



# NEST+NESTML Tutorial

NICE Conference 2025 – Heidelberg  
24.–28. March



Charl Linssen | Sebastian Spreizer | Dennis Terhorst

# Our plan for today

## 1. Introduction to **NEST** (Dennis)

- Hands-on: NEST Desktop (Sebastian)

## 2. Introduction to **eprop** in **NEST** (Charl)

- Hands-on: Learning a sine wave

## 3. Introduction to **NESTML** (Charl)

- Hands-on:
  - create custom neuron model
  - create custom synaptic plasticity rule

(120 min)

# NEST Simulator

- 25+ years of experience
- 120+ developers
- Model-independent simulation engine
- Python based user interface, C++ kernel
- Multi-threading and multi-processing to use machines efficiently
- Laptop to super computers



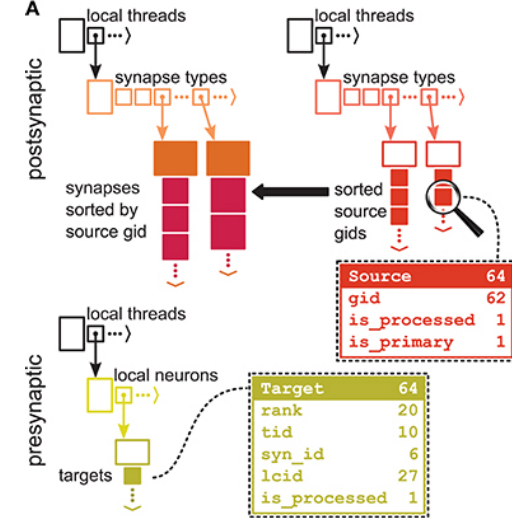
## NEST and friends

- NEST Simulator – main engine
  - pyNEST – Python interface
- NESTML – Modeling language
- NEST Desktop – teaching front-end
- PyNN – simulator independent language
- MUSIC – interface to other simulators
- NEAT – bridging to compartmental models
- ...



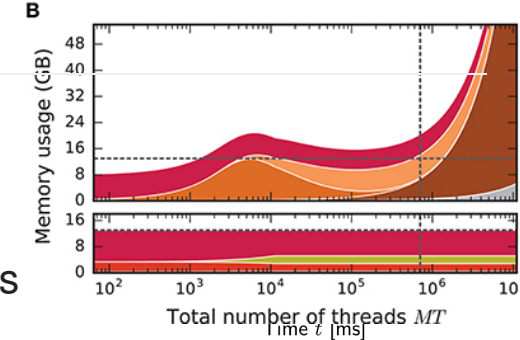
# Why is NEST relevant for neuromorphic?

- Hardware–software co-design
- Synergies for solutions
- Competition between technologies
- Verification and validation



## Why NEST – flexible preview of neuroscience

- exchangeable neuron/synapse/network models  
→ learn about explored mechanisms
  - gap junctions
  - astrocytes
  - ...
- investigate requirement limits, for example [1]
  - weight resolution
  - update timesteps





## Why is NEST relevant for neuromorphic?

- Hardware–software co-design
- Synergies for solutions
- Competition between technologies
- Verification and validation



“build the thing right” and “*build the right thing*”

---

## Why NEST – flexible preview of neuroscience

- exchangable neuron/synapse/network models  
→ learn about explored mechanisms
  - gap junctions
  - astrocytes
  - ...
- investigate requirement limits, for example [1]
  - weight resolution
  - update timesteps



