**Cortical representation of touch *in silico -* NetPyNE based simulations**

It is assumed that the network has been already instantiated with the matlab code. This includes: cells (properties and location), connections (ids of connected neurons and properties of the connections), thalamic inputs (spike trains and connections from the barreloids to cortical cells). The matlab program provides several files when setting the network. For example, these files can be generated with the function “run\_sim.m” set up to thalamic inputs (commented afterwards). In this Python-based simulation, two are needed to define the network: 1) The file “xxxxx\_ConData.mat”, 2) The file “xxxxx\_Thalamic\_Spike\_Trains.mat”.

Based on this information, this program sets the conditions via NetPyNE to perform the simulations in the NEURON simulator. In particular, the dynamics of each cell in the cortical network is based on a modified version of the Izhikevich model, with a history-dependent threshold. These dynamics are specified in a .mod file, based on a previous implementation of the Izhikevich model by Salvador Dura-Bernal. The new file, “izhi2007b\_dyn\_thr.mod”, includes the dynamics of the threshold (for certain types of cells, “celltype<5”, see COMMENTS section), and the interface with the hoc interpreter to set all properties and communications. Two points are worthy to note:

1. The Izhikevich model has a voltage level at which a spike is declared and the reset rule is applied. This voltage level is set by the parameter “vpeak”, here sets at 10 mV. Once this voltage is reached, the reset rule inside the .mod program is applied. However, the voltage update is not performed in the .mod file, but in the NEURON simulator. So, a different (lower) “monitor” threshold is set in this program (via NetPyNE, netParams.defaultThreshold = 0.0) to establish when a spike occurs, so all associated spiking events are sent to the NetCons (objects that perform the synaptic communications). This monitor threshold is set higher enough so false positives are not allowed, but lower enough so time discretization enables this to be reached and not skipped in the .mod calculation (otherwise, a finer time discretization may be necessary). Take into account that the difference between this “monitor” threshold for declaring the communication event and the threshold used to apply the reset rule results in an effective delay, that eventually can be evaluated.
2. The initial condition is set in the NET\_RECEIVE block. It could be vr (rest potential), but any other specified via a new parameter.

All populations are set according to the information in the matlab structure. Individual cellular properties are loaded with a specific “cellsList”, with all needed individual properties (locations, not very important now that the connections are already established in the matlab structure, and properties relative to the intrinsic dynamics) within the popParams specifications.

Worthy to note is that regarding the locations, in NetPyNE it is customary to set the depth as the y-coordinate, so this was taken into account when reading individual positions.

Thalamic inputs are set as a population of a specific object called “VecStim”, that also has a .mod file associated. Each cell in this population produces a spike train at specific times, which are loaded from the “xxxxx\_Thalamic\_Spike\_Trains.mat”. Cellular locations, which are irrelevant, are set in a very narrow spatial domain at the origin.

The synaptic mechanism is programmed in a specific .mod file, “FluctExp2Syn.mod”, which includes the dynamics of the gating variable, the stochastic nature of the effective transmission (failure rate), the stochastic nature of the amplitude at the postsynaptic side, and short-term learning dynamics. Also, there is commented block NET\_RECEIVE which includes a “flag\_print” to monitor how things are calculated, useful during development stages.

All connections are read from the “ConData” structure inherited from matlab. Identity of pre- and post-synaptic cells, as well as the mean amplitude and the delay, are individually set in the “connParams” dictionary in NetPyNE. Here, the individual identifier (id) of each cell is RELATIVE to the “conds” in the rule, which in this case coincide to the identifier from matlab (minus 1, since in Python everything starts from 0). Other parameters characterizing the synapses of individual connections are set with arbitrary values, except from the type (excitatory/inhibitory).

Once these connections are specified, the network is instantiated with these arbitrary values, via the command “sim.create(…)”. Then, objects corresponding to the connections (NetCon, accessed through the corresponding “hObj” in NetPyNE objects) are modified to specify individual synaptic characteristics of each connection. Worthy to note is that, at this point, for each connection the identity of each pre/post cell is via the “global” id, so it has to be disentangled the global and the relative identifier (necessary to load the properties). The code assumes two situations: The population corresponding to the thalamic input is specified and then all populations located at the cortex, or viceversa (the population corresponding to the thalamic inputs are NOT interleaved within the cortical populations).

A final remainder: Here we have three .mod files (for the dynamics of individual cells, for the synapses, and for the input spike trains). BEFORE running the NetPyNE code (and even before starting an IDE, if the program will be run from there), the .mod filed should be compiled. That is, from a terminal (or command window), setting the current directory to the location of the program, it has to be executed “nrnivmodl”.

Comment to add: Scale of the currents [mA/cm2], when the area is 100 um2.

