universität freiburg

Introduction to NEST - the NEural Simulation Tool

nest :: simulated()

Wenqing Wei & Stefan Rotter
Bernstein Center Freiburg & Faculty of Biology

31. July, 2023

Overview

- What is NEST?
- Why NEST?
- How to use NEST?
- Examples

What is NEST?

- the NEural Simulation Tool
- able to create, connect and simulate large neural networks
- representing and implementing biological realistic parameters
- Investigate the dynamics and functions of networks and single neuron models

See NEST simulator documentation at https://nest-simulator.readthedocs.io/

Why NEST?

- A large number of published neuron models are implemented in NEST.
- NEST provides over 10 synapse models, including static and plastic models.
- NEST offers a lot of helpful examples, ranging from single neuron simulation to large neural networks.
- NEST is fast and effecient.
- Unicode from local desktop to supercomputers.
- NEST has a very active community.
- NEST is open source and free!
- …

How to use NEST?

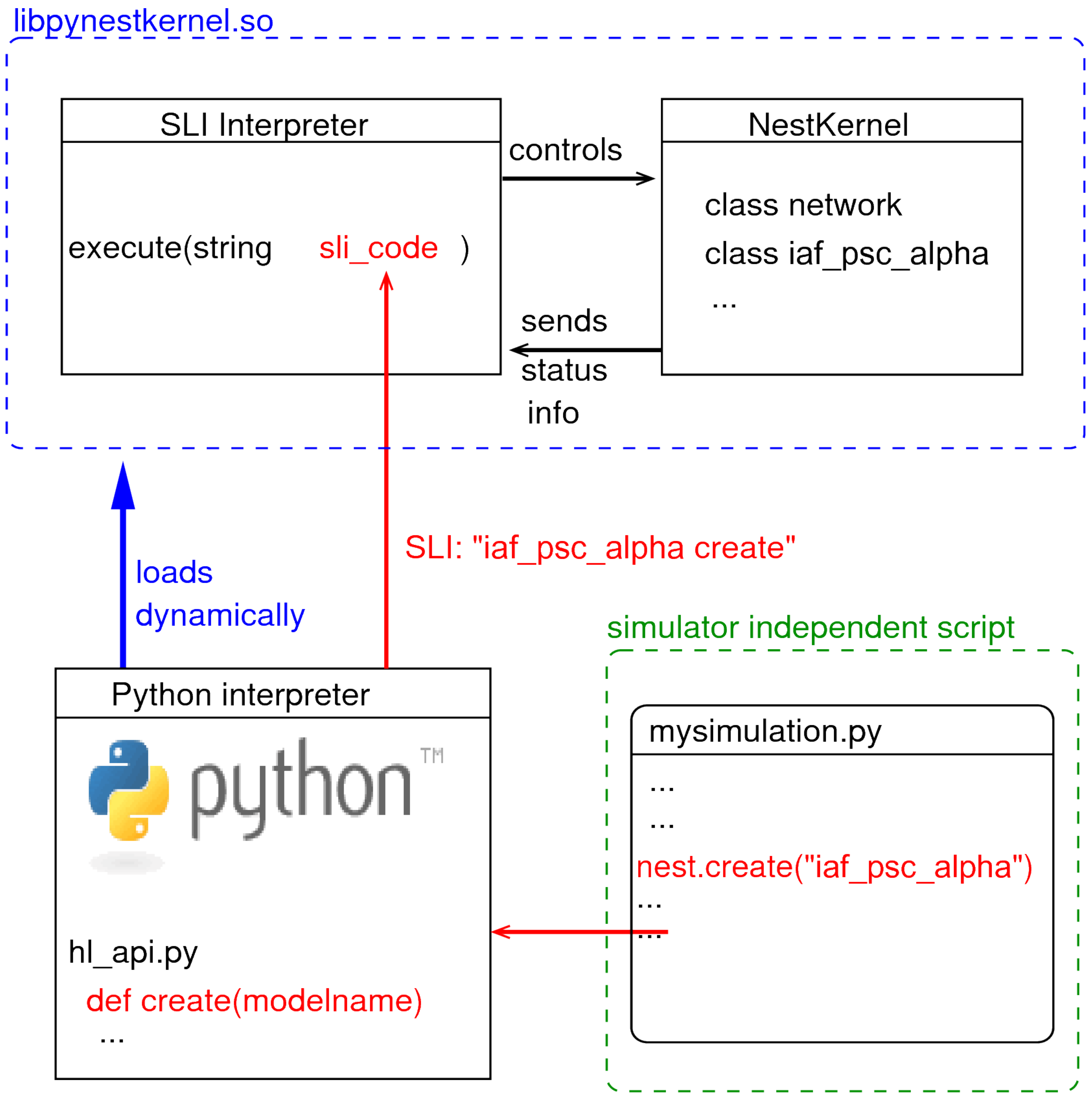
- **PyNEST - Python interface**

```
import nest
stimulus = nest.Create('poisson_generator', 1,
                       {'rate': 6500.})
neuron = nest.Create("iaf_psc_delta", 1)
sd = nest.Create("spike_recorder", 1)
nest.Connect(stimulus, neuron)
nest.Connect(neuron, sd)
nest.Simulate(1000.)
```

