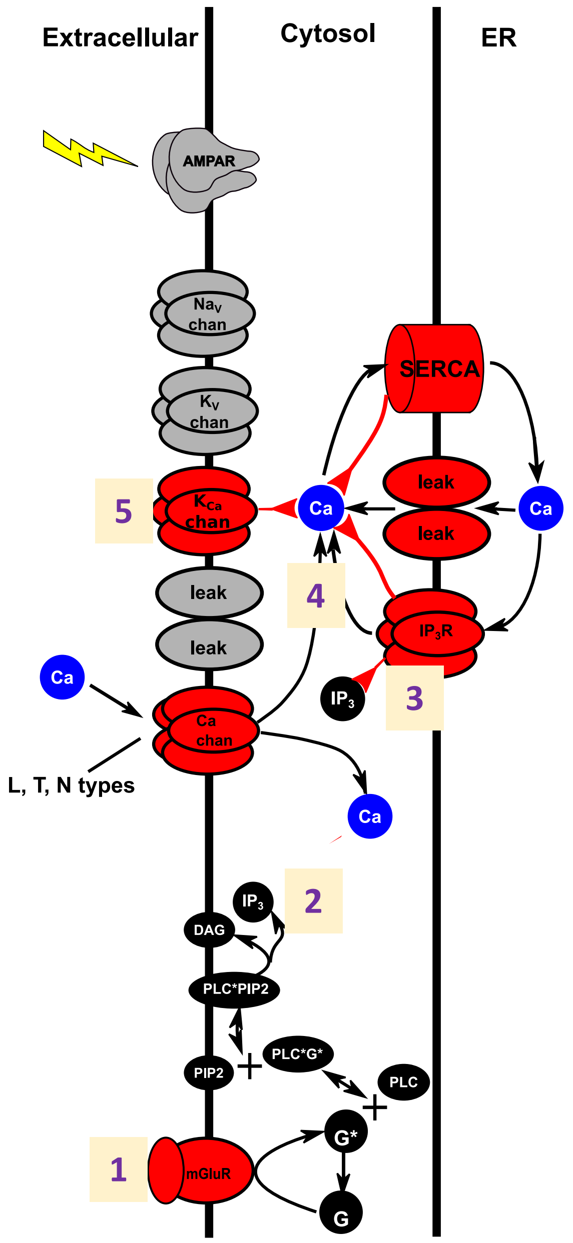
4) ER Ca released to cytosol

5) kBK / Kca channels (sensitive to Ca) open

6) K flows inside cell

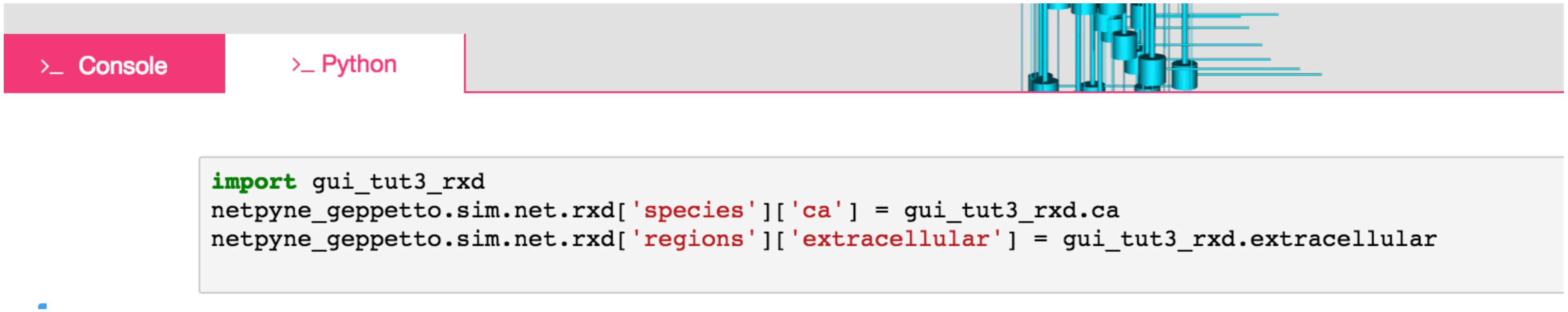
7) hyperpolarizing K current

8) reduces cell firing



NetPyNE GUI Tutorial: Multiscale net

10) Run reaction-diffusion (RxD) code via Jupyter notebook



```
import gui_tut3_rxd
