

---

# **NEST Desktop**

**Sebastian Spreizer**

**Mar 06, 2023**



# CONTENTS

<b>1</b>	<b>Conceptual approach</b>	<b>3</b>
<b>2</b>	<b>Content structure</b>	<b>5</b>
<b>3</b>	<b>Version info</b>	<b>7</b>
3.1	About . . . . .	7
3.2	Troubleshootings . . . . .	26
3.3	User guide . . . . .	32
3.4	Lecturer guide . . . . .	87
3.5	Deployer guide . . . . .	90
3.6	Developer guide . . . . .	99





Hello there! :-)

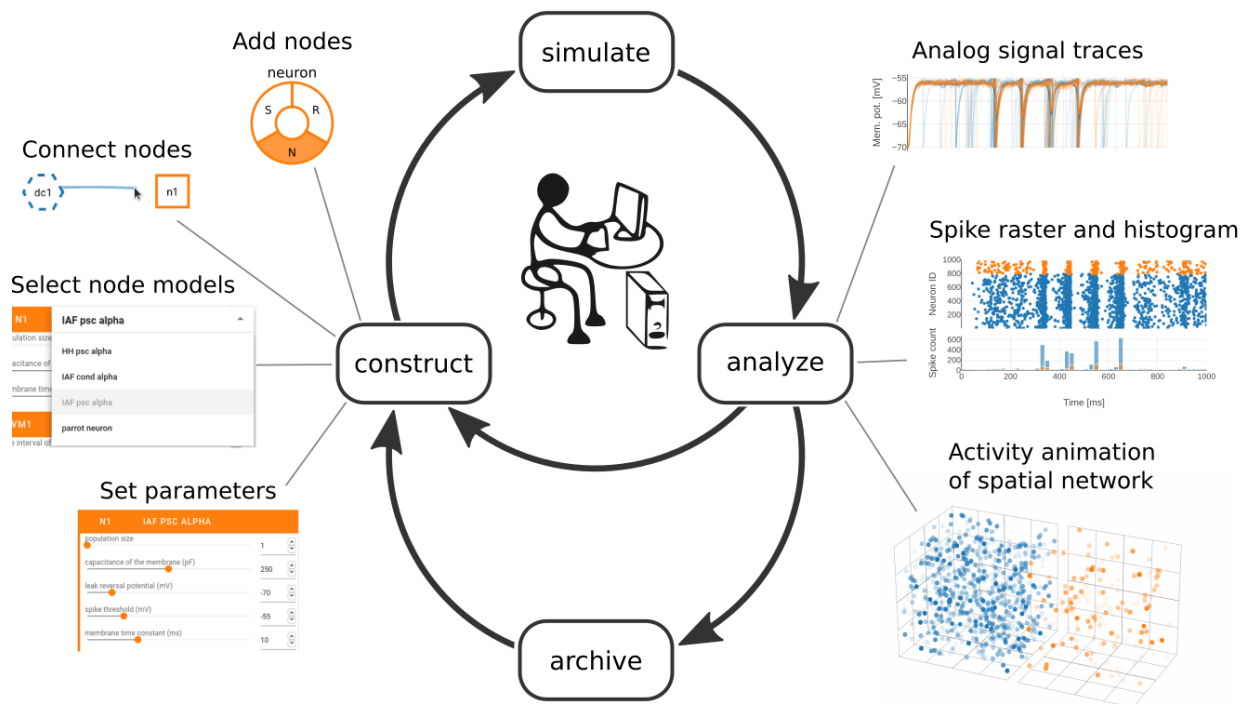
NEST Desktop is a web-based GUI application for NEST Simulator, an advanced simulation tool for the computational neuroscience.

It's so great that you want to use NEST Desktop!



## CONCEPTUAL APPROACH

NEST Desktop enables to construct a neuronal network model graphically and to perform a simulation experiment. Thus, no programming skills are required.



You can tryout NEST Desktop as a restricted [live demo](#) without the simulation backend.





## CONTENT STRUCTURE

The documentation is organized in four sections. Select the appropriate section that fits your needs:



## VERSION INFO

On ReadTheDocs, it is possible to select versions of this documentation. These versions basically relate to the program versions (as found in the GitHub repository). This can be noticed when clicking on the “Edit on GitHub” text at the top right.

Since the changes between patch level versions (e.g. 3.0.0 and 3.0.1) are usually not noticeable, we show a single branch for the recent minor version releases (e.g. ‘3.0’ for all 3.0.x releases). This branch contains all patch releases and always points to the latest patch release of that release branch. Please keep this in mind when searching for information on a specific version of NEST Desktop!

---

**General**

**GitHub**

**Docker**

**Python**

**Conda**

**AppImage**

**Snap**

## 3.1 About

### 3.1.1 Events

#### 2023

##### 13 - 24 Feb

BSc course “Simple Neuron Models” at BCF<sup>1</sup>, Freiburg, Germany.

##### 18 Jan

Part of the talk at CENIA, presented by Markus Diesmann, Santiago, Chile.

---

<sup>1</sup> BCF - Bernstein Center Freiburg, Faculty of Biology, University of Freiburg, Freiburg, Germany

### 18 - 20 Jan

Poster at [7th HBP Student Conference](#), Madrid, Spain.

## 2022

### 07 - 10 Nov

Session talk “NEST Desktop” at [Simulate with EBRAINS](#), online.

### 13 - 16 Sep

Poster “NEST Desktop: Explore new frontiers” at [Bernstein Conference 2021](#), Berlin, Germany.

### 20 - 21 Jul

Talk “NEST Desktop” at HCI Summer Colloquium, Trier, Germany.

### 16 Jul

NEST Desktop is part of the onsite tutorial “T1: From single-cell modeling to large-scale network dynamics with NEST Simulator.” of [CNS 2022](#), presented by Jasper Albers, Pooja Babu and Charl Linssen, Melbourne, Australia.

### 03 - 09 Jul

Part of the workshop “NEST Workshop: network and plasticity” at [9th BNNI](#), presented by Jasper Albers, Krakau, Poland.

### 31 Jun

Satellite tutorial at [CNS](#), online.

### 23 - 24 Jun

Workshop “NEST Desktop: A “Let’s Play Together” for neuroscience” and Poster “NEST Desktop: Explore new frontiers” at [NEST Conference](#), online.

### 13 - 15 Jun

Workshop at [EBRAINS BASSES](#), presented by Johanna Senk, Rome, Italy.

### 26 - 28 Apr

MSc course “Biophysics of Neurons and Networks” at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

### 07 - 18 Feb

BSc course “Simple Neuron Models” at BCF<sup>Page 7, 1</sup>, online, (Freiburg, Germany).

## 2021

### 16 Dec

Talk “NEST Desktop” at HCI Winter Colloquium, online, (Trier, Germany).

### 11 Nov

Published paper “NEST Desktop, an Educational Application for Neuroscience” on [eNeuro](#).

### 14 - 15 Oct

Break-out session and poster “NEST Desktop” at [HBP Summit 2021](#), online (Brussels, Belgium).

### 22 Sep

Talk “Simulation of networks with point neurons (NEST)” at [8th BNNI 2021](#), online.

### 21 - 23 Sep

Poster “NEST Desktop” at [Bernstein Conference 2021](#), online.

### 03 Sep

Hand-on Session “NEST Desktop” at [EBRAINS & IBRO 2nd Virtual Master Class](#), online.

**29 Jul**

Hand-on Session “NEST Desktop” at [EBRAINS & IBRO 1st Virtual Master Class](#), online.

**13 Jul**

Talk “NEST Desktop” at [PhD Seminar](#), online.

**06 Jul**

Talk “NEST Desktop” at [NFDI-Neuro Webinar](#), online.

**03 Jul**

Tutorial “Interactive design and analysis of point neuron spiking networks with synaptic plasticity using NEST Simulator”, presented by Dr. Linssen, at [CNS 2021](#), online.

**28 - 29 Jun**

Talk “NEST Desktop” at [NEST Conference](#), online (As, Norway).

**16 Jun**

Preprint on [bioRxiv](#).

**03 - 07 May**

MSc course “Biophysics of Neurons and Networks” at BCF<sup>Page 7, 1</sup>, online (Freiburg, Germany).

**08 - 09 Apr**

“NEST Desktop insitufication” on In-Situ Hackathon, online (HCI).

**08 - 19 Feb**

BSc course “Simple Neuron Models” at BCF<sup>Page 7, 1</sup>, online, (Freiburg, Germany).

**2020****30 Sep - 01 Oct**

Hand-on Session and Poster at [Bernstein Conference 2020](#), online (Berlin, Germany).

**18 - 22 Jul**

Tutorial with NESTML, presented by Dr. Linssen, at [CNS 2020](#), online (Melbourne, Australia).

**29 - 30 Jun**

Talk “NEST Desktop” at [NEST Conference](#), online (As, Norway).

**02 - 17 Jun**

MSc course “Biophysics of Neurons and Networks” at BCF<sup>Page 7, 1</sup>, online (Freiburg, Germany).

**16 Apr**

Presentation and demo at [NeuroMat](#), online (Sao Paulo, Brazil).

**03 - 06 Feb**

Talk and Demo/Hand-on session at [HBP Summit and Open Days](#), Athene, Greece.

**2019****28 Nov**

2nd HPAC Platform Training, Heidelberg, Germany.

**20 Oct**

Live demo, presented by Prof. Plesser, at HBP Booth at SfN, Chicago, USA.

**18 - 20 Sep**

Poster/Live presentation at [Bernstein Conference](#), Berlin, Germany.

### 22 Jul

Talk and Tutorial/Hand-on session at INM-6<sup>3</sup>, Jülich, Germany.

### 18 Jul

NESTML/NEST-desktop integration workshop, BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

### 24 - 25 Jun

Talk and Tutorial/Hand-on session “NEST Desktop” at NEST Conference at NMBU<sup>4</sup>, As, Norway.

### 16 Apr

Kick-Off workshop at HCI<sup>2</sup>, Trier, Germany.

### 25 - 31 Mar

Tutorial workshop for IICSSS at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

### 11 - 22 Feb

BSc course “Simple Neuron Models” at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

## 2018

### 26 - 27 Sep

Poster/Live presentation NEST Desktop at Bernstein Conference, Berlin, Germany.

### 27 - 28 Aug

Technical meeting at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

### 25 - 26 Jun

Talk “NEST Web API” at NEST Conference at NMBU<sup>4</sup>, As, Norway.

### 23 - 27 Apr

MSc course “Biophysics of Neurons and Networks” at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

### 12 - 23 Feb

BSc course “Simple Neuron Models” at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

## 2017

### 19 - 20 Dec

Talk “NEST Desktop” at NEST Conference, Jülich, Germany.

### 20 - 22 Nov

Live presentation with Ad Aertsen at Neural networks mini school, Strasbourg, France.

### 02 - 05 May

MSc course “Biophysics of Neurons and Networks” at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

### 24 Jan

Talk (Informal Seminar) “NEST Desktop” at BCF<sup>Page 7, 1</sup>, Freiburg, Germany.

---

<sup>3</sup> INM-6 - Institute of Neuroscience and Medicine (INM-6), Jülich Research Center, Jülich, Germany

<sup>4</sup> NMBU - Norwegian University of Life Sciences, As, Norway

<sup>2</sup> HCI - Human-Computer Interaction - Department IV - Computer Science, University of Trier, Trier, Germany

2016

Dec

The development start of NEST Desktop.

## 3.1.2 Releases

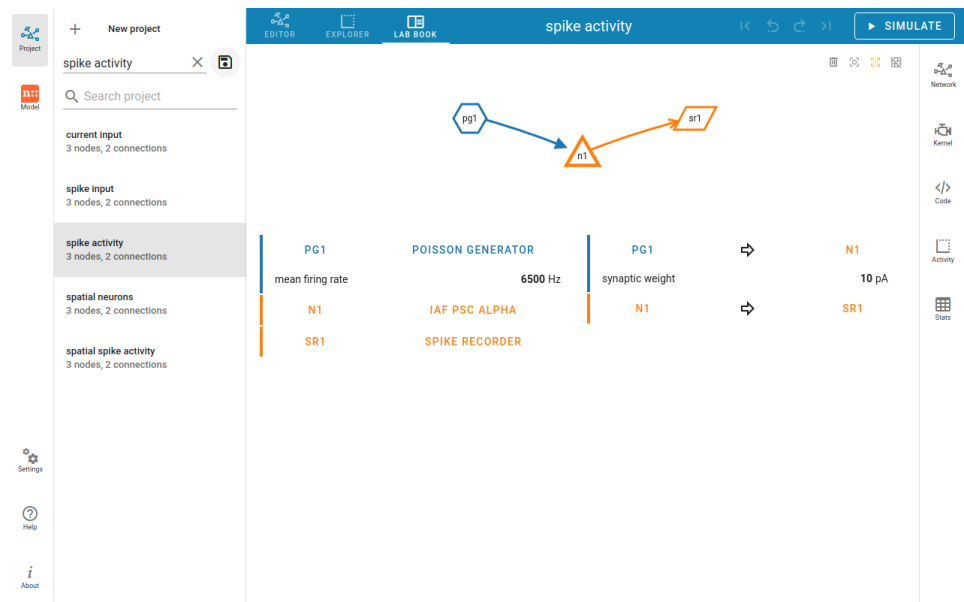
### v3.2.x

Introduce copy models, compartmental neurons, synapse weight recordings, code templates and backend status.

28 Feb 23	v3.2.0
-----------	--------

### v3.1.x

Introduce model management, simulation option with Insite.