## **Why NEST – co-design: different world, similar solutions**

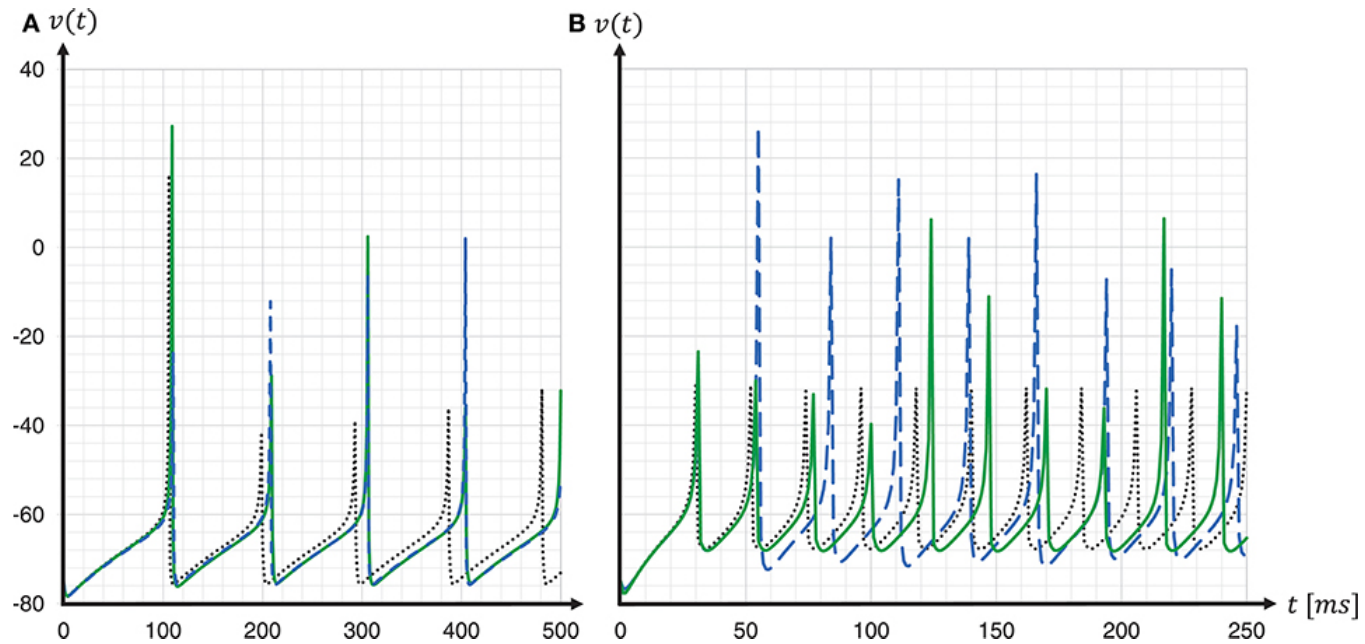
- model implementations
- architectures for handling data
- communication structures
- memory bottlenecks

## Why NEST – cross-check from different perspective

- cross-validation and verification
  - mathematical models
  - software
  - hardware

## Verification – example 1: Spike timing

With and without fixed-point data type conversion



Trensch et al., Rigorous Neural Network Simulations: A Model Substantiation Methodology for Increasing the Correctness of Simulation Results in the Absence of Experimental Validation Data – [10.3389/fninf.2018.00081](https://doi.org/10.3389/fninf.2018.00081)

## Verification – example 2: Integration method

### Comparison of simulation engines

- numerical solver can be biased, leading to different firing rates for certain types of models

Plesser, Commentary: Accelerating spiking neural network simulations with PymoNNto and PymoNNtorch – [10.3389/fninf.2024.1446620](https://doi.org/10.3389/fninf.2024.1446620)

## Memory & Communication – example 3: NEST kernel

### Example

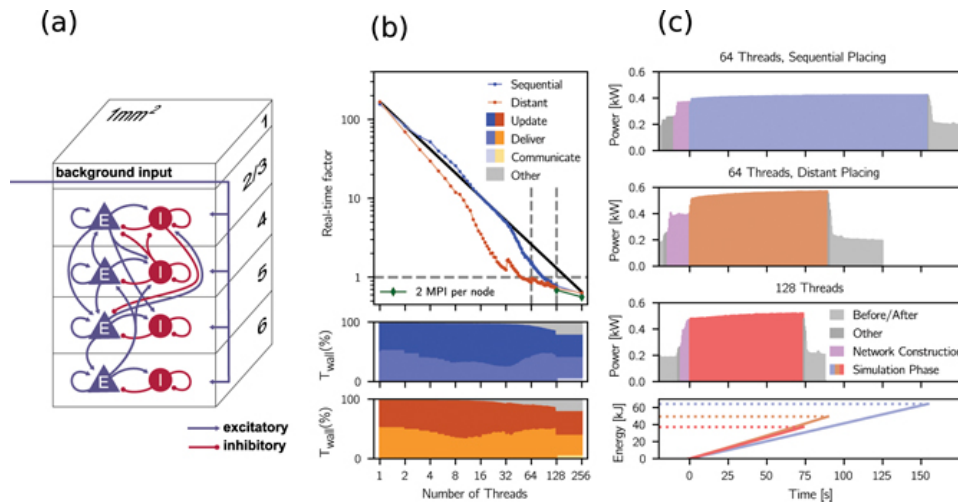
- internal division of communication structures
- minimized required synchronization points

