Exercises: Random Networks

It is recommended to work through Part 2 of *Introduction to PyNEST* before starting.

1) Specifying random networks

According to "Dale's law", neurons in the cortex either excite *all* their postsynaptic targets or inhibit them. The population of neurons in the network can therefore be subdivided into an "excitatory" and an "inhibitory" subpopulation. In this exercise, you will study the effect of Dale's law on the network dynamics. To this end, set up and simulate two different types of excitatory-inhibitory random networks, one ignoring Dale's law (network type I) and one respecting Dale's law (network type II, see below).

Hint: To lower the frustration level while developing the simulation and analysis scripts, use a smaller network size, say $N = 1250$, and a shorter simulation time, say $T = 100$ms. Remember to increase them again before you show your solutions to a tutor!

Sharing parameters

Create a script `params.py` that (sensibly!) names and specifies all the parameters that will be shared by the two network types. These parameters are:

network size $N = 12500$
neuron model: leaky integrate-and-fire with delta-shaped postsynaptic currents [`iaf_psc_delta`]
resting potential $E_\mathsf{L} = -70$mV
spike threshold $V_\mathsf{th} = -55$mV reset potential $V_\mathsf{reset} = E_\mathsf{L}$
membrane capacitance $C_\mathsf{m} = 250$pF
membrane time constant $\tau_\mathsf{m} = 10$ms
external DC input $I_\mathsf{e} = 400$pA
excitatory synaptic strength $J_\mathsf{E} = 0.2$mV
scaling factor for inhibition $g = 6$
synaptic delay $d = 0.1$ms simulation time $T = 1000$ms

Network Type I: non-Dale

Create a script `nondale.py` that imports `params.py` and sets up a network of the specified size and neuron model, using the given neuron parameters. The network should be connected as follows: for each neuron, draw $K_\mathsf{E} = \gamma K$ and $K_\mathsf{I} = (1-\gamma)K$ inputs ($\gamma = 0.8$, $K = \epsilon N$, $\epsilon = 0.1$) randomly and independently from the pool of N neurons and connect them to the target cell with weights J_E and $J_\mathsf{I} = -gJ_\mathsf{E}$, respectively and delay d. Initialise the membrane potentials V_m randomly by drawing them from a uniform distribution between E_L and V_th. Record the spiking activity (spike senders and spike times) from all neurons in the network.

Network Type II: Dale

Create a script `dale.py` that imports `params.py` and sets up a network of the specified size and neuron model, using the given neuron parameters. Subdivide the pool of N neurons into an excitatory and an inhibitory population $\mathcal{E}$ and $\mathcal{I}$, respectively (population sizes $N_\mathsf{E} = |\mathcal{E}| = \gamma N$ and $N_\mathsf{I} = |\mathcal{I}| = (1-\gamma)N$). For each postsynaptic neuron (independently of whether it belongs to $\mathcal{E}$ or $\mathcal{I}$), draw K_E inputs randomly and independently from the excitatory pool $\mathcal{E}$ and K_I inputs from the inhibitory pool $\mathcal{I}$. Randomise the initial membrane potentials and record all spikes as for the non-Dale network.

2) Analysing the data

For each network, perform the following analyses. If you wish, you can define these functions in a fourth Python file which is imported into both simulation scripts.

a) Raster plot

Plot the spiking activity in a raster plot (neuron indices vs. spike times)

b) Average firing rate

Compute the average firing rate as

$$r = \frac{\text{number of spikes in time interval } [0, T)}{NT}. \tag{1}$$

c) Spike count histogram

Compute and plot the spike count histogram $n(t_i; h)$ of the superposition of all spike trains in bins of width $h = 10\,\mathrm{ms}$. **Hint:** construct a `numpy.array` that contains the edges of the bins, e.g. $[0, h, 2h, \ldots T - h, T]$ and then use `matplotlib.pyplot.hist` to construct and plot the histogram of all the recorded spike times according to those bins.

d) Fano Factor

As a measure of the magnitude of the population-activity fluctuations, compute the *Fano factor*

$$\mathsf{F} = \frac{\mathrm{Var}_i[n(t_i; h)]}{\mathrm{Mean}_i[n(t_i; h)]} \tag{2}$$

of the compound spike-count signal calculated in **c**

e) Writing to file

Now your analyses are all working, investigate how to get the spike detector to write its data to file, and read the data in from file before running the analyses. Overwriting of files is not configured on individual detectors, but for all detectors on the level of the kernel:

```
nest.SetKernelStatus({"overwrite_files": True})
```

f) Comparison of results

Compare the raster plots, average firing rate, spike count histogram and Fano factor for the Dale and non-Dale network. Which measures are different? Can you explain why?