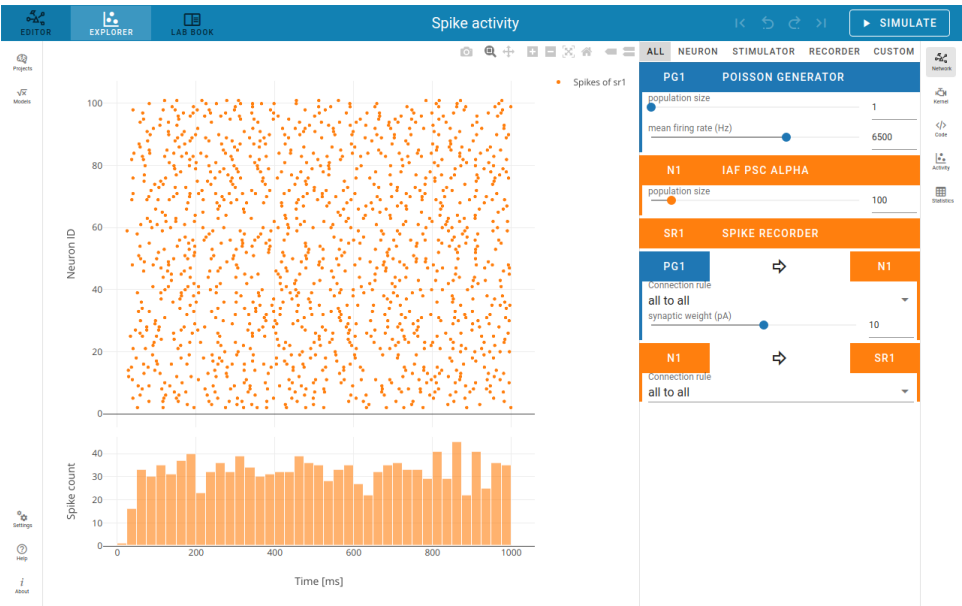
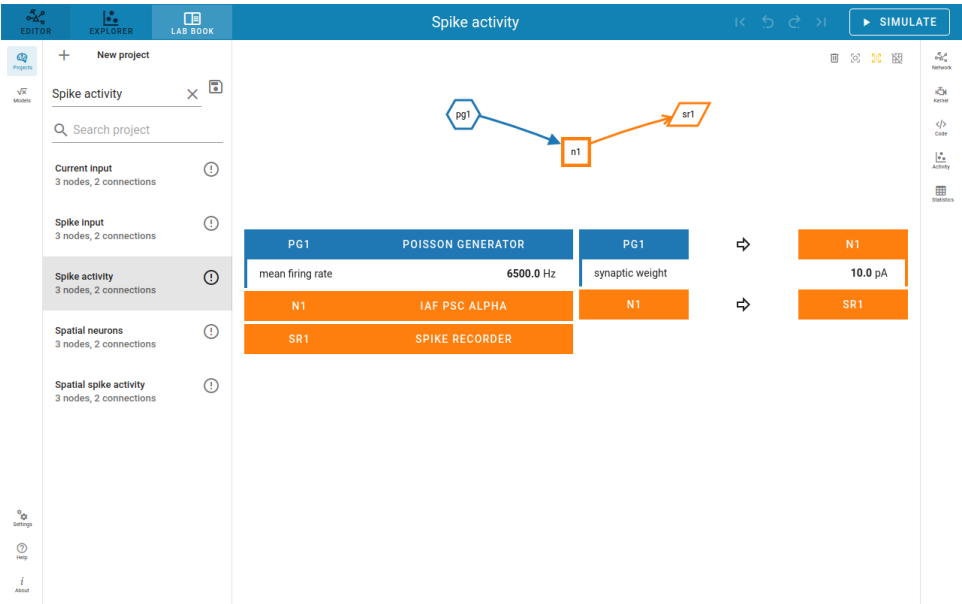




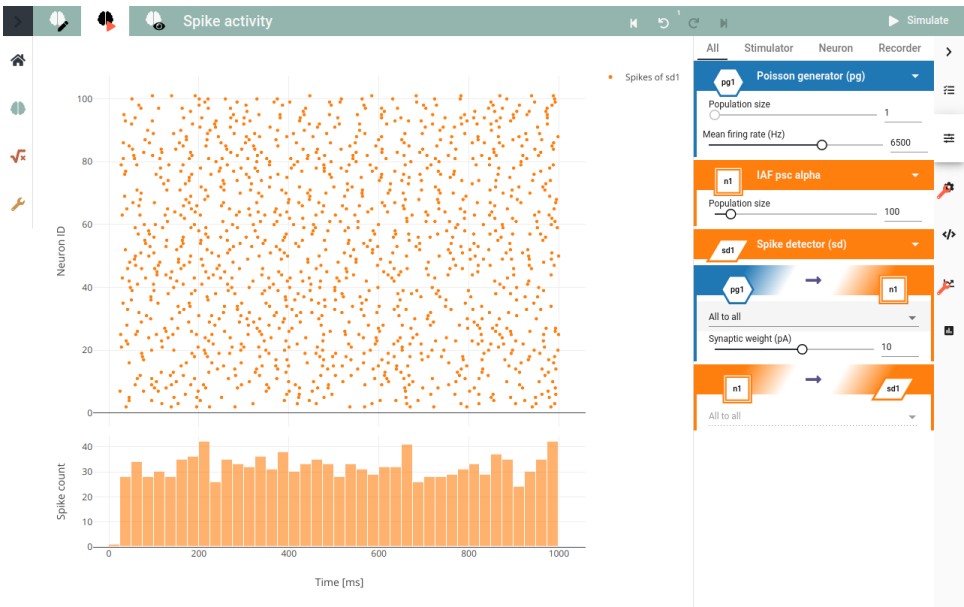
01 Jun 22	v3.1.4
20 May 22	v3.1.3
25 Feb 22	v3.1.2
22 Feb 22	v3.1.1
18 Feb 22	v3.1.0



v3.0.x



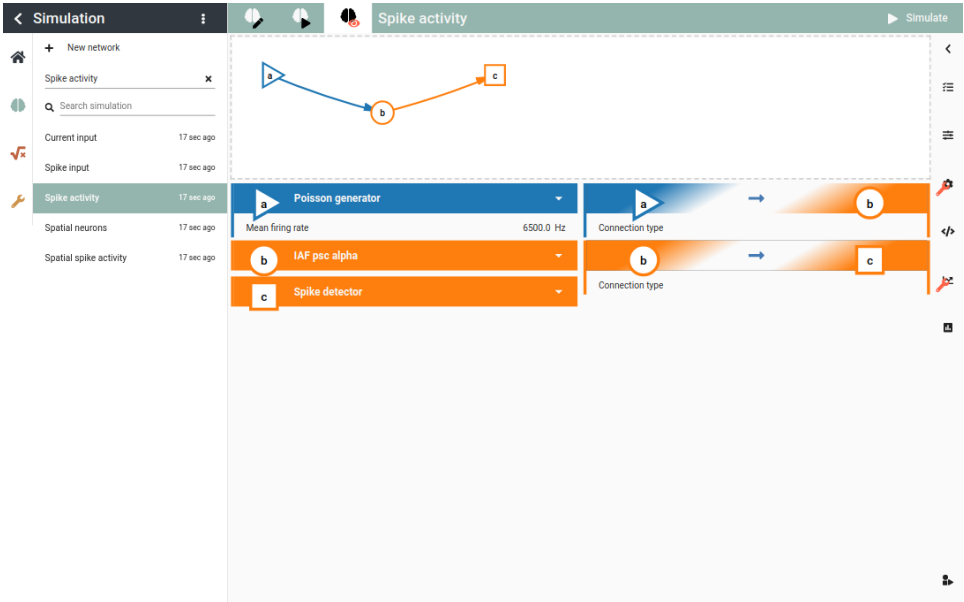


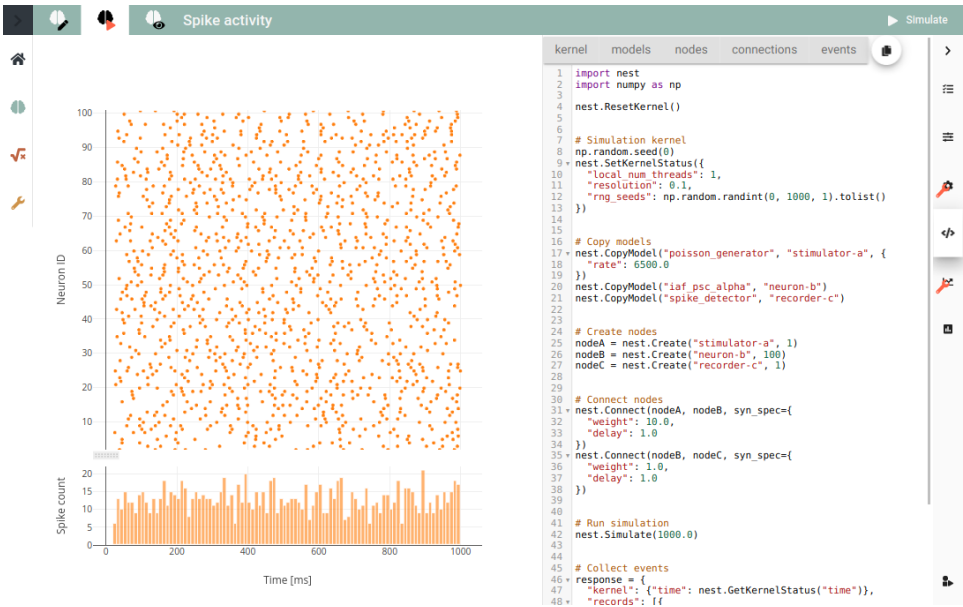


Define node shapes according to graphical notation of neuronal networks. Introduce network history to undo changes.

04 Mar 21	v2.5.1
23 Oct 20	v2.5.0

v2.4.x





Introduce code editor for simulation script.

15 Jul 20	v2.4.1
28 Jun 20	v2.4.0

v2.3.x

Deploy NEST Desktop on HBP service with OIDC and on bwCloud with ansible. Use yarn instead of npm.

23 May 20	v2.3.2
22 May 20	v2.3.1
22 May 20	v2.3.0

v2.2.x

**Simulation**

- + New network
- Spike activity x
- List of saved simulations
- Search simulation
- Current input 2 sec ago
- Spike input 2 sec ago
- Spike activity 2 sec ago
- Spatial neurons 2 sec ago
- Spatial spike activity 2 sec ago

**Spike activity**

**a Poisson generator**

Mean firing rate	6500.0 Hz
Start time	0.0 ms
Stop time	10000.0 ms

**b IAF psc alpha**

Capacitance of the membrane	250.0 pF
Leak reversal potential	-70.0 mV
Constant external input current	0.0 pA
Initial membrane potential	-70.0 mV
Reset potential of the membrane	-70.0 mV
Spike threshold	-55.0 mV
Duration of refractory period	2.0 ms
Membrane time constant	10.0 ms
Rise time of the excitatory synaptic alpha function	2.0 ms
Rise time of the inhibitory synaptic alpha function	2.0 ms

**c Spike detector**

Start time	0.0 ms
Stop time	10000.0 ms

**Connection rules:**

- a → b: Connection rule: all\_to\_all; Synaptic model: static\_synapse; Synaptic weight: 10.0 pA; Synaptic delay: 1.0 ms
- b → c: Connection rule: all\_to\_all; Synaptic model: static\_synapse; Synaptic weight: 1.0 pA; Synaptic delay: 1.0 ms

**Spike activity**

**a Poisson generator**

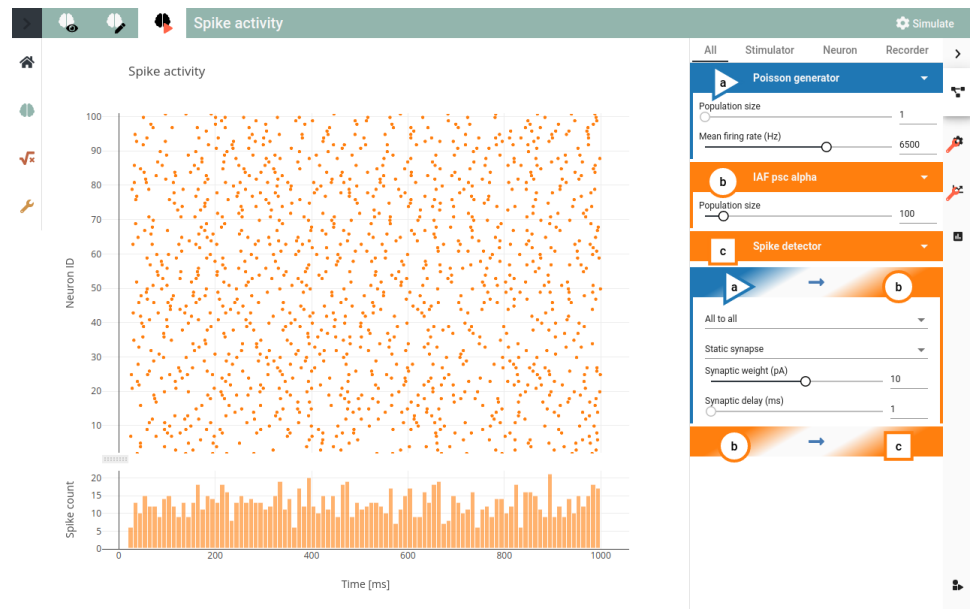
Population size	1
Mean firing rate	Hz <input checked="" type="checkbox"/>
Start time	ms <input type="checkbox"/>
Stop time	ms <input type="checkbox"/>

**b IAF psc alpha**

Population size	100
Capacitance of the membrane	pF <input type="checkbox"/>
Leak reversal potential	mV <input type="checkbox"/>
Constant external input current	pA <input type="checkbox"/>
Initial membrane potential	mV <input type="checkbox"/>
Reset potential of the membrane	mV <input type="checkbox"/>
Spike threshold	mV <input type="checkbox"/>
Duration of refractory period	ms <input type="checkbox"/>
Membrane time constant	ms <input type="checkbox"/>
Rise time of the excitatory synaptic alpha function	ms <input type="checkbox"/>
Rise time of the inhibitory synaptic alpha function	ms <input type="checkbox"/>

**c Spike detector**

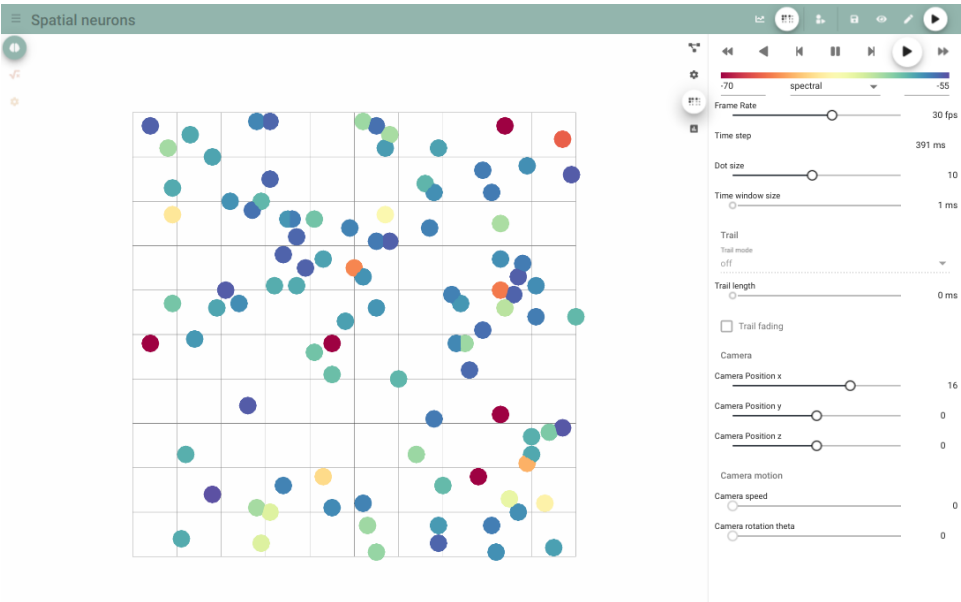
Start time	ms <input type="checkbox"/>
Stop time	ms <input type="checkbox"/>



Introduce tabs for project views as well as side bars for navigation and controller.