Figure from: Jordan et al., Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers – [10.3389/fninf.2018.00002](https://doi.org/10.3389/fninf.2018.00002)

## Validation – example 4: Time and energy

- Is NC really faster?
- Is NC really more energy-efficient?

Microcircuit (PD14) as standard benchmark leading to comparability and constructive competition.



RTF	$E_{\text{syn-event}}$ ( $\mu\text{J}$ )	References
6.29	4.39	2018, NEST [2]
2.47	9.35	2018, NEST [2]
26.08	0.30	2018, GeNN [3]
1.84	0.47 <sup>a</sup>	2018, GeNN [3]
1.00	0.60	2019, SpiNNaker [8]
1.06	—	2021, NeuronGPU [9]
0.70	—	2021, GeNN [10]
0.67	0.33	NEST, AMD EPYC Rome (one node, 2 MPI)
0.53	0.48	NEST, AMD EPYC Rome (two nodes, 4 MPI)

Kurth et al., Sub-realtime simulation of a neuronal network of natural density – [10.1088/2634-4386/ac55fc](https://doi.org/10.1088/2634-4386/ac55fc)

## Why NEST – Summary

- offers a lot of opportunities and experience in an active and interested community
- established scalable platform also for reference implementations of future hardware



**EBRAINS**

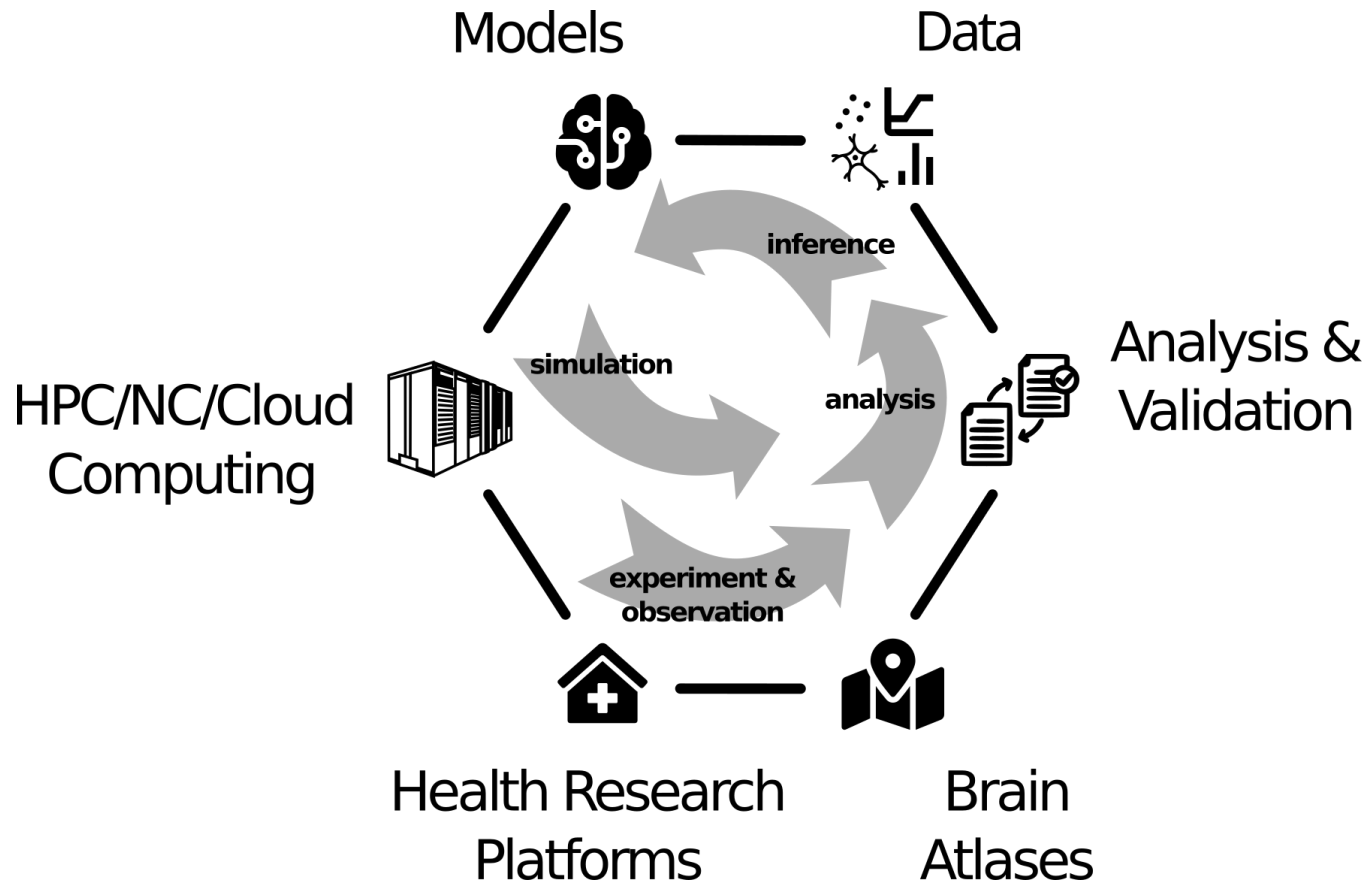
<https://ebrains.eu/register>



- successor of HBP
- digital research infrastructure
- provides interoperable tools, services and resources
- curated ecosystem
- <https://ebrains.eu/register>



EBRAINS





## Technical components

- Collaboration: knowledge graph, drive, wiki, ...
- Compute: Jupyter lab, HPC, cloud
- Storage: data, atlas, ...
- Software: EBRAINS Software Disistribution
- Services: gitlab, notes, office, ...

Single sign-on.

<https://ebrains.eu>

**EBRAINS** Infrastructure About Focus areas News & events Contact [Sign in](#)

**An open research infrastructure that gathers data, tools and computing facilities for brain-related research, built with interoperability at the core.**

[Get started ▶](#)

- Data**
- Brain atlases**
- Modelling, simulation & computing**
- Validation & inference**
- Health research platforms**

## NEST Desktop

- originally for teaching
- low threshold, graphical, web-based
- quick prototyping
- simple analyses