- **Complement PyNEST with PyNN**

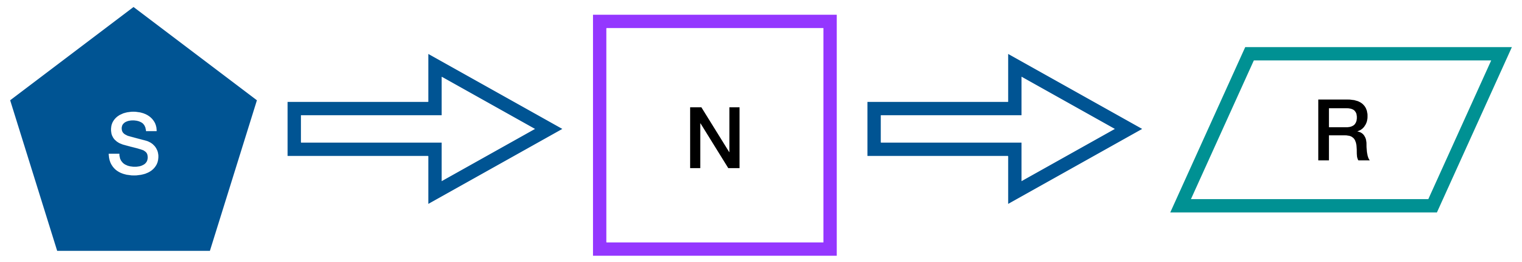
```
from pyNN import nest
```

- **A stand alone application (nest)**



PyNEST - Python interface to the NEST simulator
(Grab from [NEST documentation])

Available models - Stimulation devices



Current generator

- ac_generator - produce an alternating current (AC) (sine-shaped)
- dc_generator - produce a direct current (DC) input (constant)
- step_current_generator - provide a piecewise constant DC input current
- noise_generator - generate a Gaussian white noise current
- …

Spike generator

- spike_generator - generate spikes from given array with predefined spike times
- poisson_generator - generate spikes with a Poisson distribution
- sinusoidal_poisson_generator - produce sinusoidally modulated Poisson spike trains
- inhomogeneous_poisson_generator - generate Poisson spike trains with a piecewise constant rate
- …