27 Jan 20	v2.2.15
27 Jan 20	v2.2.14
20 Jan 20	v2.2.13
16 Jan 20	v2.2.12
30 Dec 19	v2.2.11
04 Dec 19	v2.2.10
04 Dec 19	v2.2.9
03 Dec 19	v2.2.8
27 Nov 19	v2.2.7
27 Nov 19	v2.2.6
27 Nov 19	v2.2.5
24 Nov 19	v2.2.4
24 Nov 19	v2.2.3
24 Nov 19	v2.2.2
21 Nov 19	v2.2.1
08 Nov 19	v2.2.0

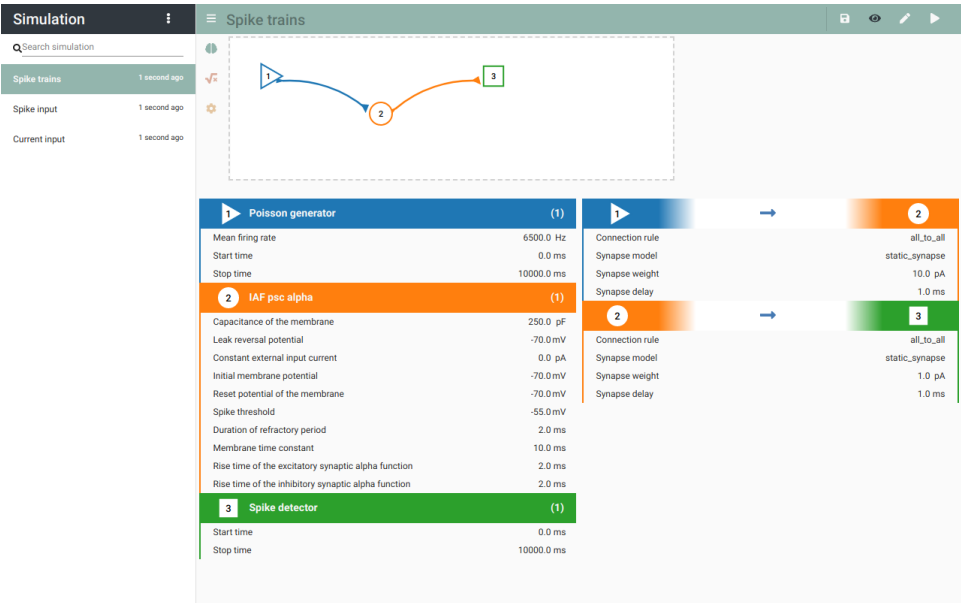
v2.1.x

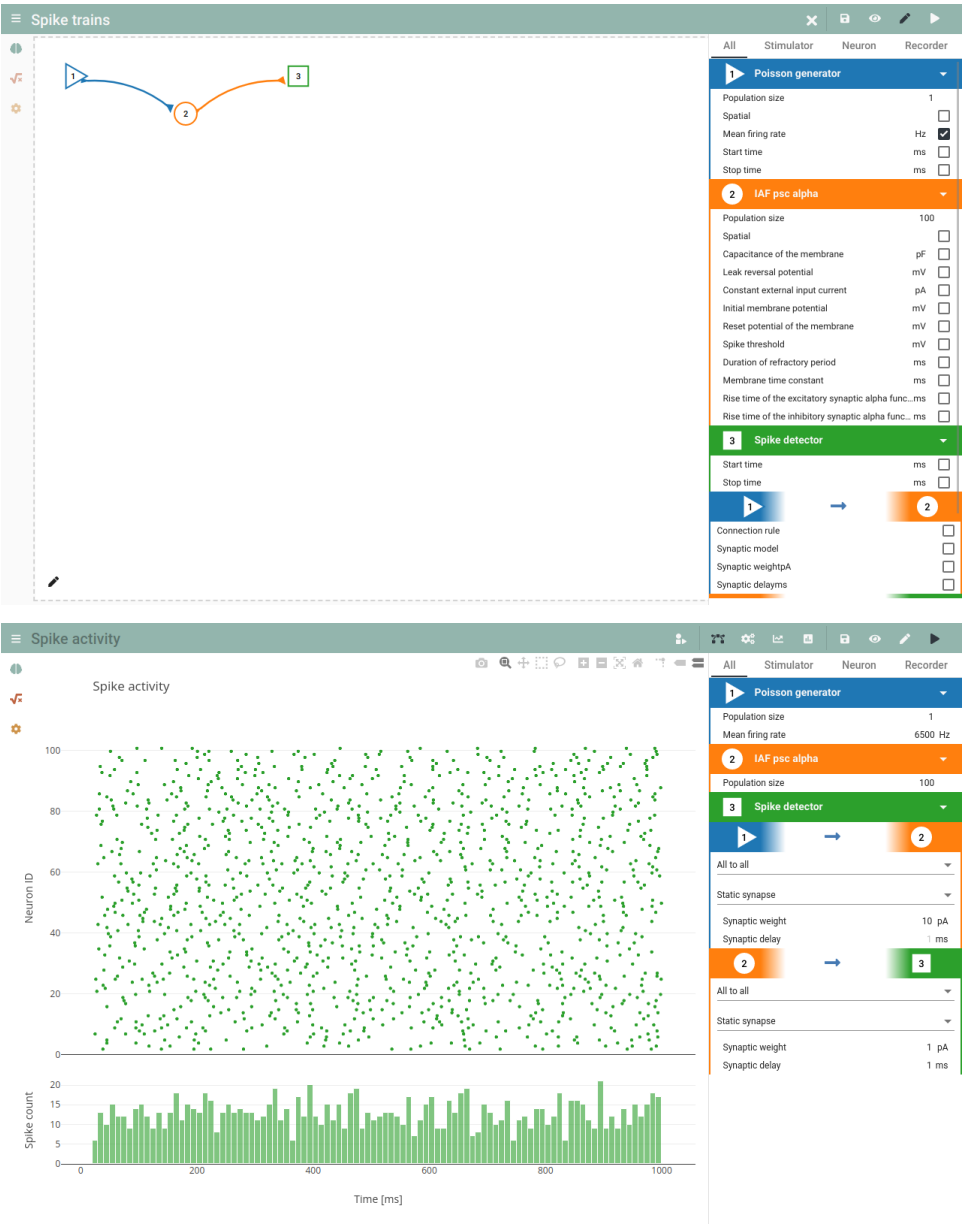


Introduce Three.js for animated activity graph of spatial network.

05 Nov 19	v2.1.3
05 Nov 19	v2.1.2
04 Nov 19	v2.1.1
29 Oct 19	v2.1.0

v2.0.x



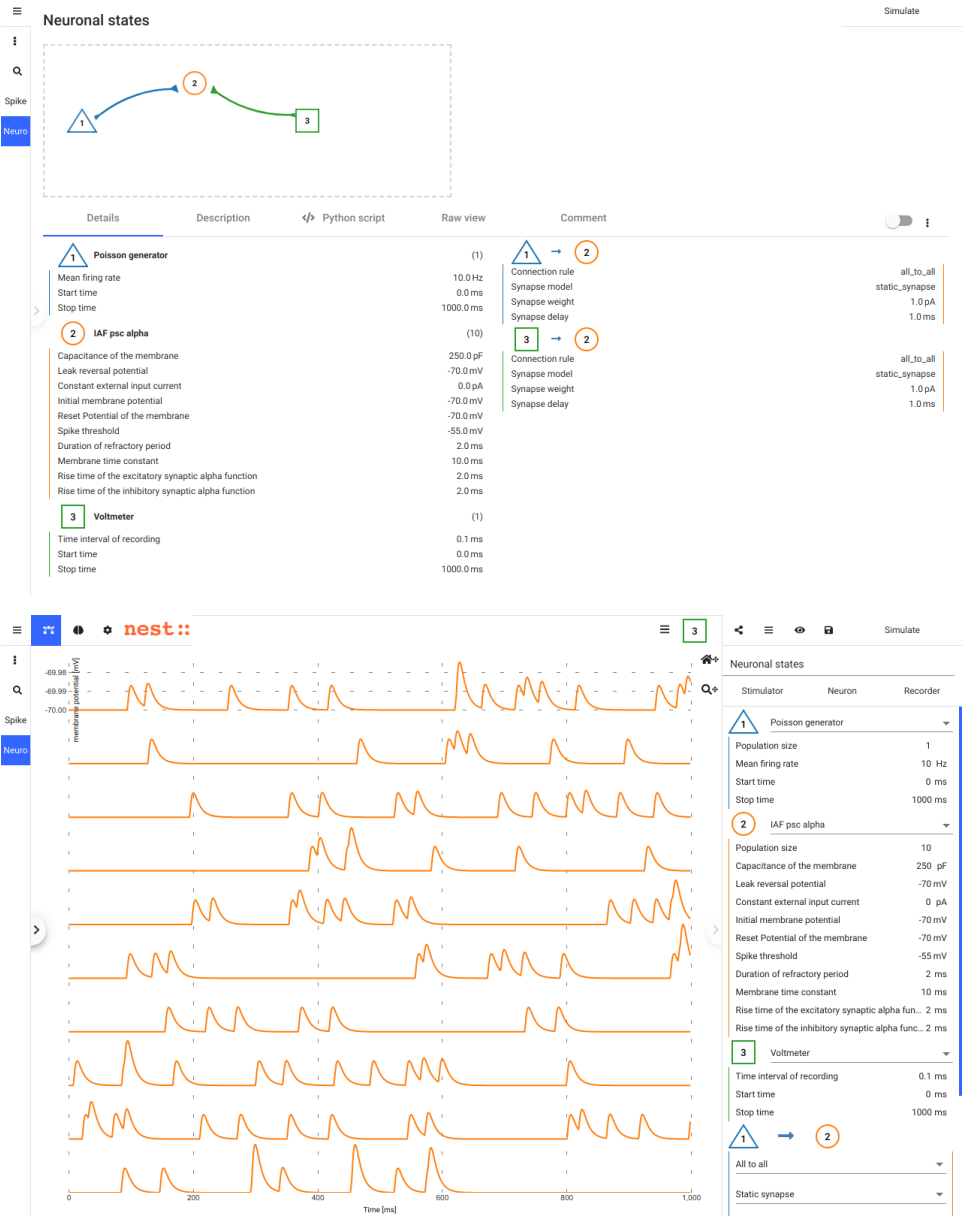


Introduce Plotly.js for interactive activity graph.

02 Oct 19	v2.0.7
30 Sep 19	v2.0.6
25 Sep 19	v2.0.5
25 Sep 19	v2.0.4
23 Sep 19	v2.0.3
16 Sep 19	v2.0.2
15 Sep 19	v2.0.1
13 Sep 19	v2.0.0



v1.x



Membrane potential [mV]

Time [ms]

0

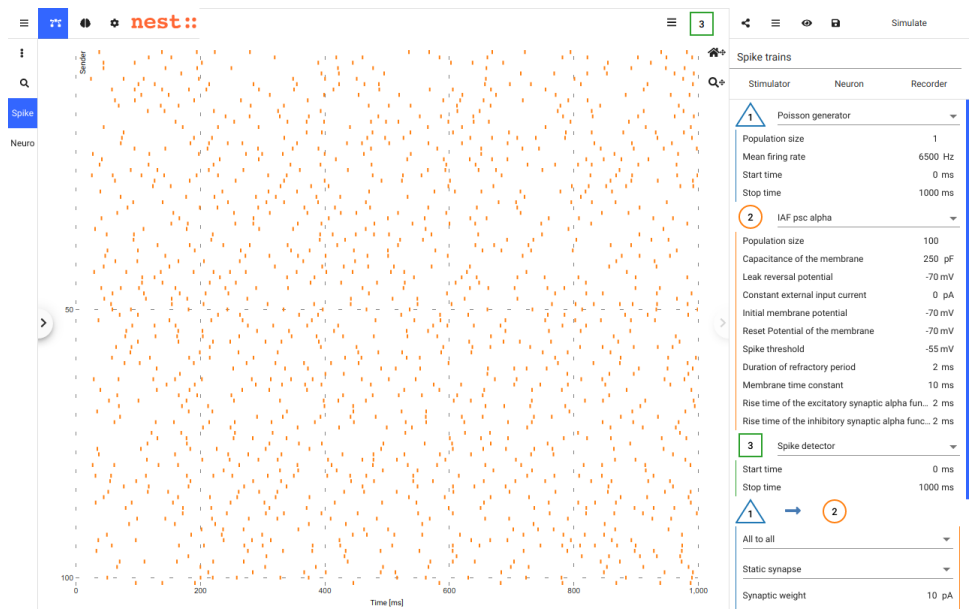
200

400

600

800

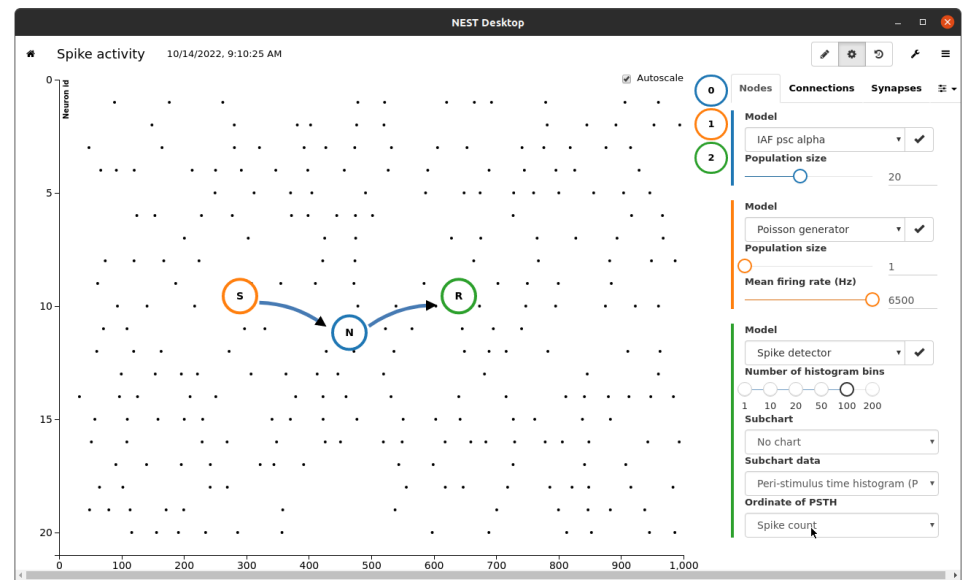
1,000

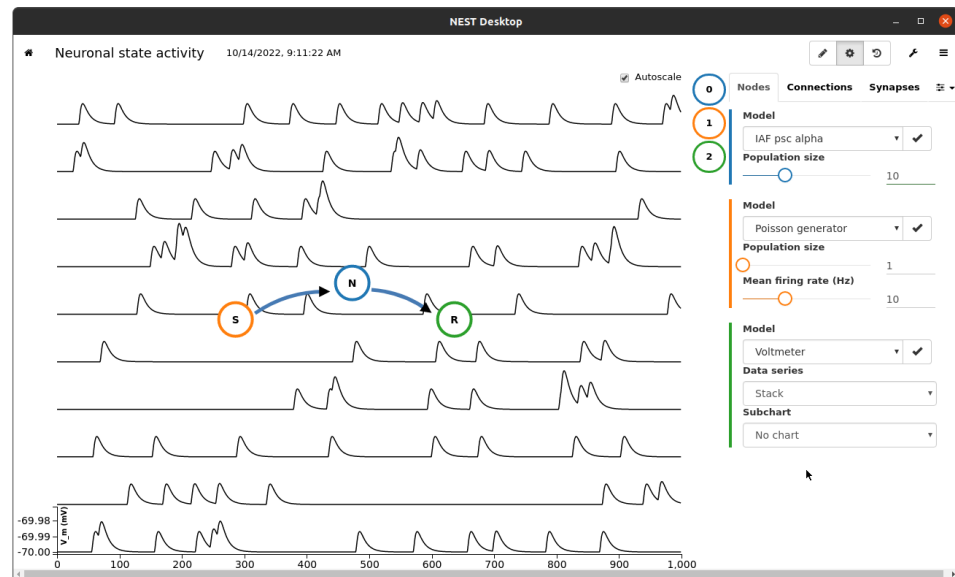


NEST Desktop runs in nginx and requires the backend [NEST Server](#).

23 Jul 19	v1.5
31 May 19	v1.4
19 Apr 19	v1.3
18 Mar 19	v1.2
18 Dec 18	v1.0

v0.x





NEST Desktop runs in Electron and requires the backend [NEST Server Simulation](#). The app uses Angular as web framework and D3.js for network and activity graphs.