netpyne_geppetto.sim.net.rxd['species']['ca'] = gui_tut3_rxd.ca
netpyne_geppetto.sim.net.rxd['regions']['extracellular'] = gui_tut3_rxd.extracellular
```

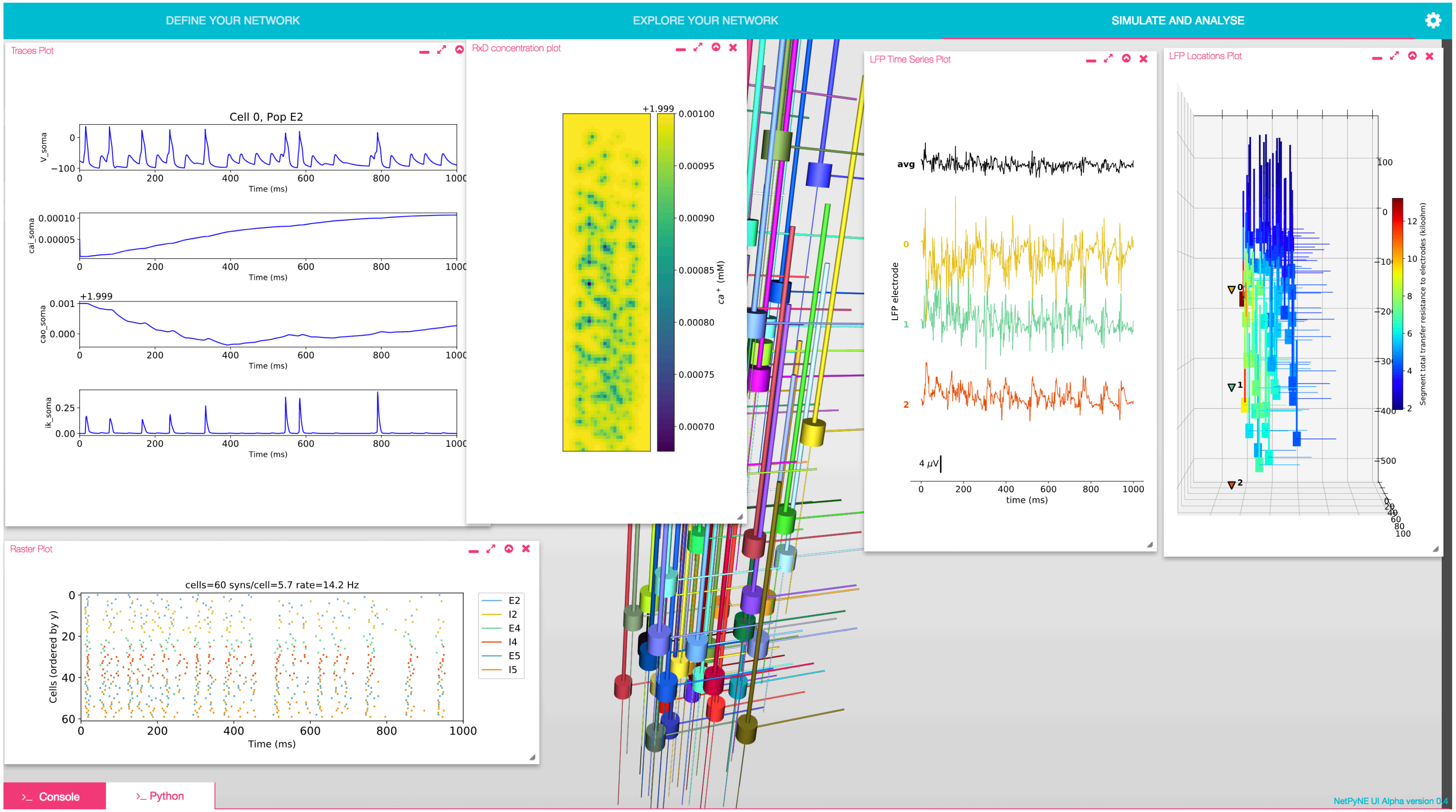
Copy paste from here:

```
import gui_tut3_rxd
netpyne_geppetto.sim.net.rxd['species']['ca'] = gui_tut3_rxd.ca
netpyne_geppetto.sim.net.rxd['regions']['extracellular'] = gui_tut3_rxd.extracellular
```