Available models - Neuron models

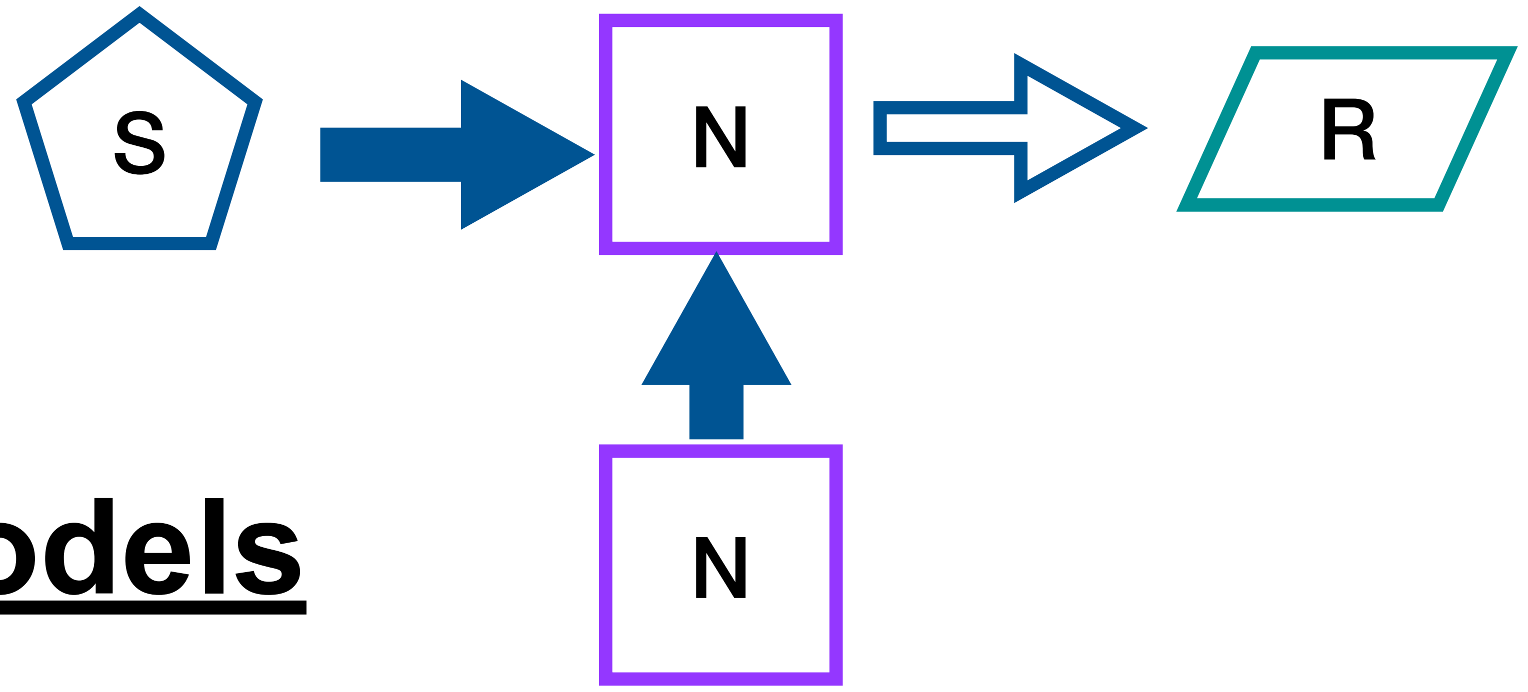
Current-based

- iaf_psc_delta - Leaky integrate-and-fire (LIF) neuron model with delta-shaped PSC
- iaf_psc_alpha - LIF neuron model with alpha-function shaped PSC
- hh_psc_alpha - Hodgkin-Huxley neuron model with alpha-function shaped PSC
- …