22 Apr 18	v0.15.3
17 Apr 18	v0.15.1
28 Feb 18	v0.15.0
28 Feb 18	v0.14.0
07 Feb 18	v0.13.0
29 Jan 18	v0.12.0
22 Nov 17	v0.11.0
06 Oct 17	v0.10.0
20 Jun 17	v0.9.3
20 Jun 17	v0.9.2
29 Apr 17	v0.9.1
28 Apr 17	v0.9.0
18 Apr 17	v0.8.2
12 Apr 17	v0.8.1
10 Apr 17	v0.8.0
23 Mar 17	v0.7.2
15 Mar 17	v0.7.1
15 Mar 17	v0.7.0
04 Mar 17	v0.6.3
28 Feb 17	v0.6.2
27 Feb 17	v0.6.1
24 Feb 17	v0.6.0
09 Feb 17	v0.5.5
09 Feb 17	v0.5.4
07 Feb 17	v0.5.3
06 Feb 17	v0.5.2
23 Jan 17	v0.5.1
20 Jan 17	v0.5.0
09 Jan 17	v0.4.0
09 Jan 17	v0.3.12

continues on next page

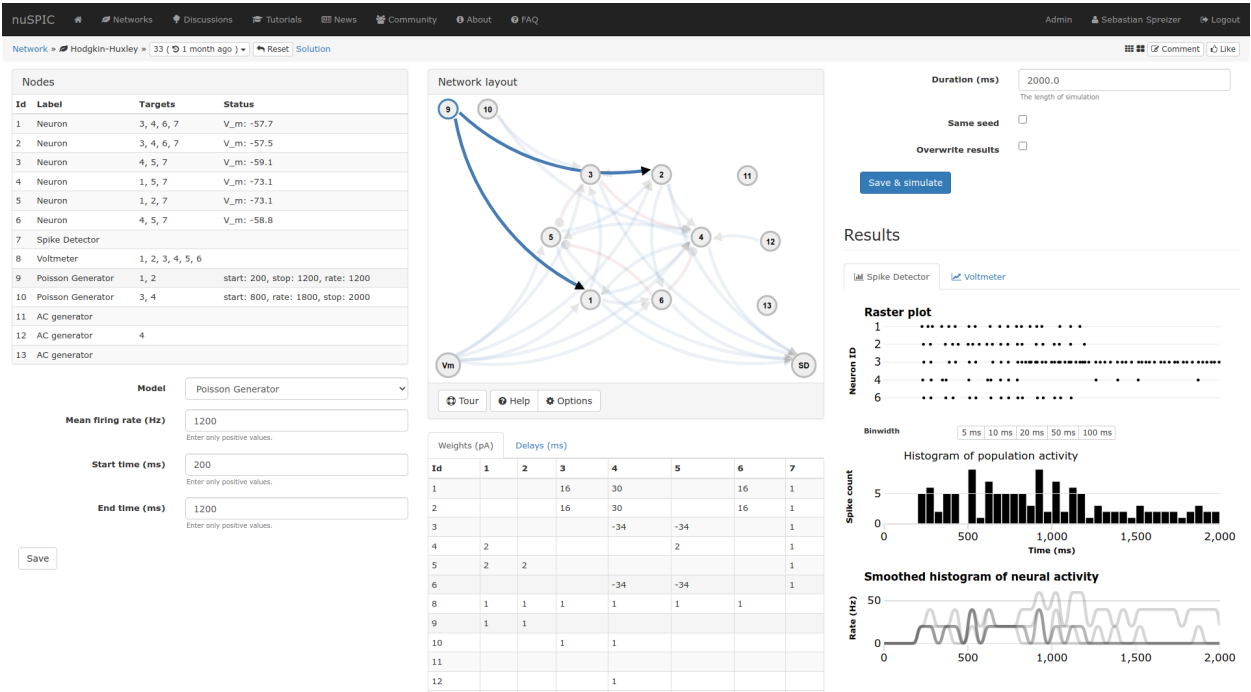
Table 1 – continued from previous page

05 Jan 17	v0.3.11
04 Jan 17	v0.3.10
04 Jan 17	v0.3.9
03 Jan 17	v0.3.8
29 Dec 16	v0.3.7
21 Dec 16	v0.3.6
21 Dec 16	v0.3.5
21 Dec 16	v0.3.4
20 Dec 16	v0.3.3
19 Dec 16	v0.3.1
19 Dec 16	v0.3.0
14 Dec 16	v0.2.1
14 Dec 16	v0.2.0
08 Dec 16	v0.1.0

3.1.3 Predecessor software

nuSPIC

Before NEST Desktop, there existed a software called nuSPIC, which targeted a similar goal as NEST Desktop:  
Before NEST Desktop, there existed a software called nuSPIC, which targeted a similar goal as NEST Desktop:



nuSPIC is also a web interface working with NEST Simulator. However, the development is inactive and the project is outdated.

See also:

- Online prototype [nuSPIC](#)

- Vlachos, I., Zaytsev, Y. V., Spreizer, S., Aertsen, A. and Kumar, A. (2013). Neural System Prediction and Identification Challenge Front. Neuroinform. 7:43. doi:[10.3389/fninf.2013.00043](https://doi.org/10.3389/fninf.2013.00043)
- nuSPIC: Neural Systems Prediction and Identification Challenge: Scientists from Freiburg present an online tool to create and analyse neuronal networks. -> [BCF News](#)

### 3.1.4 Abstract

#### An educational application for neuroscience

Simulation software for spiking neuronal network models matured in the past decades regarding performance and flexibility. Nevertheless, the entry barrier remains high for students and early career scientists in computational neuroscience since these simulators typically require programming skills and a complex installation. Here, we describe an installation-free graphical user interface (GUI) running in the web browser, which is distinct from the simulation engine running anywhere, on the student's laptop or on a supercomputer.

This architecture provides robustness against technological changes in the software stack and simplifies the deployment process for students/autodidacts and for teachers. Our new open source tool, NEST Desktop<sup>1</sup>, comprises graphical elements for creating and configuring network models, running simulations, as well as for visualizing and analyzing the results. NEST Desktop allows students to explore important concepts in computational neuroscience without the need to learn a simulator control language before.

Our experiences so far highlight that NEST Desktop helps advancing both quality and intensity of teaching in computational neuroscience in regular university courses. We view the availability of the tool on public resources like the European ICT infrastructure for neuroscience EBRAINS as a contribution to equal opportunities<sup>2</sup>.

A paper for NEST Desktop is available on [eNeuro](#).

#### References

### 3.1.5 Funding

This project has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under Specific Grant Agreement No. 785907 (Human Brain Project SGA2) and No. 945539 (Human Brain Project SGA3). This project was funded by the Helmholtz Association Initiative and Networking Fund under project number SO-092 (Advanced Computing Architectures, ACA). This work was supported by the DFG Excellence Cluster BrainLinks-BrainTools (grant EXC 1086).

### 3.1.6 Citation

In order to cite NEST Desktop in general, please use the DOI [10.5281/zenodo.5037050](https://doi.org/10.5281/zenodo.5037050) for all versions (always redirecting to the latest version). If you like to refer to a single version, you can find these also on Zenodo, e.g. [10.5281/zenodo.5037051](https://doi.org/10.5281/zenodo.5037051) for Version 3.0. You can use the reference to the paper for NEST Desktop (DOI: [10.1523/ENEURO.0274-21.2021](https://doi.org/10.1523/ENEURO.0274-21.2021)) mentioned above as well, if that is more appropriate in the context of your reference.

You will also find the exports for the citation managers on Zenodo and eNeuro.

<sup>1</sup> <https://github.com/nest-desktop/nest-desktop>

<sup>2</sup> <https://ebrains.eu/service/nest-desktop>

## 3.2 Troubleshootings

Having trouble getting something working? Got a question that the rest of our docs can't answer? Maybe we can help with some answers to commonly asked questions and troublesome spots.

### 3.2.1 Error messages

#### Server not found

NEST Desktop cannot find the NEST Simulator. It has two possible reasons:

- NEST Desktop has a wrong URL under which it tries to contact the server. ( the [FAQ for NEST Simulator](#).)
- NEST Server is not running. Try to (re-)start NEST Server.
- Use simulation service (e.g. on EBRAINS): The user authorization to the backend might be not granted.

---

**Hint:** Check NEST Server is running (if the URL is `localhost:52425`):

- in URL of Browser: `http://localhost:52425`
  - in Terminal: `curl http://localhost:52425`
  - in Python: `import requests; requests.get('http://localhost:52425')`
- 

#### Internal server error

It says that the back end (i.e. `nest-server`) ended with an internal error. In this case, you have to review the log of the back end.

#### NEST error

NEST Simulator produces a value error, e.g. `The value must be strictly negative..` Please have a look at the official [NEST documentation](#) to obtain the correct syntax for the commands.

### 3.2.2 Frequently Asked Questions

#### NEST Simulator

##### How can I change the URL of the NEST Simulator?

On the settings page you can find (and change) the URL of the NEST Simulator.

##### How can I check NEST Simulator?

On the settings page you can click on a CHECK button. If a chip with NEST version appears, this indicates that the selected NEST Simulator is working.

## Project

### How can I create a new project?

There are two ways: You find a button in the project toolbar and an item + New Project in the projects menu .

See also:

- *Create new project in project toolbar*
- *Create new project in projects menu*

### How can I duplicate a project?

In the project menu you will find a button to clone a project.

See also:

- *Duplicate project in the project list*

### How can I rename the current project?

You find a method to rename the current project in the project menu or you can edit the name in the project bar directly.

See also:

- *Rename project in the project list*
- *Edit project name in project bar*

### How can I save projects?

You find a save icon appended in each loaded project item.

See also:

- *Save a project in the project list*

### How can I delete projects?

You find a button in the project toolbar or a menu item in the project menu to delete multiple projects.

See also:

- *Delete multiple projects in the project toolbar*
- *Delete a project in the project list*

### How can I export projects?

You find a button in the project toolbar or a menu item in the project menu to export projects to a file.

See also:

- *Export multiple projects in the project list*
- *Export a project in the project list*

### How can I import projects?

You find a button in the project toolbar or a menu item in the project menu to import multiple projects from various sources.

See also:

- *Import project in the project toolbar*
- *Import project in the project list*

### Network

#### Where can I find the network controller?

You will find the network controller by clicking on the network icon () in the right controller. Models, nodes and connections are stacked as card panels in the network controller.

See also:

- [Network controller](#)

#### How can I empty a network?

In the network graph you will find top right a trash button that empties the network.

See also:

- [Network graph](#)

#### How can I create nodes?

In the network graph you can click with the right mouse button, then a selector panel appears to select an element type of the new node.

See also:

- [Create nodes in the usage guide](#)

#### How can I connect nodes?

In the network graph you can click on the connector of a source node, then move the mouse towards the target node and click on the target node.

See also:

- [Connect nodes in the usage guide](#)

#### How can I connect a node with multiple nodes?

Hold down the ALT key when clicking on the target nodes.

#### How can I (un)select a node / a connection?

When a node or connection is selected you can press ESC to unselect it or in the network graph you can click on another node or connection to select it (and to remove the selection of the former one).

Click on the background area of the network graph or on the selected entry in the network controller to unselect a node or connection. An other method to (un)select is to click on the node label or the connection toolbar in the network controller on the right side again.

#### How can I colorize nodes?

You will find the method to color in the context menu of the node by clicking with the right mouse button on the node shape in the network graph or the node toolbar in the controller.

#### How can I change the color cycle of nodes?

In the network settings you will find the way to change the color cycle.

#### How can I delete a node / connection?

You will find this method in the context menu of the node or connection by clicking with the right mouse button on the element graph in the network graph or on the colored toolbar in the network controller.

#### How can I change the node model?

You can click on model name twice and it opens a dropdown displaying models.

See also:

- [Change node model in network controller](#)



**How can I modify parameters?**

You will find a list of parameters in the network controller. If they are not visible, click on the model selection to check the visibility of the parameters.

See also:

- *Modify parameters in the controller*

**How can I reset all parameter values?**

In the context menu of a node or connection you will find the method to reset all parameters of the corresponding node or connection.

**How can I reset a parameter value?**

In the context menu of a parameter (by clicking the right button on a parameter) you can find the method to reset a parameter. It also shows the default value of the parameter.

**How can I set a connection to “inhibitory”?**

You can assign a negative value to the weights in the connection controller.

**How can I get the distribution for parameters?**

You are able to activate the distribution of the parameters in the export mode.

**How can I get a spatial node?**

In the context menu of the node, you can (un)set the spatial mode of the node.

**How can I generate grid/free positions?**

When the node is spatial, a position item will replace the population item. Click on the position item to open a popup of the position specifications. Modifying a value will generate positions, at the end of the panel you will find a button to generate positions.

**How can I generate an array?**

In the context menu of the array parameters (e.g. the spike times of a spike generator) you will find a method to generate an array.

**Simulation****How can I start a simulation?**

Click on the SIMULATE button in top right of the page to start the simulation.

**How can I stop a simulation?**

Unfortunately, the option to stop/cancel a simulation is not implemented in NEST Simulator and therefore not supported by NEST Desktop.

**How can I activate “simulation after change”?**

In the menu of the SIMULATE button (by clicking on its caret at the right side) you will find an option to activate simulation after change.

**How can I activate “simulation after load”?**

In the menu of the SIMULATE button (by clicking on its caret at the right side) you will find an option to activate simulation after load.

**How can I activate “simulation after checkout”?**

When you go to another network version of the history, it automatically starts the simulation. In the menu of the SIMULATE button (by clicking on its caret at the right side) you will find an option to activate simulation after checkout.

**Where can I find the kernel controller of the simulation?**

The kernel controller can be shown by clicking on the engine icon on the right side.

See also:

- [Kernel controller](#)

### Where can I set the simulation time?

You will find the simulation time in the kernel controller.

#### See also:

- [Change simulation time in the kernel controller](#)

### Where can I change the time resolution of the kernel?

You will find the time resolution for the NEST Simulator in the kernel controller.

**Warning:** Please verify that the recording interval is equal to or larger than the time resolution of the simulation!

#### See also:

- [Change time resolution in the kernel controller](#)

### Where can I change the seed?

You can find the seed value in the kernel controller.

#### See also:

- [Change seed in the kernel controller](#)

### How can I activate the seed randomization?

You can find an option to activate the seed randomization in the kernel controller.

#### See also:

- [Activate seed randomization in the kernel controller](#)

### How can I find the Python script code of the simulation?

On the right side you can find a code symbol <\> opening the code editor.

#### See also:

- [Code editor](#)

## Activity

### How can I download the activity data of a single recorder?

In the context menu of the recorder you will find a menu option to download events of this recorder.

### How can I download activity data of all recorders?

In the projects dialog to download projects you can find options to download network activities of projects.

## Activity chart graph

### How can I drag/zoom the chart?

You will find those modes in the mode bar (top) in the activity graph. For dragging or zooming, simply click on the chart.

### How can I reset the view to the default one?

Click on the house icon in the mode bar (top) to reset the view to the default one.

### How can I download a plot of the chart?

Click on the photo icon (top) to download the plot of the chart. You can choose which format will be used.

**Where can I find the activity controller?**

You can find the activity controller by clicking on the *chart* icon on the right side.

**How can I modify the bin size of the PSTH?**

In the chart controller you will find a tick slider to modify the bin size.

**How can I change the labeling of axes or the title?**

Click on the label of the axe or the title to change it.

**How can I hide/show dots/lines?**

Click on the legend to alter the visibility of the dots/lines.

**Activity animation graph****How can I rotate the camera?**

Click and hold the (left) mouse button on the animation area and then move it around to rotate the camera.

**Where can I find the activity controller?**

You can find the activity controller by clicking on the *axes* icon on the right side.

**How can I stop an animation?**

Go to the animation controller. You will find a pause icon to stop the animation.

**How can I increase/decrease the animation speed?**

In the animation controller you will find a forward or backward button to alter the animation speed.

**How can I change the colorscale of dots?**

In the animation controller you will find a colormap of the current colorscale. A little below you will find an options to select the colorscale.

**How can I change the size of dots?**

In the animation controller you can find a slider to adjust the dot size.

