FYS388_V15_Exercise07

March 18, 2015

1 FYS388 Exercise 7: Single neuron models in NEST

1.1 Introduction

1.1.1 Purpose

Through this set of exercises, you will gain basic user experience with NEST and with visualizing results from NEST simulations with Python. You will also gain some experience with the response properties of simple spiking neuron models.

1.1.2 Recommendations

- You should use the Python interface to NEST for the exercise.
- You can do this exercise using Python from the command line or your favourite Python IDE. In that case, you should write up your report on this exercise in your favourite document generating system.
- If you are comfortable with IPython Notebooks, your are encouraged to do all your work on the exercise in this notebook and hand in the exercise as a completed notebook.
- Note that some NEST output, e.g., output from `nest.PrintNetwork()`, will not show up in the IPython Notebook but in the terminal window in which you started `ipython notebook`. [For geeks: this is because the NEST C++-kernel writes directly to `stdout` instead of passing output to Python.]
- You may need to add the path to NEST to your Python path. You can to this inside Python with the following lines (the given path works for the Live CD, adapt as necessary):

```
import sys
sys.path.append('/home/nest/Source/nest-2.6.0_install/lib/python2.7/site-packages/')
```

1.1.3 Leaky integrate-and-fire models in NEST

The membrane potential of the leaky integrate-and-fire neuron models we will use is given by (see also Exercise 2 from the first part of the course)

$$\frac{dV}{dt} = -\frac{V - E_L}{\tau_m} + \frac{1}{C}\left(I_e + I_{\text{syn}}\right) \tag{1}$$

where I_e is a constant input current built into the neuron, while I_{syn} is the total synaptic input current. In the absence of synaptic input ($I_{\text{syn}} = 0$), the differential equation has the solution

$$V(t) = V_0 e^{-t/\tau_m} + \left(E_L + \frac{I_e \tau_m}{C}\right)\left(1 - e^{-t/\tau_m}\right) \tag{2}$$

where $V_0 = V(t = 0)$. The asymptotic value of the membrane potential is thus

$$\lim_{t\to\infty} V(t) = E_L + \frac{I_e \tau_m}{C} \ . \tag{3}$$

A note on units. In NEST, we use the following units, which are mutually compatible:

- times: ms
- voltages: mV
- currents: pA
- capacitances: pF
- conductances: nS

The neuron fires a spike whenever V reaches the threshold potential V_{th}, when membrane potential is reset to E_L and the neuron becomes refractory for a given period (default 2 ms). During the refractory time, $V(t) = E_L$, while $I_{\text{syn}}(t)$ evolves unaffected.

We will consider three variants of the integrate-and-fire neuron, which differ in the type of input current supplied by the synapses: `iaf_psc_delta`, `iaf_psc_exp`, and `iaf_psc_alpha`, which have the following synaptic input currents:

Name	Synaptic current
delta	$I_{\text{syn}}(t) = \sum_k w_k \delta(t - t_k)$
exp	$I_{\text{syn}}(t) = \sum_k w_k e^{-(t-t_k)/\tau_{\text{syn}}} \Theta(t - t_k)$
alpha	$I_{\text{syn}}(t) = \sum_k w_k \frac{e(t-t_k)}{\tau_{\text{syn}}} e^{-(t-t_k)/\tau_{\text{syn}}} \Theta(t - t_k)$

Here, t_k are the arrival times of presynaptic spikes, while the w_k are the pertaining synaptic weights, and τ_{syn} the synaptic time constant. In practice, the models have two different time constants, for excitatory ($w \geq 0$) and inhibitory ($w < 0$) synapses, respectively. $\Theta(t)$ is the Heaviside step function.

Spikes arriving through delta synapses let the membrane potential jump instantaneously by w, while the other two synapse types inject a synaptic current decaying exponentially from w or rising from 0 at $t = t_k$ to w at $t = t_k + \tau_{\text{syn}}$, before decaying exponentially.

1.2 Task 1: Neuron driven by intrinsic current

In this task, you will study a neuron driven by an intrinsic current and adjust the current to achieve different firing rates.

1.2.1 Network configuration

Create a network model with the following components:

1. An `iaf_psc_delta` model neuron
2. A `voltmeter` recording from the neuron
3. A `spike_detector` recording spikes from the neuron

Set the external current I_e injected into the neuron to a value you choose. It is the `I_e` property of the neuron with units pA.

In this and all following tasks, set the `interval` property of the `voltmeter` to 0.1 ms.

To make experimentation easy, I recommend that you define a Python function `build_network()` that calls `nest.ResetKernel()` first to delete any existing network, and then builds the network anew, including setting I_e.

1.2.2 Simulation and Visualization

For a given value of I_e, simulate the network. Then extract all spike times from the spike detector and membrane voltages (including the times for the respective voltage values) from the voltmeter and plot them in a single graph: The voltage trace as a line, and the spikes as dots or lines above the voltage trace.

1.2.3 Exploration

You can perform the following explorations starting from the equations given in the introduction, by manual or by automatized search. You should document in your report how you have proceeded.

1. Find the smallest and the largest values of I_e that evoke exactly one spike during 50 milliseconds simulated time.
2. Find a value of I_e that evokes exactly 20 spikes during 1 second simulated time.
3. Find the value of I_e that yields spikes with an interspike interval of precisely 20 ms.

1.3 Task 2: A neuron exposed to noise

In this task, you will study the same network as in Task 1, but with noisy background current added.

1.3.1 Network configuration

Use the same configuration as in Task 1, but add

4. A `noise_generator` providing noisy current input to the neuron

Set the mean μ and standard deviation σ of the noise current to values you choose (parameters `mean` and `std`, both in pA).