To execute press Ctrl + Enter

NetPyNE GUI Tutorial: Multiscale net

11) Run simulation and plot results [fig needs updating]

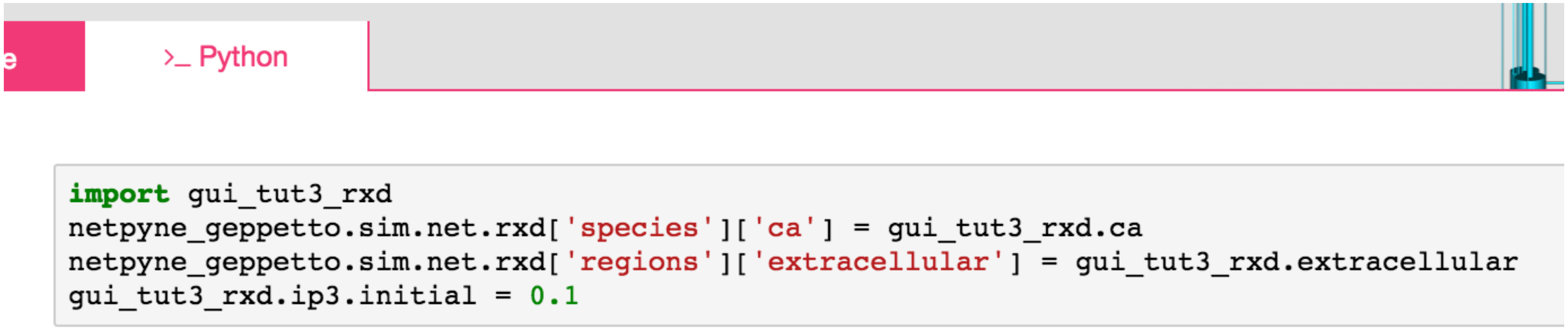


NetPyNE GUI Tutorial: Multiscale net

12) Reload web to remove current model

Repeat steps: 1) Import model and 6) Instantiate network

Now run RxD code in jupyter but set inital ip3 to 0.1 (high value):



```
import gui_tut3_rxd
netpyne_geppetto.sim.net.rxd['species']['ca'] = gui_tut3_rxd.ca
netpyne_geppetto.sim.net.rxd['regions']['extracellular'] = gui_tut3_rxd.extracellular
gui_tut3_rxd.ip3.initial = 0.1
```