**I cannot find the “trailing” effect box for dots in the activity controller?**

It only works with the animation of the spikes.

**Model****What is the terminology of this model?**

This model includes neuron, synapse and device (stimulus / recorder) models.

**How can I read the documentation of a model?**

In the context menu of a node you will find a documentation of these models, but also in the model section.

**3.2.3 Services****EBRAINS****Why cannot I find NEST Simulator?**

Sometimes the issue is resolved when you check NEST Simulator.

If not, the problem lies in the expired token for the user authentication of the *EBRAINS* platform. This happens in a new session when you re-visit the page after the browser is closed.

A simple solution is to reload the page (CTRL + SHIFT + R) so that you can re-login.

**Warning:** Please avoid to accidentally delete site data you want to keep! Ensure to export your projects if you want to keep them - the save button stores them only in the browser storage!

### 3.3 User guide

The user guide provides detailed documentation of the GUI of NEST Desktop.

#### 3.3.1 How to use NEST Desktop

##### Setup Guide

This guide provides a detailed documentation on how to install and start both instances: NEST Desktop and NEST Simulator.

---

**Note:** To enable the full functionality of NEST Desktop, you also need to install NEST Simulator on your computer. NEST Simulator provides an API Server which can forward requests to the simulation engine. In summary, you have to start NEST Server as well.

You can find the detailed information on NEST Server in [NEST Simulator user documentation](#).

---

Docker (or Docker Compose) and Apptainer provide both NEST Desktop and NEST Simulator, so you have everything you need to run NEST Desktop and NEST Simulator.

Alternatively, you can install NEST Desktop with the `conda` or `pip` command.

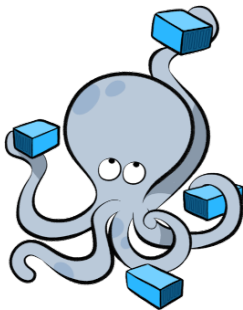
You are able to download and start the NEST Desktop application, e.g. AppImage (without NEST Simulator) or Snap (with NEST Simulator) in Linux.

If you only have NEST Desktop (i.e., NEST Simulator is not running as back-end), you can create networks but cannot run simulations within the application. In this case, you have to start NEST Server in a terminal or with Docker.

**See also:**

[NEST Server user documentation](#)

##### Docker Compose



Docker is a virtualization software packaging applications and its dependencies. Docker Compose is a tool for running multi-container applications on Docker which uses the Compose file format.

**See also:**

For further information, please see the [official page of Docker Compose](#).

## Installation

Docker Compose is available on multiple platforms. This guide demonstrates some of the ways to install it on Linux, Windows and Mac.

### Linux

Install Docker and Docker Compose in Terminal

```
apt install docker.io docker-compose
```

### Windows and macOS

Docker Compose is included in Docker Desktop for Windows and macOS. For more information, take a look at the [installation guide of Docker Desktop](#).

## Pull and start Docker containers

1. Get the configuration file for Docker Compose ([docker-compose.yml](#))

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop/main/docker-compose.yml
```

2. Start NEST Desktop and NEST Simulator in a single command:

```
docker-compose up
```

Now, the service starts the containers for NEST Desktop and NEST Simulator. You can use NEST Desktop in the web browser at <http://localhost:54286>.

**The installation is now complete!** *Now you can start constructing networks for the simulation!*

**See also:**

For more information (like running the containers without root password, etc.), please read the full documentation of [NEST Desktop Docker](#).

### Apptainer



Apptainer, former Singularity, is an application container for **Linux** systems. For more information read the full documentation of Apptainer [here](#).

### Get recipes

1. Clone a working copy from the repository and go to the folder:

```
git clone https://github.com/nest-desktop/nest-desktop-apptainer
cd nest-desktop-apptainer
```

2. Register the bash command for NEST Desktop Apptainer:

```
export PATH=$PATH:$PWD/bin/
```

---

**Note:** You will have to repeat this every time you end a terminal session. If you like to register this command permanently, please proceed according to the [full documentation](#).

---

### Build image

3. Build the Apptainer images (it will ask for sudo password):

```
nest-desktop-apptainer build
```

---

**Note:** This command (and the following ones) need to be executed inside the folder where the container files are located, i.e. the `nest-desktop-apptainer` folder.

---

### Start container

4. Start the Apptainer instances of NEST Desktop and NEST Simulator:

```
nest-desktop-apptainer start
```

Now NEST Desktop is started. You can use NEST Desktop in the web browser at <http://localhost:54286>.

**The installation is now complete!** *Now we can start constructing networks for the simulation!*

For more information read the full documentation of [NEST Desktop Apptainer](#).

**Warning:** If the apptainer (esp. NEST Simulator) is running, your system is exposed for unauthorized access!

## Conda



Anaconda provides packages for [NEST Desktop](#), and [NEST Simulator](#). These packages can be installed with Conda. We highly recommend installing at least version 3 of NEST. Since NEST 3, the API server (i.e., NEST Server) is already implemented.

## Install with Conda

1. Create a Conda environment called nest3 and install NEST Simulator:

```
conda create -n nest3 nest-simulator
```

2. Activate the Conda environment nest3:

```
conda activate nest3
```

3. Install the dependencies for the API Server of NEST Simulator:

```
conda install flask flask-cors RestrictedPython gunicorn
```

4. Install NEST Desktop

```
conda install nest-desktop
```

## Start with Conda

1. Start NEST Server as the back end:

The API Server for NEST Simulator is referred to as **NEST Server**.

```
nest-server start
```

NEST Server is now running at <http://localhost:52425>.

2. Start NEST Desktop (in another terminal session):

```
nest-desktop start
```

NEST Desktop is now started and available in the web browser at <http://localhost:54286>.

**The installation is now complete!** *Now you can start constructing networks for the simulation!*

### See also:

For more information read the full documentation of the command API [here](#).

## Python



PyPI contains packages of NEST Desktop and NEST Simulator. We recommend to install both packages.

## NEST Simulator

1. Install NEST Simulator (SKIP THIS STEP IF YOU HAVE NEST 3 INSTALLED.):

Read the full installation guide of NEST Simulator [here](#).

We highly recommend installing NEST 3. With NEST 3, the API server (i.e., NEST Server) is already implemented.

2. Install the dependencies for the API Server of NEST Simulator:

```
pip install flask flask-cors RestrictedPython gunicorn
```

3. Start NEST Server as the back end:

The API Server for NEST Simulator is referred to as **NEST Server**.

```
nest-server start
```

NEST Server is now running at <http://localhost:52425>. You can find the detailed information on NEST Server [here](#).



## NEST Desktop

### 1. Install NEST Desktop

NEST Desktop is available on PyPI and can be installed with the `pip` command:

```
pip3 install nest-desktop [--user] [--upgrade]
```

For more information, please read the complete installing guide [here](#).

### 2. Start NEST Desktop (in another terminal session):

```
nest-desktop start
```

Now NEST Desktop is started. You can use NEST Desktop in the web browser at <http://localhost:54286>.

**The installation is now complete!** *Now you can start constructing networks for the simulation!*

#### See also:

For more information read the full documentation of the command API [here](#).

## AppImage

You can download an AppImage from the [releases page](#).

Click on the `.AppImage` file to open NEST Desktop.

---

**Note:** Start the API Server of NEST Simulator manually before you open NEST Desktop.

---

## Snap



You can download NEST Desktop via Snap.