Conductance-based

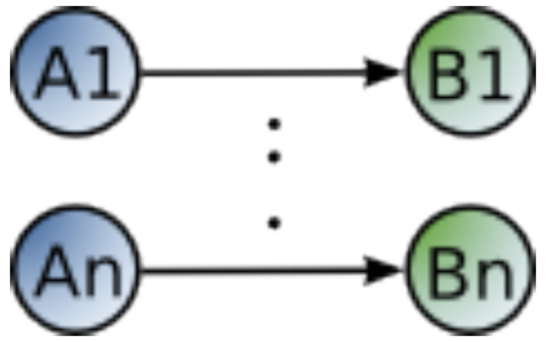
- iaf_cond_alpha - LIF neuron model with conductance-based synapses of alpha function
- iaf_cond_exp - LIF neuron model with conductance-based synapses of exponential function
- hh_cond_exp_traub - an implementation of a modified Hudgkin-Huxley model (Brette et al, 2007)
- …

Available models - Connections

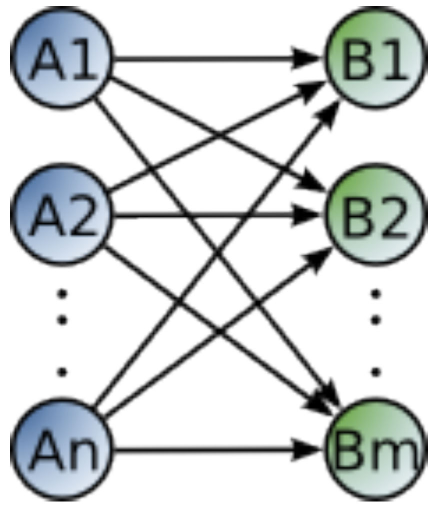


Connection rules

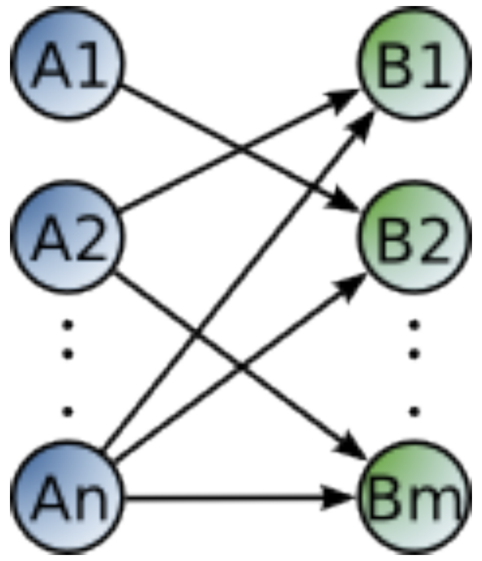
- one_to_one



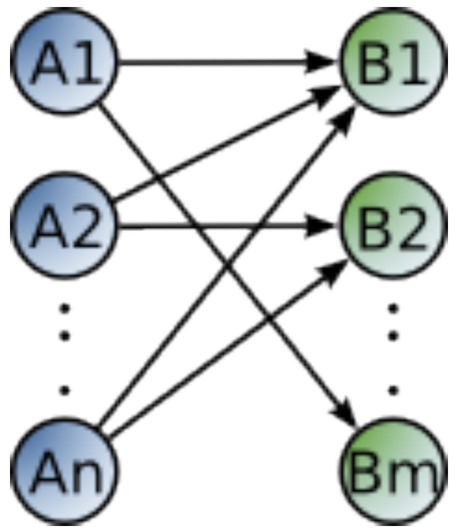
- all_to_all



- fixed_indegree



- fixed_outdegree