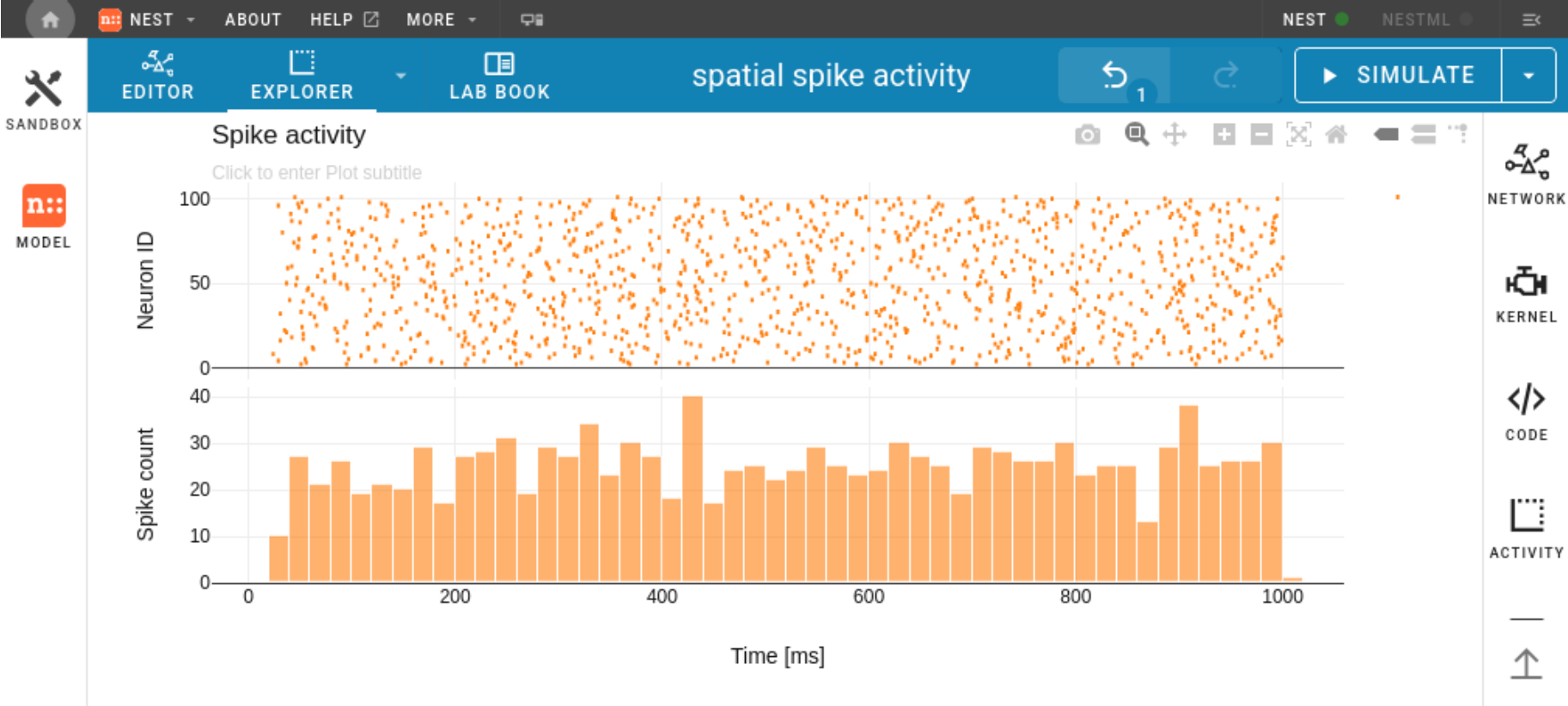
<https://nest-desktop.apps.ebrains.eu>



# NEST Desktop - Construct network models

The screenshot displays the NEST Desktop interface for constructing a network model. The top navigation bar includes 'EDITOR', 'EXPLORER', and 'LAB BOOK' tabs, with the current view being 'spatial spike activity'. A 'SIMULATE' button is visible on the right. The main workspace shows a network diagram with three components: a Poisson generator (PG1), a neuron (N1), and a spike recorder (SR1). A blue arrow labeled '10' connects PG1 to N1, and an orange arrow connects N1 to SR1. The right sidebar contains a 'PROPERTY' panel with two sections: 'stimulator model' (Poisson generator) and 'neuron model' (IAF psc alpha). The 'stimulator model' section shows a 'mean firing rate' slider set to 6500 Hz. The 'neuron model' section shows 'Positions' set to 2D and 100. Below the property panel, a 'CONNECTIONS' panel shows 'all\_to\_all' connections between PG1 and N1, and N1 and SR1. The bottom right sidebar has icons for NETWORK, KERNEL, CODE, and ACTIVITY.

# NEST Desktop - Simulated activity



# NEST Desktop – Code export

The screenshot displays the NEST Desktop interface. At the top, there is a navigation bar with 'NEST', 'ABOUT', 'HELP', and 'MORE' menus. Below this is a toolbar with 'EDITOR', 'EXPLORER', and 'LAB BOOK' tabs. The main workspace is titled 'spatial spike activity' and features a 'SIMULATE' button. On the left, a 'MODEL' sidebar shows a network diagram with three nodes: a hexagon labeled 'PG1', a triangle labeled 'N1', and a square labeled 'SR1'. A blue arrow points from 'PG1' to 'N1' with the weight '10', and an orange arrow points from 'N1' to 'SR1'. The right side of the interface contains a code editor with the following Python code:

```
1 import nest
2 import numpy as np
3
4 nest.ResetKernel()
5
6 # Set simulation kernel
7 nest.SetKernelStatus({
8     "local_num_threads": 1,
9     "resolution": 1,
10    "rng_seed": 1
11 })
12
13 # Create nodes
14 pg1 = nest.Create("poisson_generator", params={
15     "rate": 6500,
16 })
17 n1 = nest.Create("iaf_psc_alpha", 100,
18     positions=nest.spatial.free(
19         nest.random.uniform(-0.5, 0.5),
20         num_dimensions=2
21     )
22 )
```