```
snap install nest-desktop
```

**The installation is now complete!** *Now you can start constructing networks for the simulation!*

### Basic Usage Guide

This is a basic usage guide for the Graphical User Interface (GUI) of NEST Desktop.

---

**Note:** If you want to see a quick start guide for in NEST Desktop, we have prepared a [video](#) showing the steps how to *Construct networks* and *Explore activity*.

---

Once you start NEST Desktop, you can see the start page containing an image of a laptop with the NEST logo on its screen. At the bottom it shows a short description of NEST Desktop (left) and some useful links and the current version (right).



# NEST Desktop

An educational GUI for neuroscience

[+ START A NEW PROJECT](#)

[📁 LOAD A PROJECT](#)


NEST Desktop is a web-based GUI application for NEST Simulator, an advanced simulation tool for computational neuroscience. The application enables the rapid construction, parametrization, and instrumentation of neuronal network models.

The primary objective is to provide an accessible classroom tool that allows users to rapidly explore neuroscience concepts without the need to learn a simulator control language at the same time.

Documentation	<a href="https://nest-desktop.readthedocs.io">https://nest-desktop.readthedocs.io</a>
Source code	<a href="https://github.com/nest-desktop/nest-desktop">https://github.com/nest-desktop/nest-desktop</a>
License	MIT License
Current version	3.1.0
Contact	✉ Sebastian Spreizer

 EBRAINS

 Co-funded by the European Union

 Human Brain Project



---

**Note:** You can reload the page if NEST Desktop has somehow crashed.

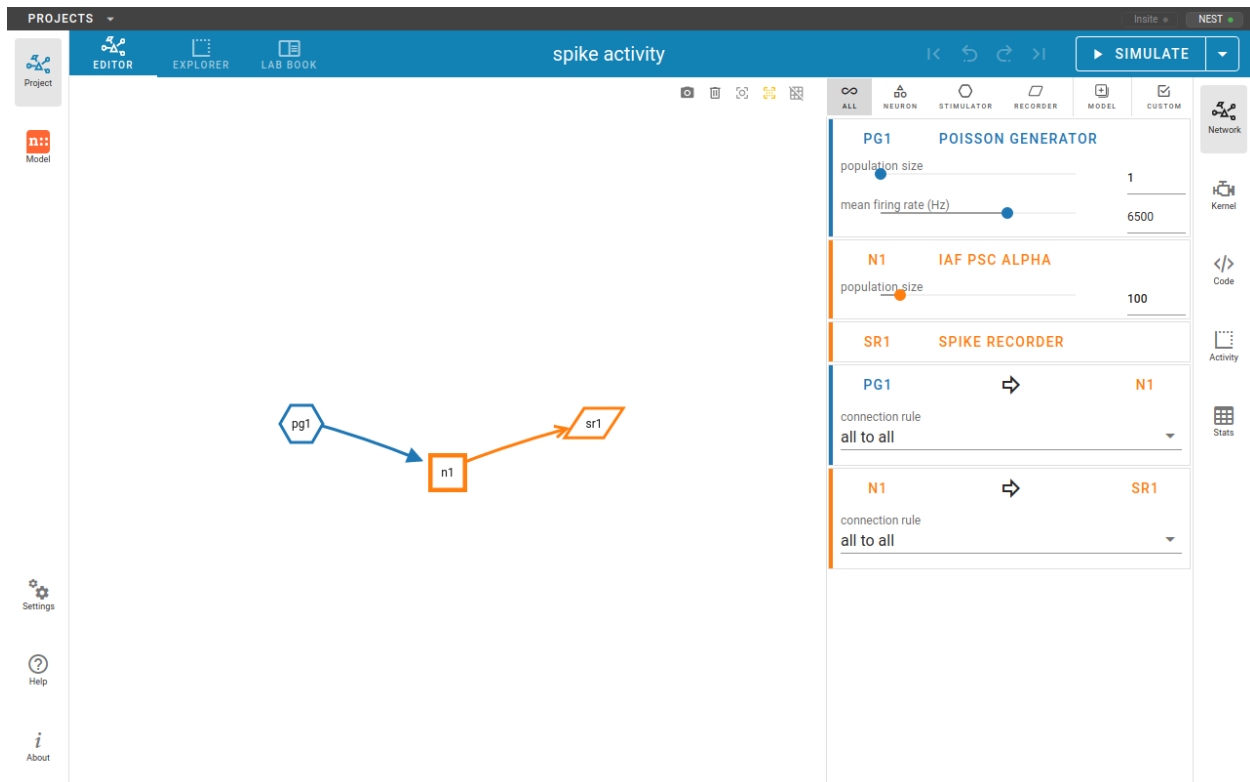
---

## First steps

The video shows the first steps to construct a network and explore its activity.

## Construct networks

If you want to construct a network, you will have to open the network editor. The network editor shows the network graph composed of nodes (shapes) and connections (lines).



## Create nodes

In order to create a new node, you can click with the right mouse button in the network editor and a *pie* panel with three letters appears to select an element type. A node is divided into three element types: stimulus (S), recording (R) device and neuron (N). Then it creates a node of the selected element type.

### Connect nodes

Forming a network of nodes is defined by making connections between and within nodes. In order to connect nodes, you can click on a connector of a node, then move the mouse towards another node and finally click on a target node. It creates a connection between source and target nodes.

---

**Hint:** By pressing the hotkey ALT and clicking a node at the same time, you enable the connecting mode or continue connecting other nodes.

---

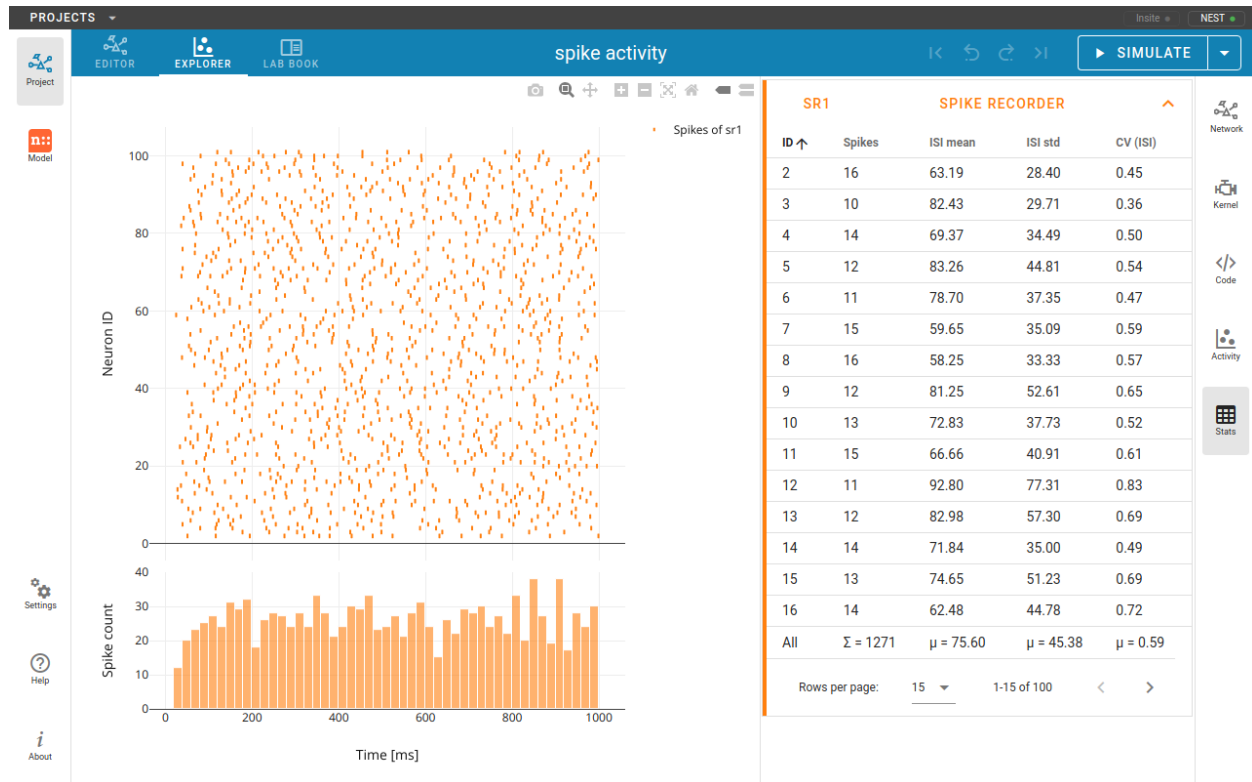
### Select model and parameters

You are able to select the model of a node in the network controller. Then it shows a list of parameters which you might want to work on. Finally, you are able to change the values of visible parameters.

### Simulate networks

You can click on the SIMULATE button to start the simulation of your network. In the code editor you can inspect the generated script code.

## Explore activity

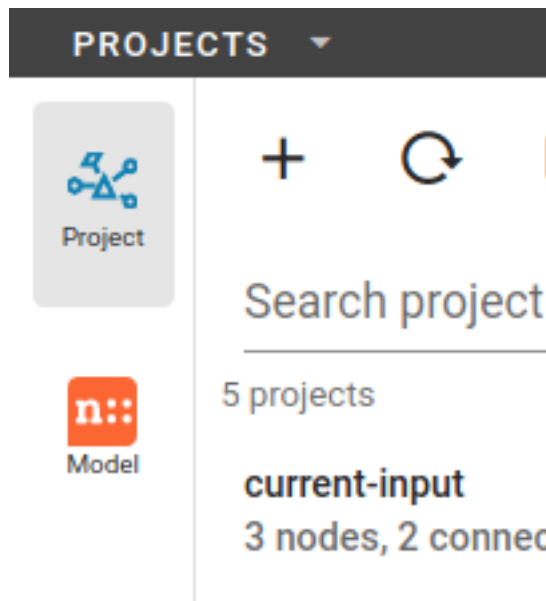


The network activity is composed of neuronal properties (positions and ids of neurons) and recorded events from recording devices. Events can be subdivided in two groups: spike events and analog signals. Spike events contain times and ids of the senders emitting events to the recording devices which can be considered as collectors (spike recorder). Analog signals contain continuous quantities from the recording devices aka samplers (voltmeter or multimeter) which query their targets at given time intervals. Network activity can be explored in [Activity chart graph](#) ( or ) [Activity animation graph](#) (), or [Activity statistics](#) ().

### 3.3.2 Advance guide

#### Project view

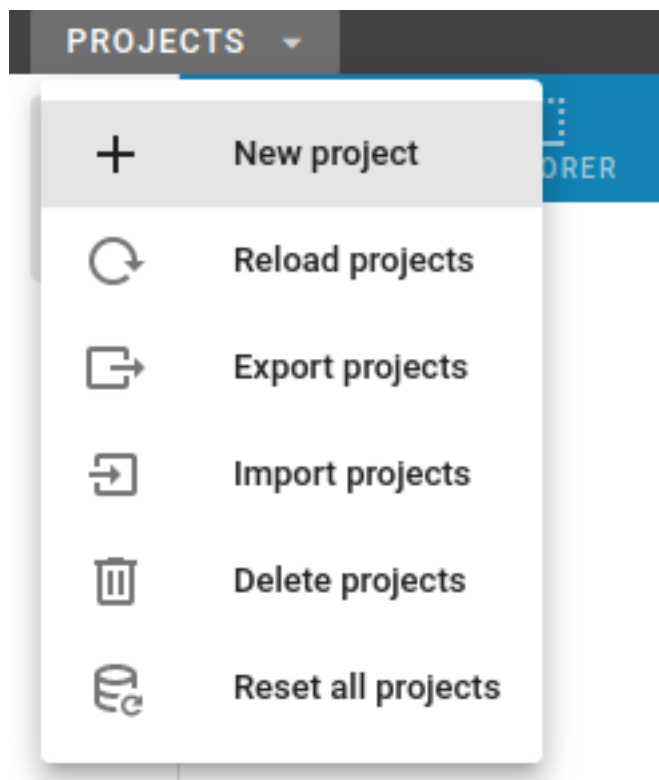
NEST Desktop has a project management helping you to organize your networks and network activity.



It contains a:ref:project-view-projects-menu in the system bar to manage multiple projects, a *Project navigation sidebar*, a *Project bar* and content for *Project subpages*.

If you want to explore the network activity of the project, you have to start the simulation before ( *Simulate networks*).

### Projects menu



The projects menu will be displayed when the user clicks the PROJECTS entry in the system bar (top black bar). The opened project menu shows the same options which are displayed as buttons in the toolbar.

In the menu you find options to create a new project () as well as to reload (), export (), import (), delete () or reset () multiple projects.

## Project dialog

It is possible to import projects from different sources: You can choose between drive (local storage), GitHub and URL (other one than GitHub URLs).

### Import projects

Select source and file

Source

drive
GitHub
URL

Path

simple\_neuron\_models

File

cond\_time\_constants.json

Import:

	Created at	Version	Valid	Selected
time constants in cond	7/20/2021, 11:22:05 AM	3.2	✓	<input type="checkbox"/>

CANCEL

IMPORT

Also you are able to export multiple projects. The selection checkbox appears when the project is loaded (check the validate box by clicking it).