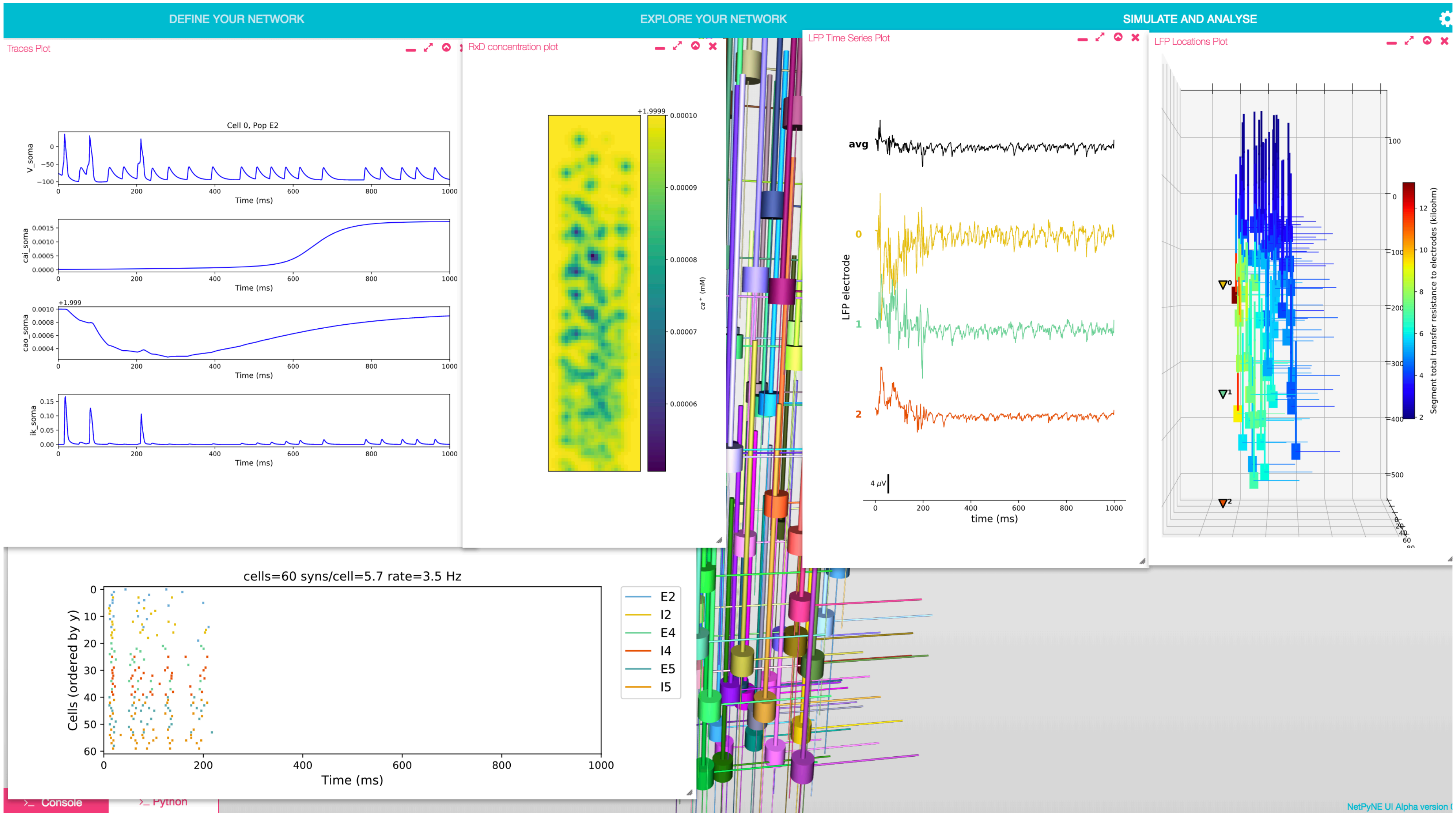
Copy paste from here:

```
import gui_tut3_rxd
netpyne_geppetto.sim.net.rxd['species']['ca'] = gui_tut3_rxd.ca
netpyne_geppetto.sim.net.rxd['regions']['extracellular'] = gui_tut3_rxd.extracellular
gui_tut3_rxd.ip3.initial = 0.1
```

To execute press Ctrl + Enter

NetPyNE GUI Tutorial: Multiscale net

13) Compare results with previous simulation



high ip3 → ER Ca released to Cyt → kBK channels open → hyperpolarizing K current → less firing