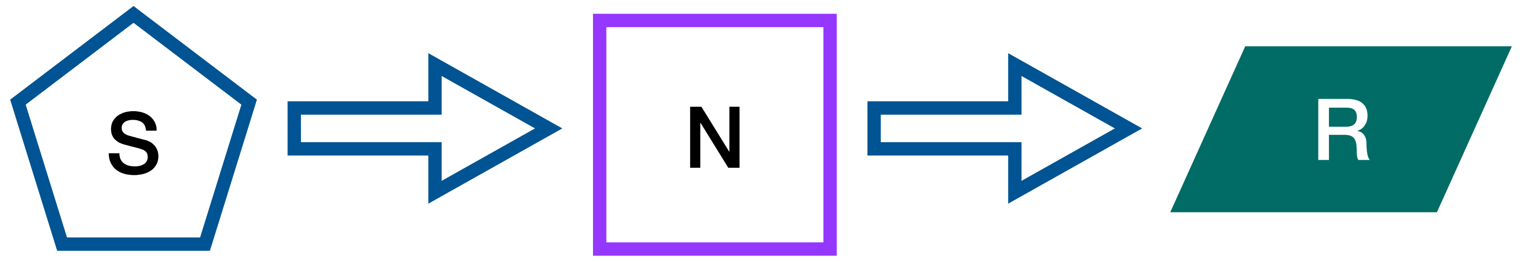


- fixed_total_number, pairwise_bernoulli, …

Synapse models

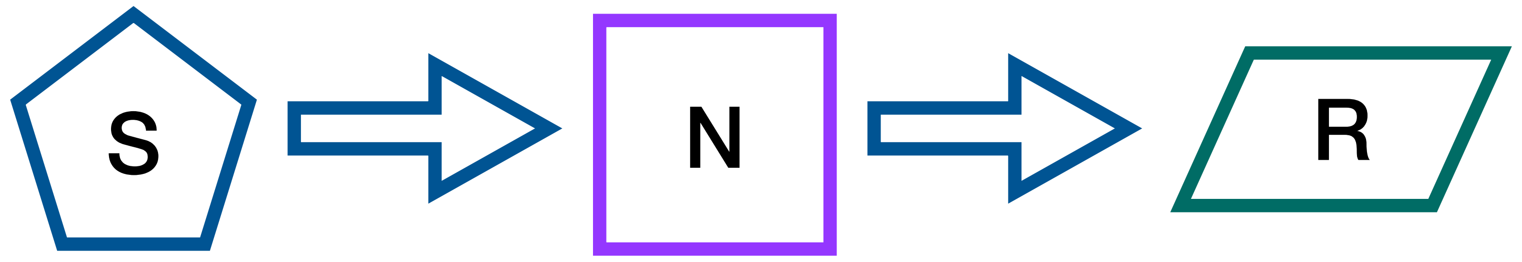
- static_synapse
- Spike-Timing-Dependent Plasticity (STDP)
  - stdp_synapse
  - stdp_dopamin_synapse
  - …
- Short-Term Plasticity
  - tsodyks2_synapse
  - …
- …

Available models - Recording devices



- spike_recorder - record all spikes from the connected neuron(s)
- voltmeter - record the membrane potentials from the connected neuron(s)
- multimeter - record analog quantities from the connected neuron(s)
- weight_recorder - record weights from synapses
- …