Full scale simulation WITHOUT recurrent dynamics, only thalamic projections:

Populations:

0-599: Thalamic spike trains

600-1919: 1320 cells Pyr L4

1920-4721: 2802 cells Pyr L4

4722-5003: 282 cells Inh L4

5004-5282: 279 cells Inh L4

5283-11978: 6696 cells Pyr L2/3

11979-12296: 318 cells Inh\_FSBS L2/3

In each pop, the first and the third third correspond to secondary barrels, the middle one is principal

12297-12308: 12 cells Inh\_FSCH L2/3

12309-12473: 165 cells Inh\_BSPV L2/3

12474-12665: 192 cells Inh\_Mar L2/3

12665-12665: 0 cells Inh\_Bit L2/3

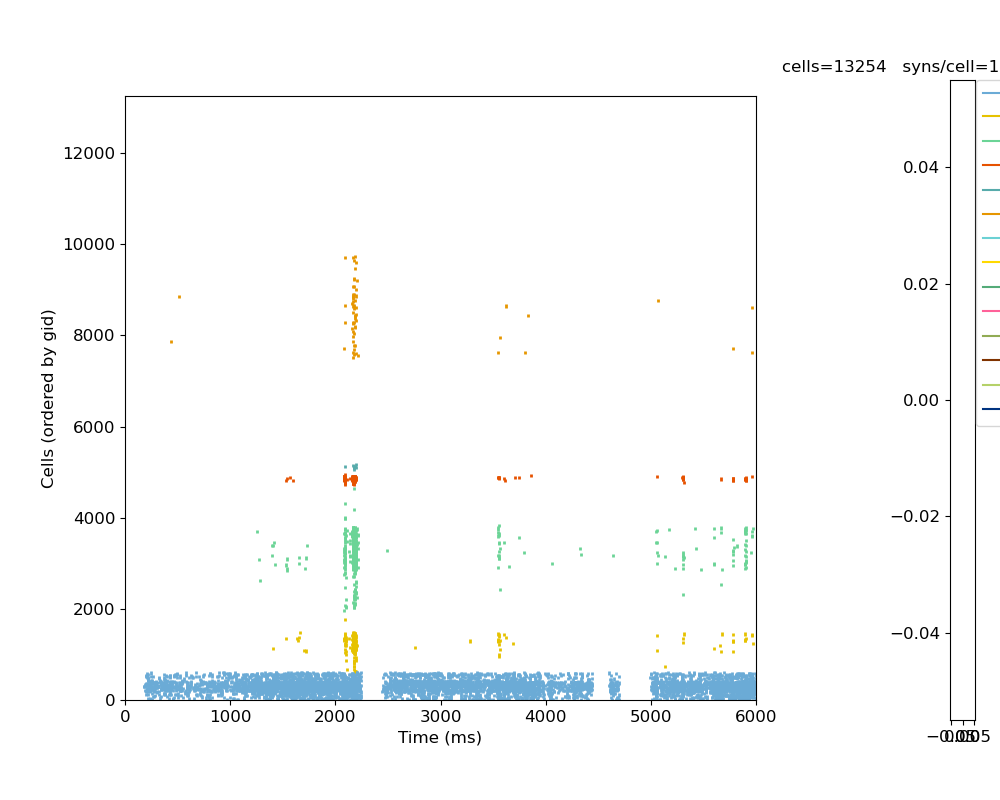
12666-12857: 192 cells Inh\_DBC L2/3

12858-12858: 0 cells Inh\_Bip L2/3

12858-12959: 102 cells Inh\_Bip L2/3

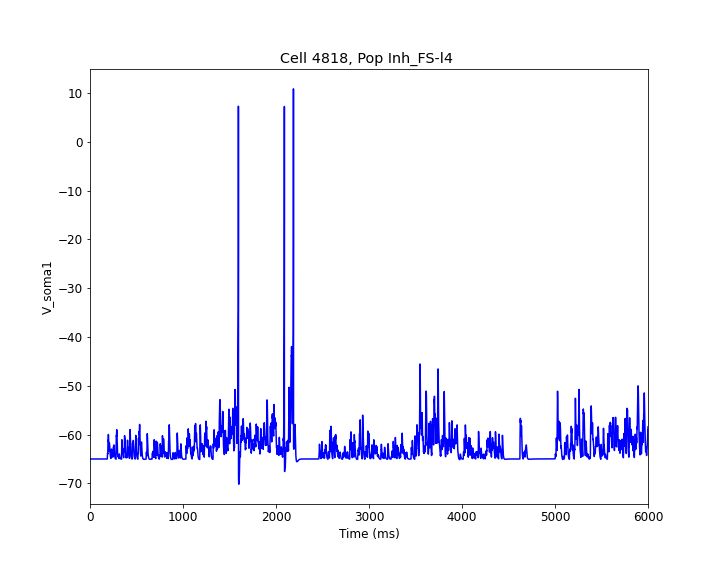
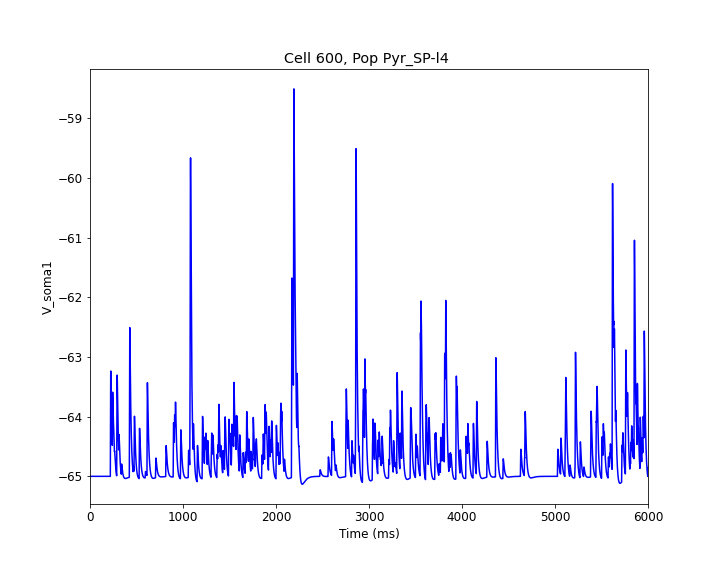
12960-13139: 180 cells Inh\_SBC L2/3

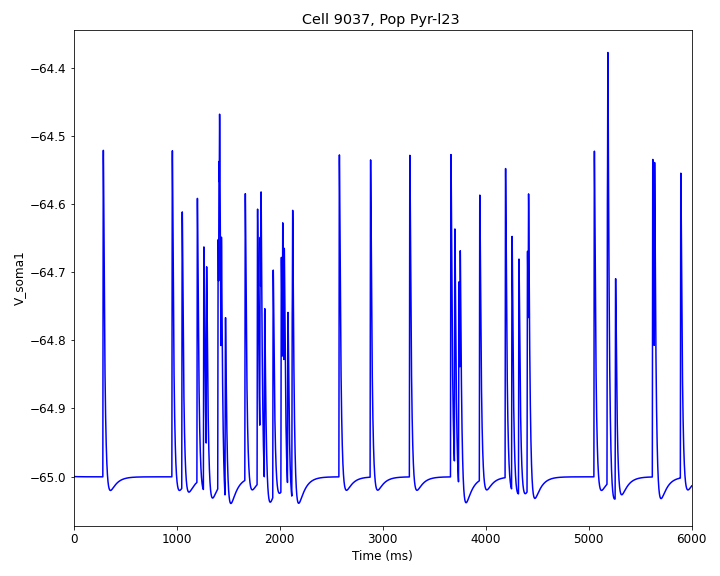
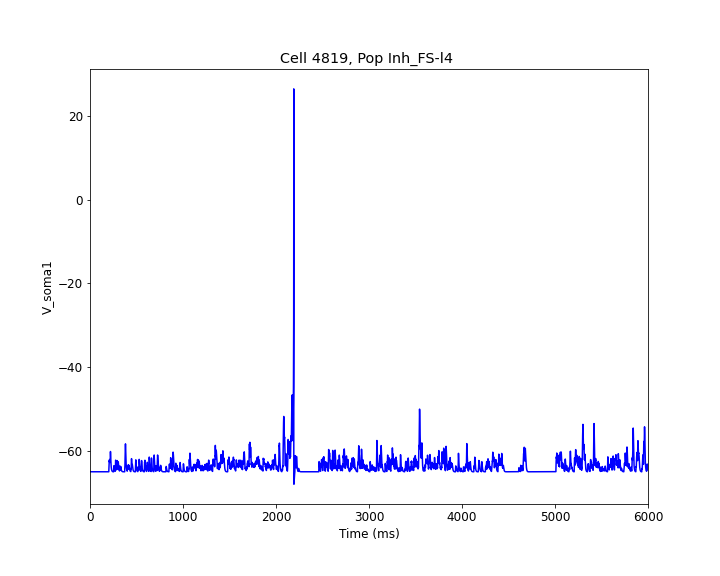
13140-12653: 114 cells Inh\_NG L2/3



Thalamic spike trains (trial 8)

Some traces I recorded:





Full scale simulation with all connections, for half of the total time (Tsim = 3 sec). Vr = -65 mV