### Export

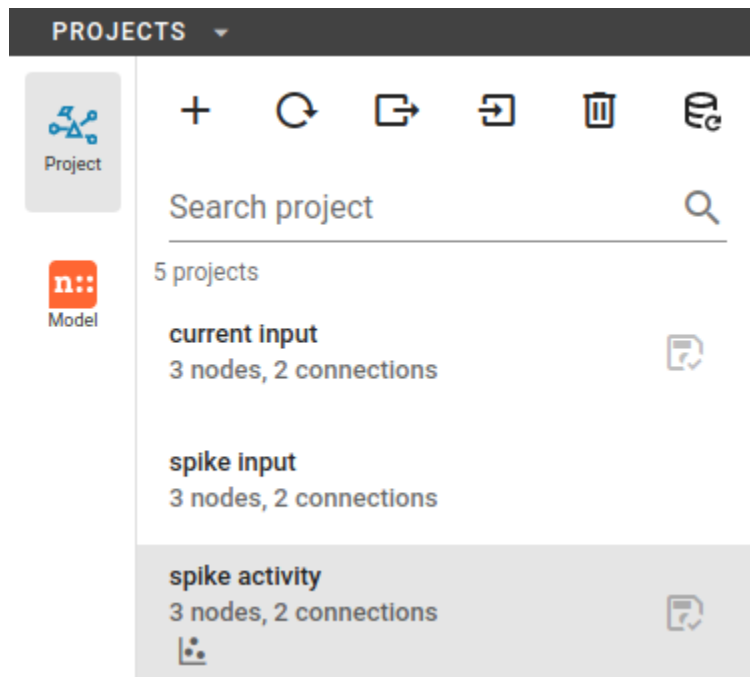
Select projects to export

Project name	Created at	Updated at	Validate	Selected	Activities
current input	11/22/2022, 4:50:52 PM	11/22/2022, 4:50:52 PM	✓	<input type="checkbox"/>	<input type="checkbox"/>
spike input	11/22/2022, 4:50:52 PM	11/22/2022, 4:50:52 PM	<input type="checkbox"/>		
spike activity	11/22/2022, 4:50:52 PM	11/22/2022, 4:50:52 PM	✓	<input type="checkbox"/>	<input type="checkbox"/>
spatial neurons	11/22/2022, 4:50:52 PM	11/22/2022, 4:50:52 PM	<input type="checkbox"/>		
spatial spike activity	11/22/2022, 4:50:52 PM	11/22/2022, 4:50:52 PM	<input type="checkbox"/>		

CANCEL

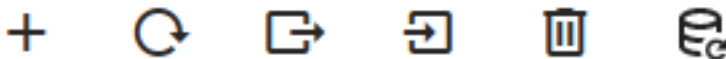
EXPORT

## Project navigation sidebar



In the navigation sidebar you find a *Project toolbar* and then a *Project list*.

## Project toolbar



At the top of the navigation sidebar, you see a toolbar containing buttons to create a new project () as well as to reload (), export (), import (), delete () or reset () multiple projects.

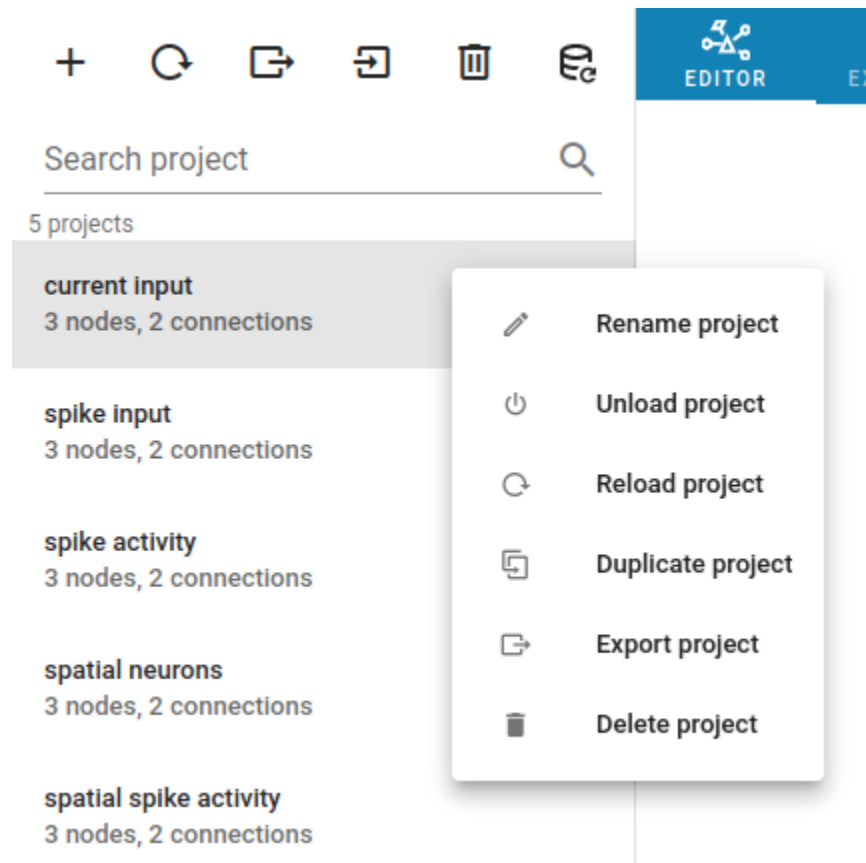
Clicking on the buttons to export, import or delete projects opens a dialog showing a list of project ( *Project dialog* ).

**Warning:** You should export projects that you want to keep: If you refresh your browser or delete the web page cookie, the project will be lost!

Creating a new project lets you construct a network from scratch ( *Construct networks* ).



## Project list



Below the buttons you find the search field and a list of the projects. Select a project to load it for the usage. Once a project is loaded, a save icon () appears on the right side. You can move the mouse on the project item, it shows three vertical dots () for a menu with options to rename (), unload (), reload (), duplicate (), export () or delete () this project.

**Warning:** Unless you click on the save button, the project is not stored in the database of the web page cookie and is lost when you reload the page!

Another important remark is that NEST Desktop stores only projects with neuronal networks in the cookie database, but all activity (i.e. simulation results) will be lost after page reload!

## Project bar



The project bar contains tabs for *Project subpages*, the project name, the *Network history* and the *Simulation button*.

**Tip:** It is useful to give project a proper name so that you can recognize your projects quickly.

## Network history

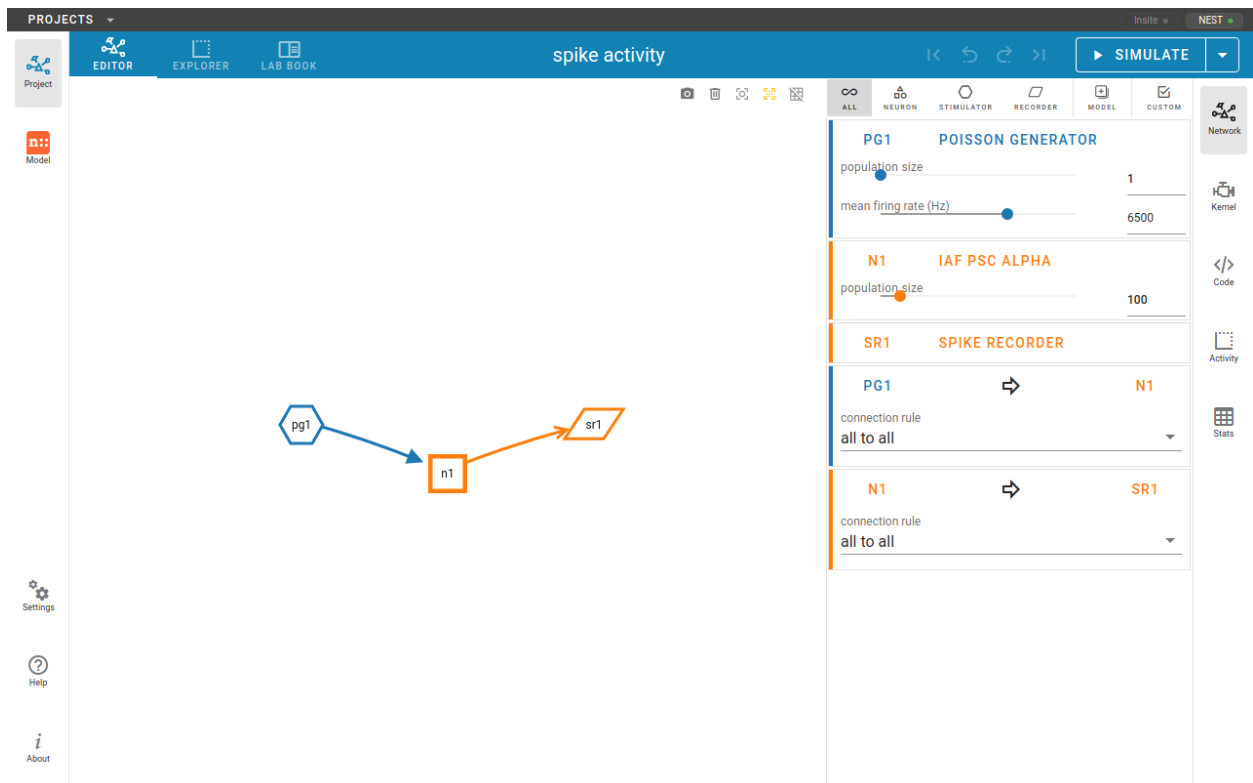
After every network change, NEST Desktop pushes a snapshot of the current network to the edit history list. With that history of the network, you can undo or redo the network changes. Loading a snapshot from this history is called *checkout network*.

## Simulation button

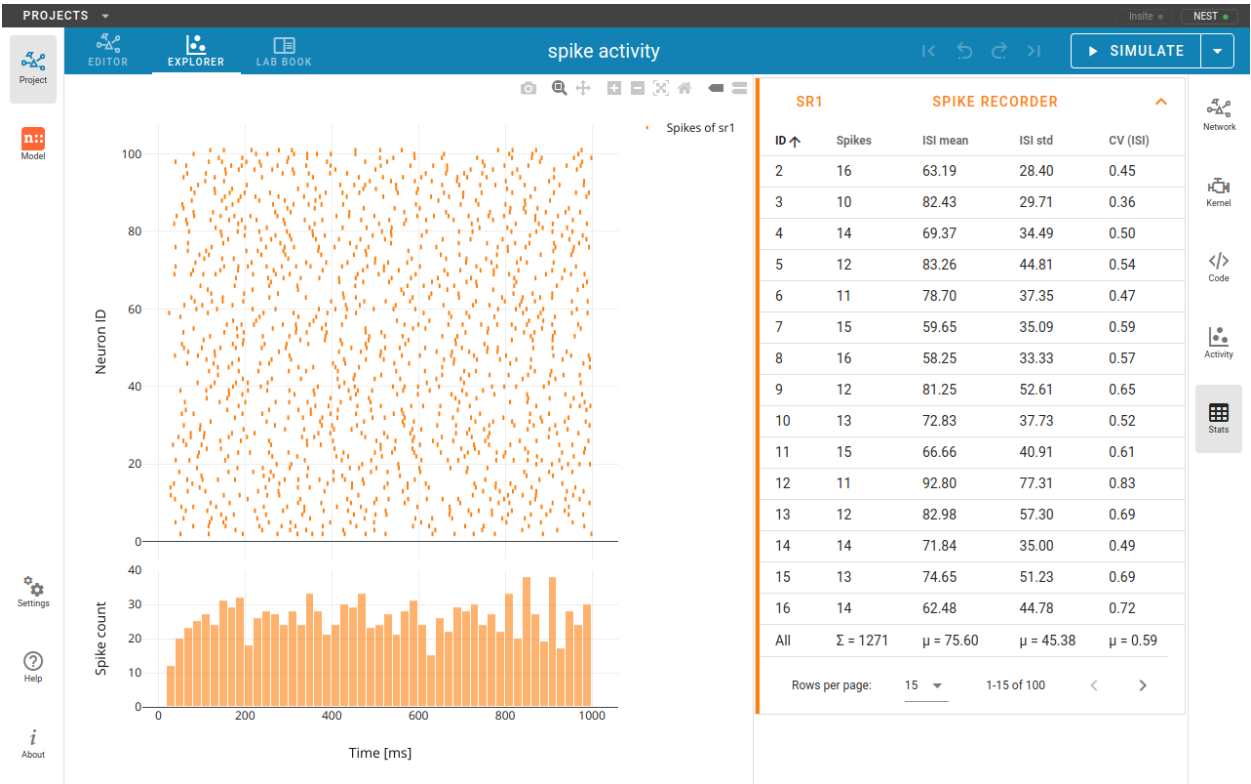
You can click on the SIMULATE button to start the simulation.

## Project subpages

## Network editor



Activity explorer



Network

Kernel

Code

Activity

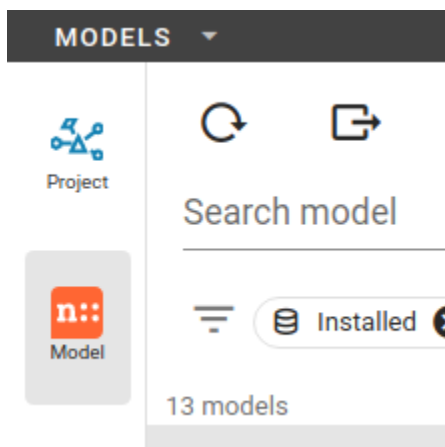
Stats

## Lab book



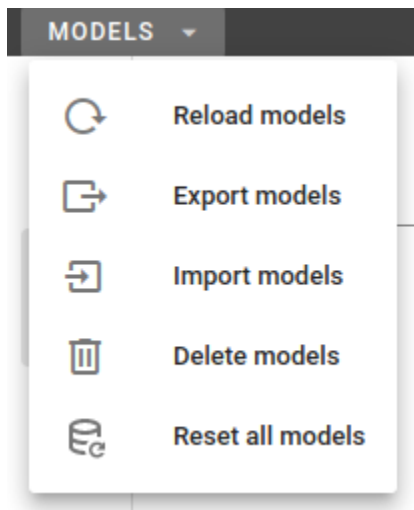
## Model view

This is the guide for the model view in NEST Desktop.



Below the icon for the project view, you can see the one of the model view, where you can *read the model description*, *explore model activities* or *edit model configurations*.

## Models menu



By clicking the right mouse button on the model icon, a menu appears where you can select actions for models.

## Model dialog

You can import models from various sources, e.g. a file you uploaded from your computer, a file from a GitHub repository or from a specified URL.

### Import models

Select a source Import from GitHub Select element type neuron Select file aeif\_cond.json

4 models found. Select models to import.

Model	Label	Version	Valid	Selected
aeif_cond_alpha	AEIF cond alpha		✓	<input checked="" type="checkbox"/>
aeif_cond_alpha_multisynapse	AEIF cond alpha multisynapse		✓	<input type="checkbox"/>
aeif_cond_beta_multisynapse	AEIF cond beta multisynapse		✓	<input type="checkbox"/>
aeif_cond_exp	AEIF cond exp		✓	<input type="checkbox"/>

CANCEL IMPORT

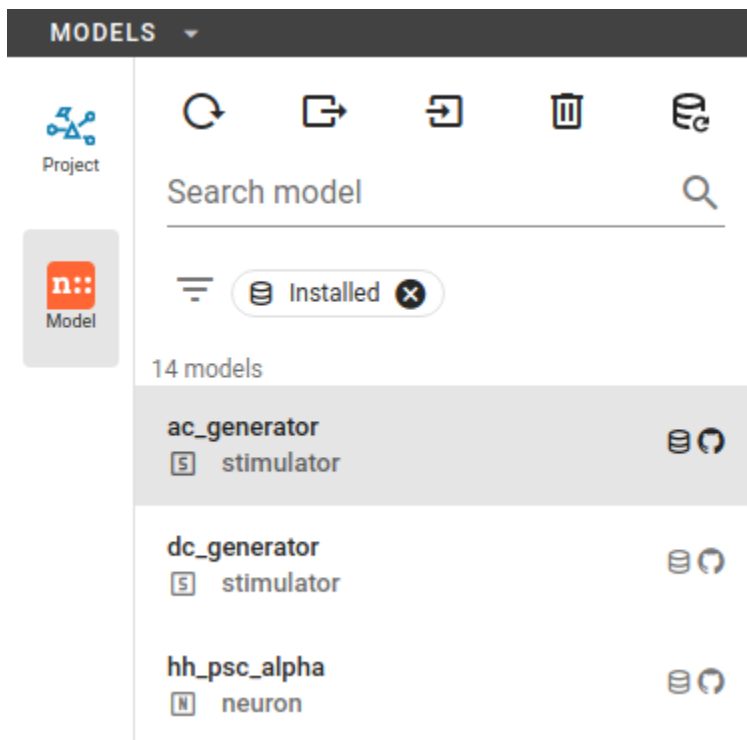
**Note:** Model files should be formatted in JSON.

---

When you select Import from GitHub, choose an element type and then a JSON file of your desired model group which includes all functions of synapse currents.

The table shows a list of models from which you can select which ones you want to import.

### Model navigation sidebar



In the navigation sidebar you find a *Model toolbar* and then *Model List*.

You can select a model to read its documentation, its activity or to edit its configuration.

### Model toolbar



At the top of the navigation sidebar, you see a toolbar containing buttons to reload (), export (), import (), delete () or reset () multiple models.

## Model List

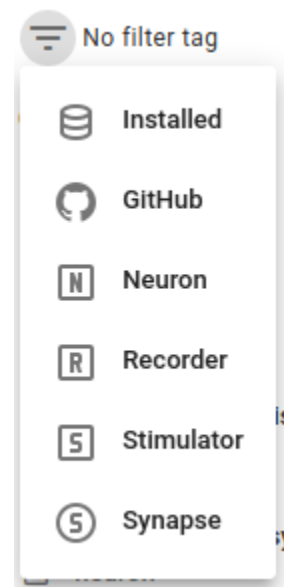
Above the model list you will find a search field and tags which you can use to filter the models in the list. Selected filter tags appear as chips under the search field.

In order to select a tag you need to click on the *filter* icon left to the search field. Multiple filter tags can be applied. Selected filter tags can be removed (click on ).

## Import models

Go to the model view and find your desired synapse model. Next, click on the icon , then select a menu item import to import it from GitHub.