A simple example



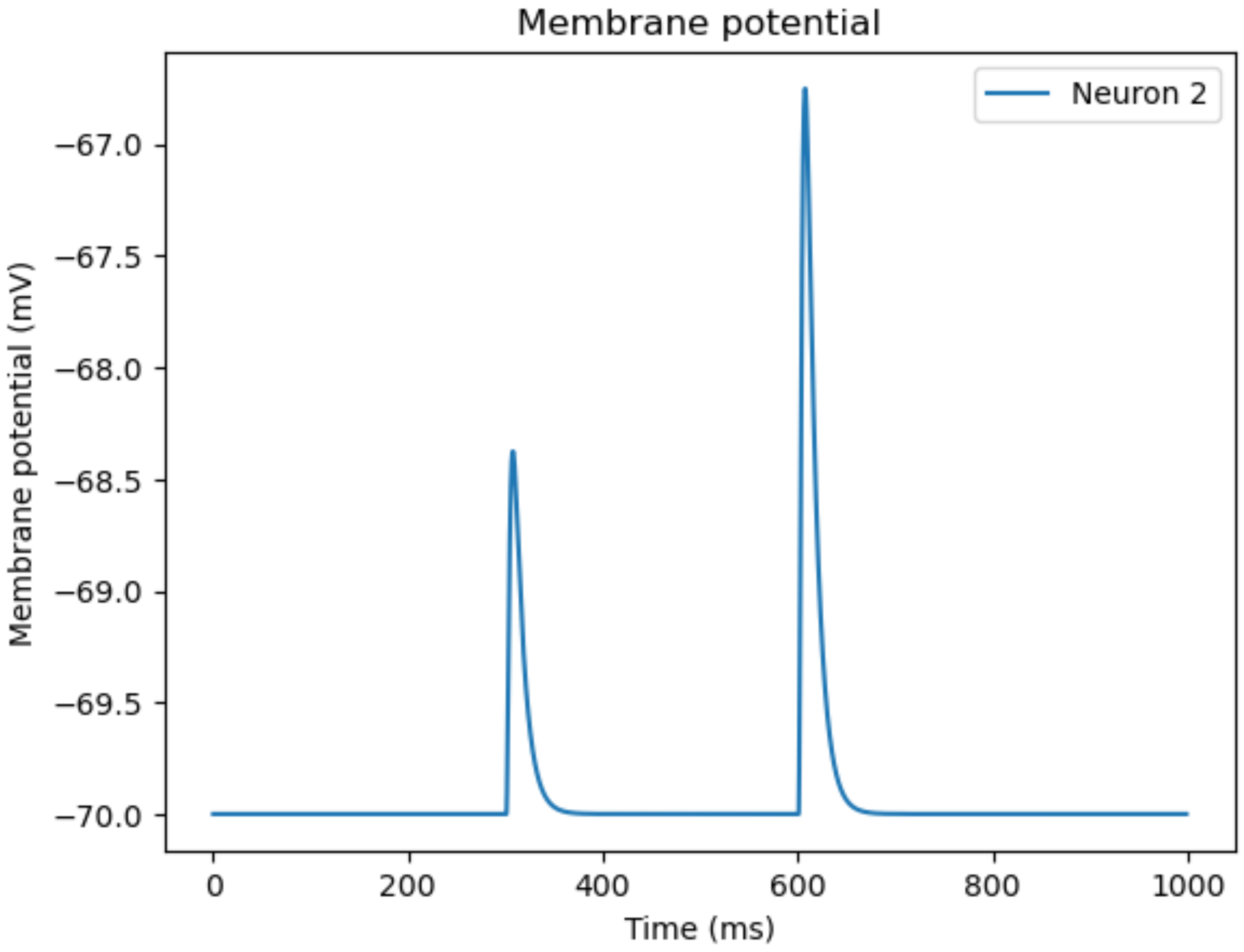

```
import nest
import numpy
import matplotlib.pyplot as plt

nest.ResetKernel()
# set simulation kernel
nest.SetKernelStatus({"local_num_threads": 1,
                      "resolution": 0.1,
                      "rng_seed": 1})
# Create nodes
# stimulator, spike input
sg1 = nest.Create("spike_generator", 1, params={
    "spike_times": [300, 600],
    "spike_weights": [10, 20],})
# neuron
n1 = nest.Create("iaf_psc_alpha", 1)
# recorder
vm1 = nest.Create("voltmeter", 1, params={"interval": 0.1,})

# Connect nodes
nest.Connect(sg1, n1, syn_spec = {"weight": 12.5,})
nest.Connect(vm1, n1)

# Run simulation
nest.Simulate(1000)
# plot out the recorded membrane potential
nest.voltage_trace.from_device(vm1)

times = vm1.events['times']
Vm = vm1.events['V_m']
plt.plot(times, Vm)
```



Install NEST

You can install NEST following the instructions by clicking the link:

Cross-platform

- Docker
- Conda

Linux

- Ubuntu
- Debian

MacOS

- via *“brew install nest”*