The noise generator uses random numbers to create a random signal. We therefore need to seed the random number generator built into NEST. When simulating with just a single process (we assume this throughout this exercise), we need to seed NEST's global random number generator and the generator for the one simulation process as follows:

```
nest.SetKernelStatus({'grng_seed': my_seed, 'rng_seeds': [my_seed + 1]})
```

Write an extended version of your network-building function that sets I_e, μ, σ and seeds the random number generators.

1.3.2 Simulation and Visualization

Proceed as in Task 1, but extend the simulation time as suggested below. All plots of voltage traces should be limited to 1 s, while histograms of voltage data and spike times should use the full set of data provided (use `numpy.hist()` to create histograms).

1.3.3 Validation

Cross-validate your network for Task 2 against the network you created for Task 1 as follows: Set I_e to 0, and μ to the value of I_e that yielded ISIs of 20 ms in Task 1. Simulate for 200 ms and compare results with the network from Task 1. Why are the curves shifted relative to each other? If you correct for the shift, do the results agree?

1.3.4 Exploration

1. Using the I_e that resulted in precisely 20 spikes in Task 1, add noise with zero mean ($\mu = 0$) and perform simulations with at least three different values of σ for 1 s biological time. Plot voltage traces and spike trains. Ideally, you should combine all voltage traces in a single graph.

2. Using the same I_e again, simulate for 100 s biological time for three different σ and plot histograms of the interspike intervals and of the membrane potential distribution.

3. Again using the same I_e, simulate for 100 s biological time for at least 20 different values of σ (choose the range based on the first steps), and for each value of σ, determine the firing rate r of the neuron. Plot the firing rate r as a function of σ.

4. For fixed values of I_e, μ and σ, chosen such that the firing rate is at least 25 spikes per second, simulate the network 30 times for 1 s biological times, using a different seed for each simulation. Plot the resulting spike times as raster plot:

    - Horizontal axis is time
    - Vertical axis is one row per simulation
    - Spike times are marked as dots

1.3.5 Challenge

The noise-rate curve in item 3 above is the result of a single stochastic simulation, i.e., the result of an *experiment*. If we repeated the simulation with a different seed, we would get a different curve. How can you obtain an estimate on the variation across repeated experiments, so that you can add error bars to the $\sigma - r$ curve?

1.4 Task 3: Neurons with Spike Input

In this task, you will explore how neurons respond to synaptic input with different time courses.

1.4.1 Network configuration

Create a network with the following components:

1. One neuron each of types `iaf_psc_delta`, `iaf_psc_exp` and `iaf_psc_alpha`
2. One `spike_generator` connected to all three neurons
3. One voltmeter recording from each of the neurons

Again, you should write a network-building function that sets the times of the spikes t_s to be emitted by the `spike_generator` (property `spike_times`), and the weights for the three synapses between spike generator and neurons (different weight for each synapse). Connection delays should be fixed at 1 ms (default value).

1.4.2 Simulation and visualization

Simulate for 100 ms biological time for each of the stimulation protocols suggested below. Then, extract the membrane potential trace for each neuron and plot all three traces in one figure.

1.4.3 Exploration

1. Simulate with input spikes $t_s = 10, 20, 27, 30, 40, 44, 50, 58, 65, 70$ and equal weights $w = 1$ pA.
2. Simulate with the same t_s, but adjust the weights between the spike generator and the neurons so that each neuron fires 10 output spikes (one per input spike).

1.4.4 Challenge

Given the set of t_s above, can you adjust the input weights and delays to the different neurons such that the output spike trains from all three neurons are identical? If not, explain why you (must) fail!

1.5 Task 4: Excitatory and Inihibitory Poisson Input (Challenge Task)

This is an additional challenge task in which you will explore the effect of Poisson input spike trains on neurons via different synapse types.

1.5.1 Network configuration

Create a network with the following components:

1. Two `poisson_generator`s, one for excitatory and one for inhibitory input
2. Two `parrot_neuron`s, receiving input from one Poisson generator each
3. One neuron each of types `iaf_psc_delta`, `iaf_psc_exp` and `iaf_psc_alpha`, each receiving input from both parrot neurons
4. One `voltmeter` and one `spike_detector` per neuron

Signals thus flow from the Poisson generators via the parrot neurons to the leaky integrate-and-fire neurons and then to the recording devices. The purpose of the parrot neurons is to make sure that all neurons receive the same Poisson spike trains, so that we can check the effect of the different synapse types without any confusion due to different input spike trains.

Of the two parrot neurons, one should be connected with a positive weight (excitatory input) and one with a negative weight (inhibitory input).

Again, write a network-building function. As we use random spike trains, remember to add seeding of the random number generators.

1.5.2 Simulation and visualization

In general, simulate for 10 s biological time, but limit visualization of voltage traces to 200 ms for clarity.

1.5.3 Exploration

1. Set the `rate` of the excitatory generator to 1000 spikes/second and that of the inhibitory generator to 0. For each of the three neurons, find the excitatory weight required to obtain an output firing rate of approximately 100 spikes/second.
2. With the excitatory rate and weight as above, set the rate of the inhibitory generator also to 1000 spikes/second. Now find inhibitory weights for all three neuron models that reduce the output firing rate to approximately 30 spikes/second.
3. Plot membrane potential traces with the weights determined above (200 ms) and histograms of the membrane potential distributions of the neurons.
4. With weights fixed to the values determined in the previous step, vary the input rates for both excitatory and inhibitory generator from 100 to 100.000 spikes/second (use logarithmic stepping). Measure and plot the output firing rates of the three neurons as functions of the input rate.