On the far right, there is a vertical sidebar with icons for 'NETWORK', 'KERNEL', 'CODE', and 'ACTIVITY'.

# EBRAINS Jupyter

The screenshot displays the EBRAINS JupyterLab interface. At the top, there is a menu bar with options: File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. On the right side of the top bar, a memory usage indicator shows 'Mem: 703 / 2048 MB'. Below the menu bar is a toolbar with icons for file operations. The left sidebar contains a file browser with a search bar labeled 'Filter files by name'. The current directory is '/ ... / tutorials / stdp\_windows /'. A table lists files and folders:

Name	Last Modified
report	2 hours ago
target	2 hours ago
stdp_syna...	2 hours ago
stdp_vogel...	2 hours ago
stdp_wind...	2 hours ago
• stdp_wind...	2 hours ago

The main area shows a Jupyter notebook titled 'stdp\_windows.ipynb'. The notebook has a toolbar with icons for file operations and a 'Markdown' dropdown menu. The current cell is titled 'Running the simulation in NEST' and contains the following text:

Let's define a function that will instantiate a simple network with one presynaptic cell and one postsynaptic cell connected by a single synapse, then run a simulation and plot the results.

```
[5]: def run_network(pre_spike_time, post_spike_time,
                    module_name,
                    neuron_model_name,
                    synapse_model_name,
                    resolution=1., # [ms]
                    delay=1., # [ms]
                    lambda=1E-6,
                    sim_time=None, # if None, computed from pre and post spi
                    synapse_parameters=None, # optional dictionary passed to
                    fname_snip=""):
```

At the bottom of the interface, there is a status bar showing 'Simple' mode, a kernel icon, '0 s', '2' cells, 'master' branch, 'EBRAINS-25.02-rc | I...', 'Mem: 663.63 / 2048.00 ...', 'Mode: Comm...', 'Ln 1, C...', and 'stdp\_windows.ip...'.

<https://lab.ebrains.eu> – All tools installed and tested.

## Hands on 🙌

Getting started with NEST

- ➔ NEST Desktop: <https://nest-desktop.apps.ebrains.eu>
  - create balanced networks; export to Python script
- ➔ EBRAINS Jupyter: <https://lab.ebrains.eu>
  - run same simulation; add more analysis



## **e-prop in NEST: Bridging to machine learning**

< Charl will present >

look for Korcsak-Gorzo and Espinoza Valverde

## Hands on 🙌

Lot's of cool stuff!

-  NEST examples: <https://nest-simulator.org/documentation>
  -  *E-prop plasticity examples: “generating sine waves”*

⇒ Try it on  EBRAINS

# NESTML

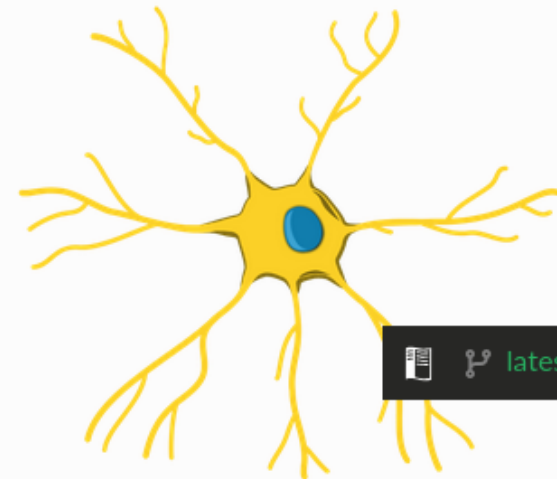
🏠 / The NESTML modeling language

## The NESTML modeling language

NESTML is a domain-specific language for neuron and synapse models. These dynamical models can be used in simulations of brain activity on several platforms, in particular the **NEST Simulator**.

NESTML combines:

- an easy to understand, yet powerful syntax;
- a flexible processing toolchain,



nest::ml

Search docs

NESTML language documentation

Installing NESTML

Running NESTML

Models library

Tutorials

Extending NESTML

Getting help

Citing NESTML

<https://nestml.readthedocs.io>

# NESTML models library



Search docs

NESTML language documentation

Installing NESTML

Running NESTML

Models library

Neuron models

Synapse models

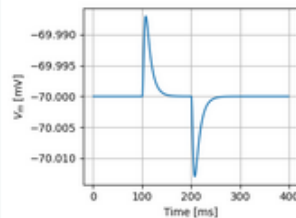
Tutorials

Extending NESTML

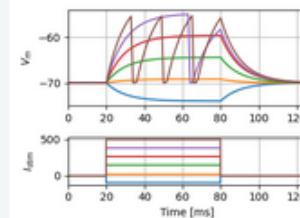
## iaf\_psc\_alpha\_neuron

iaf\_psc\_alpha - Leaky integrate-and-fire neuron model

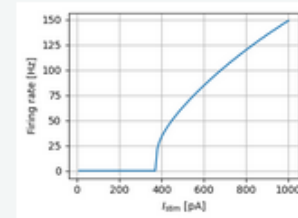
Source file: [iaf\\_psc\\_alpha\\_neuron.nestml](#)



Post-synaptic potential



Step current response



Firing rate vs. current

## iaf\_psc\_delta\_neuron

iaf\_psc\_delta - Current-based leaky integrate-and-fire neuron model with delta-kernel post-synaptic currents

[latest](#)

# Model description and code generation



[NESTML language documentation](#)

[Installing NESTML](#)

[Running NESTML](#)

[Models library](#)

[Tutorials](#)

[Extending NESTML](#)

[Getting help](#)

[Citing NESTML](#)

```
[2]: nestml_ou_model = """
model ornstein_uhlenbeck_noise_neuron:

    parameters:
        mean_noise real = 500    # mean of the noise
        sigma_noise real = 50    # std. dev. of the noise
        tau_noise ms = 20 ms     # time constant of the noise

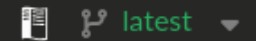
    internals:
        A_noise real = sigma_noise * ((1 - exp(-2 * resolution() / tau_noise))

    state:
        U real = mean_noise     # set the initial condition

    update:
        U = mean_noise \
          + (U - mean_noise) * exp(-resolution() / tau_noise) \
          + A_noise * random_normal(0, 1)
"""
```


Save to a temporary file and make the model available to instantiate in NEST:

```
[3]: # generate and build code
module_name, neuron_model_name_adapt_curr = \
    NESTCodeGeneratorUtils.generate_code_for(nestml_ou_model)
```



## Hands on 🙌

NESTML hands-on

-  Creating a new neuron model: <https://lab.ebrains.eu>
  - clone <https://github.com/nest/nestml>

⇒ Try it on  EBRAINS

# Wrap-up

## What we have seen

- ✓ Overview of NEST and its links to other tools and projects
- ✓ NEST Desktop and EBRAINS JupyterLab
- ✓ Eprop in NEST
- ✓ NESTML and first steps in creating neuron and synapse models

# Where to go from here...

# virtual

nest::  
conference

# 2025

17-18  
June

nest::

spiking  
neural  
network  
simulator

Forschungszentrum Jülich / Ralf-Uwe Limbach / SBC Lehmann

wide scalability

Capone et al., 2019; Potjans, 2014

open model structure

Capone et al., 2019; Senk et al., 2018

scientific insight

# Where to go from here...

 **NEST Conference 2025**

<https://nest-simulator.org/conference>

 **NEST-related publications**

<https://nest-simulator.org/publications>

 **PyNN**

<https://neuralensemble.org/docs/PyNN/>

 **More tools**

<https://www.ebrains.eu/tools>

 **Feed-back**

<https://www.surveymonkey.com/r/NICE-NEST>