## Filter models



It is possible to select filter tags to display only models with certain properties. The following filter tags are available:

### **Installed:**

Show models which are installed in NEST Desktop

### **GitHub:**

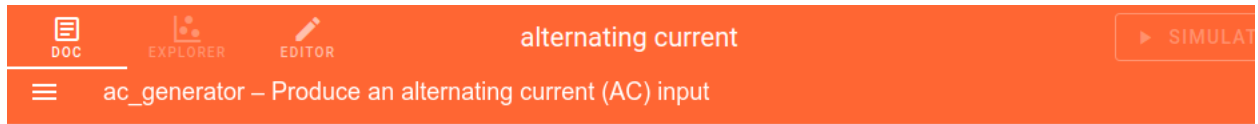
Show models which are provided in [an own GitHub repository](#)

### **Neuron/stimulator/recorder/synapse:**

Show models of the selected element type

## Model subpages

## Model documentation



### ac\_generator – Produce an alternating current (AC) input ¶

#### Description

This device produces an AC input sent by CurrentEvents. The current is given by

$$I(t) = \text{offset} + \text{amplitude} \cdot \sin(\omega t + \phi)$$

where

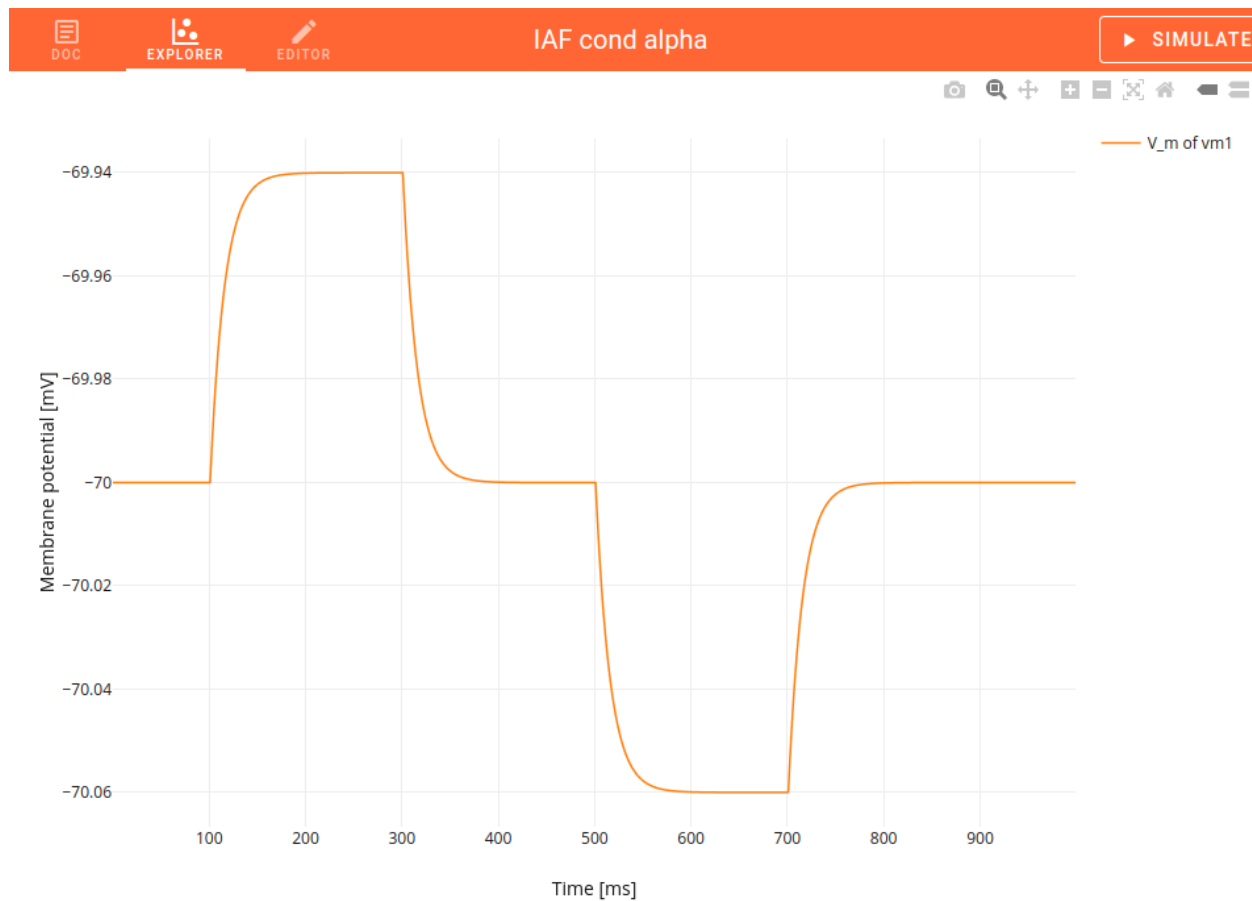
$$\omega = 2\pi \cdot \text{frequency}$$
$$\phi = \frac{\text{phase}}{180} \cdot \pi$$

All stimulation devices share the parameters `start` and `stop`, which control the stimulation period. The property `origin` is a global offset that shifts the stimulation period. All three values are set as times in ms.

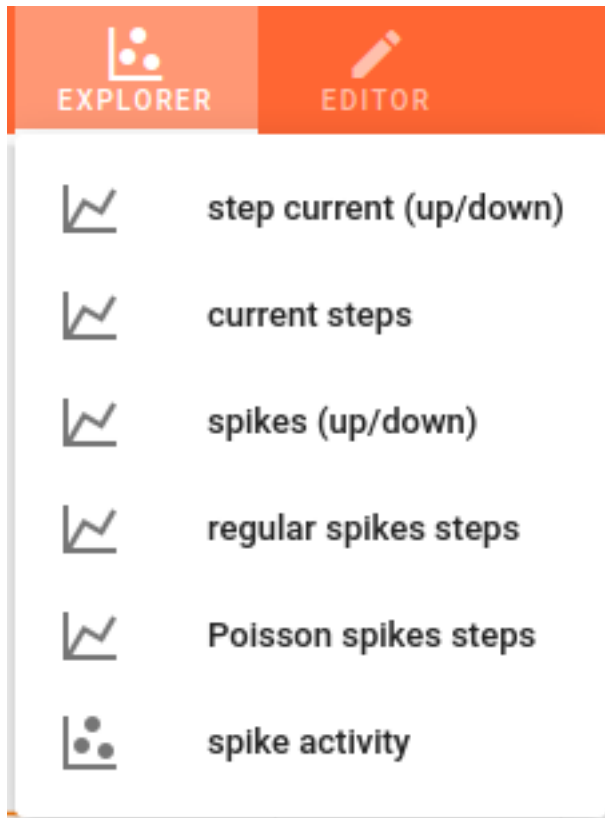
It shows the official user documentation of a selected model which also can be found on <http://nest-simulator.readthedocs.io/en/latest/models/>.



## Model explorer



You can explore the activity dynamics of **neuron** models only.



First, choose a simulation to see the neuronal response to a specific stimulus device.

Then start the simulation by clicking on the SIMULATE button.

You can use the code editor to see changes in activity.

---

**Note:** It is important to disable the Insite pipeline for the simulation (in the settings).

---

### Model editor

The model editor allows you to make changes in parameter specifications, e.g. default value, unit, label or inputs.

DOC

EXPLORER

EDITOR

alternating current

SIMULATE

label  
alternating current

Recordables: I

id	value	unit	label	input	input specifications		
amplitude	<input type="text" value="0"/>	<input type="text" value="pA"/>	Amplitude of sine current	value sli... ▼	min -1000	max 1000	step 1
frequency	<input type="text" value="0"/>	<input type="text" value="Hz"/>	frequency	value sli... ▼	min 0	max 100	step 1
offset	<input type="text" value="0"/>	<input type="text" value="pA"/>	constant amplitude offset	value sli... ▼	min 0	max 1000	step 1
phase	<input type="text" value="0"/>	<input type="text" value="deg"/>	phase of sine current	value sli... ▼	min 0	max 360	step 1
start	<input type="text" value="0"/>	<input type="text" value="ms"/>	start time	value sli... ▼	min 0	max 1000	step 1
stop	<input type="text" value="1000"/>	<input type="text" value="ms"/>	stop time	value sli... ▼	min 0	max 1000	step 1

SAVE

UPDATE MODEL FROM GITHUB

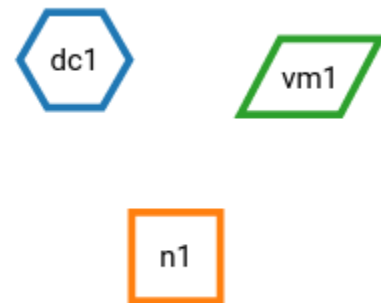
Network graph



### Node labels

Each node graph is labeled to identify the model of the node. By default, it creates a direct current generator (dc) for a stimulus and a voltmeter (vm) for a recording device. Neurons are just labeled with n. You can find the full label of the node model in the network controller.

### Node colors



Nodes and connections contain parameter configurations which are displayed in the controller panel in the side navigation. The color of nodes helps you to associate the network graph with the controller as well as the corresponding visualization of the network activity. The color of lines is defined by the source node.

### Node shapes

The specific shape defines an element type of a node:

#### Hexagon

A stimulus device alias stimulator is an instrument which only produces signals towards target nodes.

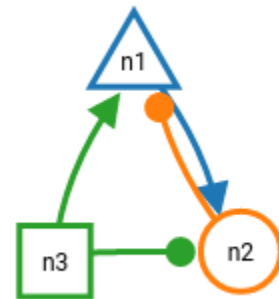
#### Parallelogram

A recording device alias recorder is also an instrument which observes states of a recordable node.

#### Others

A neuron node is the core engine of a neuronal network model which received inputs from other nodes and produces specific output using intrinsic equation. For more information about neuron shapes, see the next section.

## Neuron shapes



The shape of neurons is represented differently by the set of synaptic weights of their connections.

### Square

Neurons without connections or mixed (positive and negative) synaptic weights to neurons

### Triangle

Neurons with excitatory connections to neurons (all synapse weights are positive)

### Circle

Neurons with inhibitory connections to neurons (all synapse weights are negative)

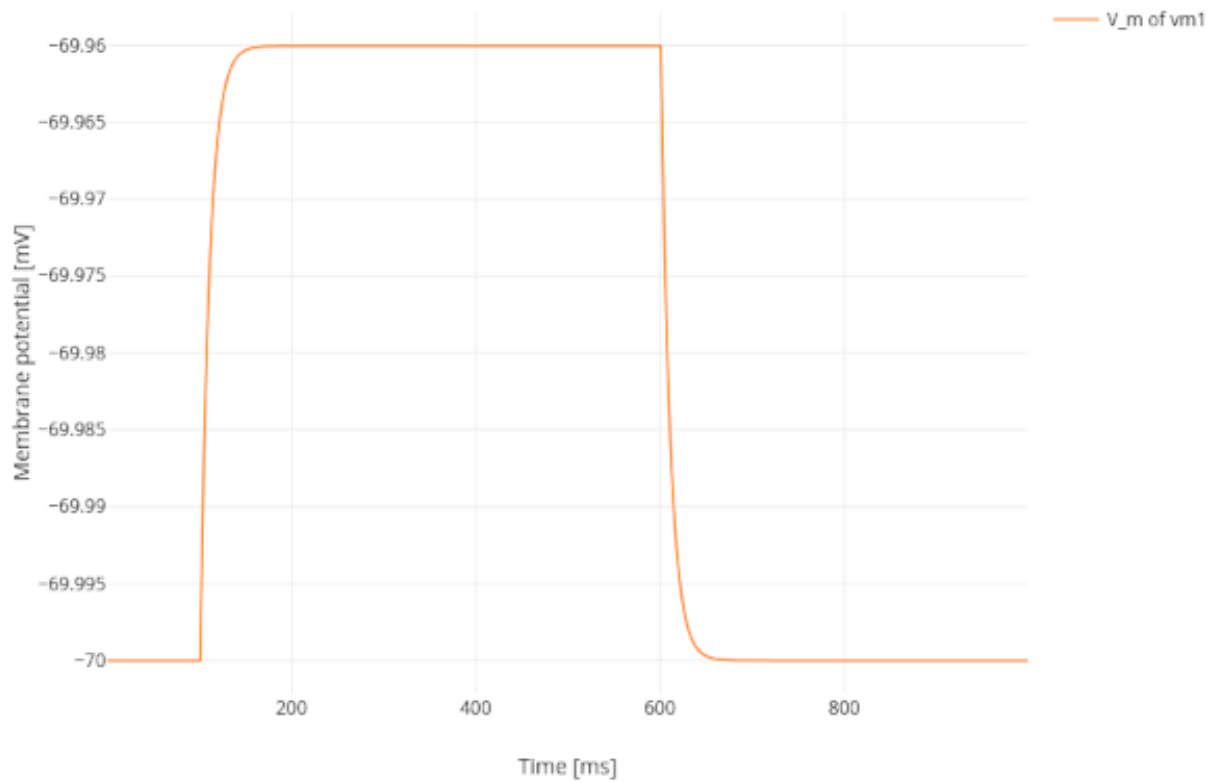
## Activity chart graph

The chart graph contains graphical panels organized in vertical stacks. Chart panels are introduced specifically to explore the network activity by mouse interaction. The simulation produces two different types of data sets: Spike events (recorded by spike recorder) contain times and sender ids whereas analog signals contain continuous quantities from the recording devices (voltmeter or multimeter).

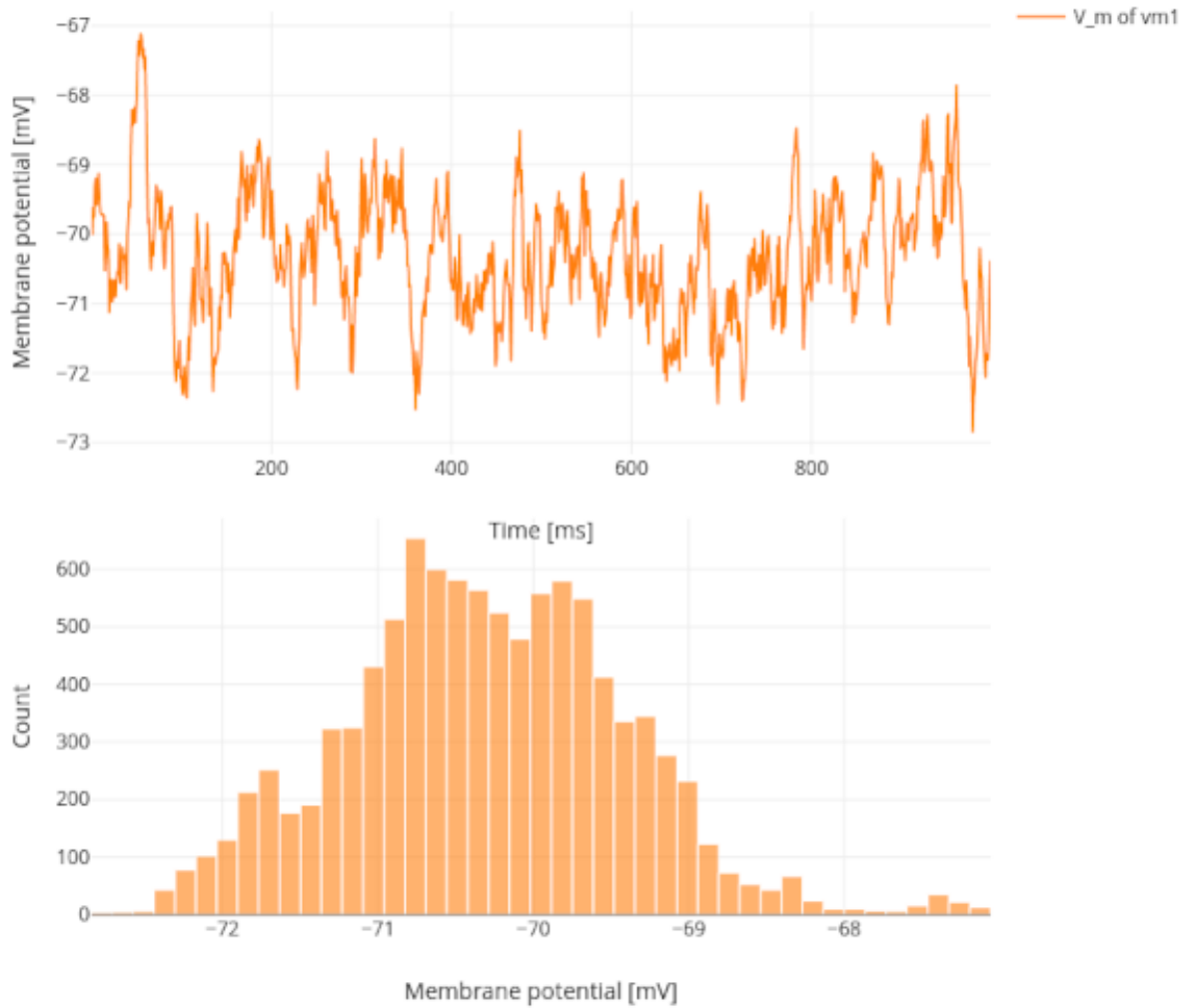
### See also:

*Use controller for activity graph*

## Analog signals

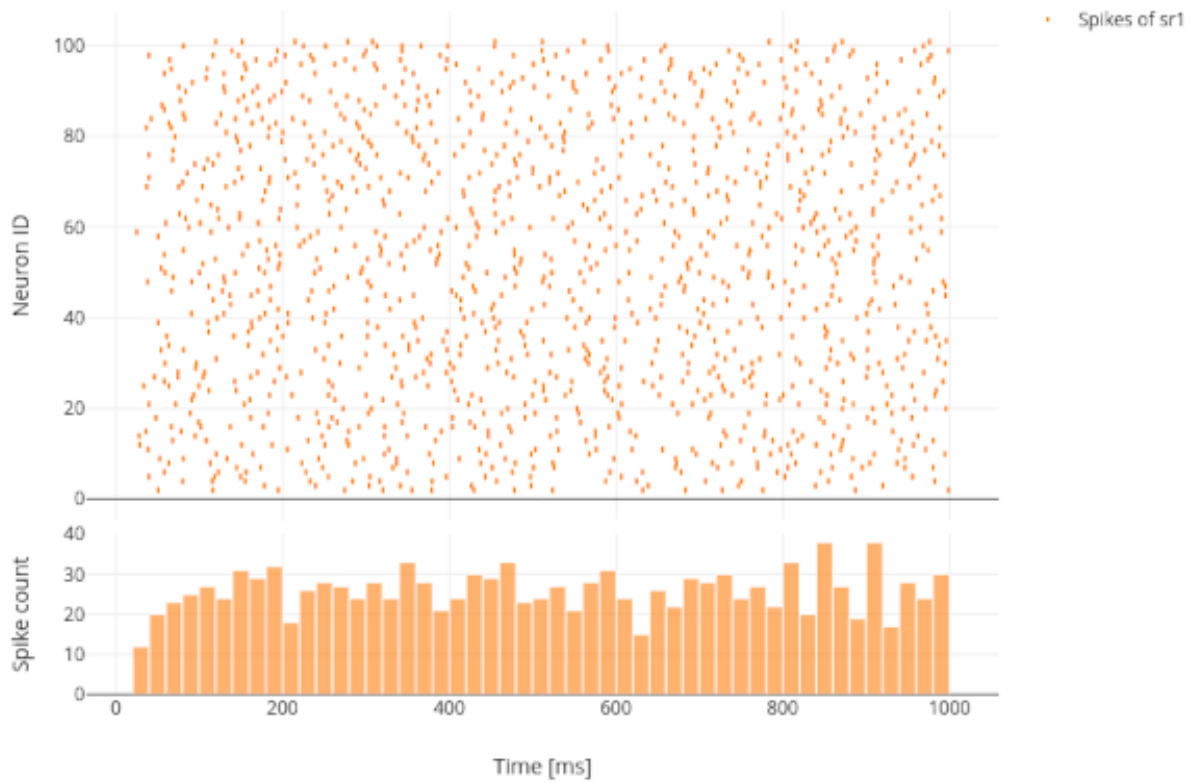


By default, it displays a line trace of the membrane potential.



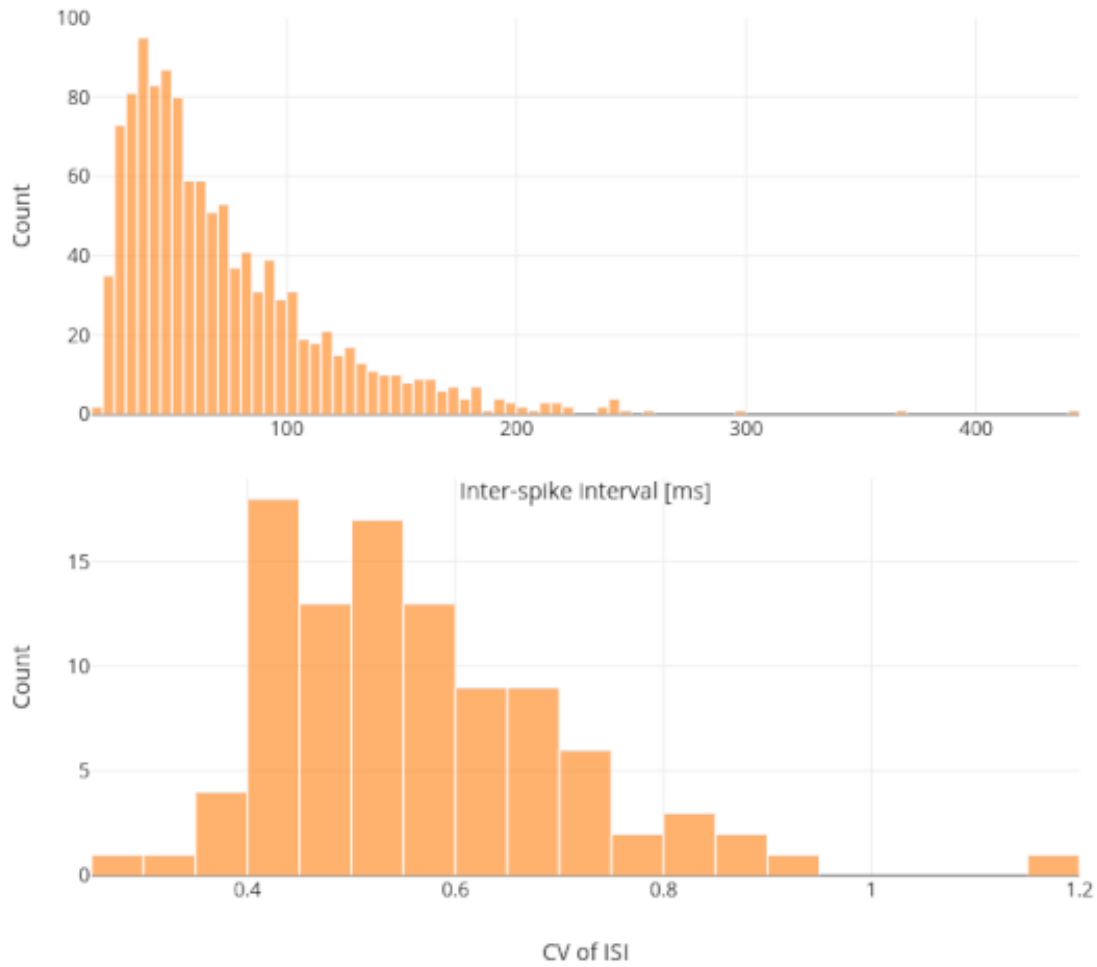
With noise input (noise generator), it shows noise behavior (fluctuation) of the membrane potentials and histogram of distributed values.

## Spike activity

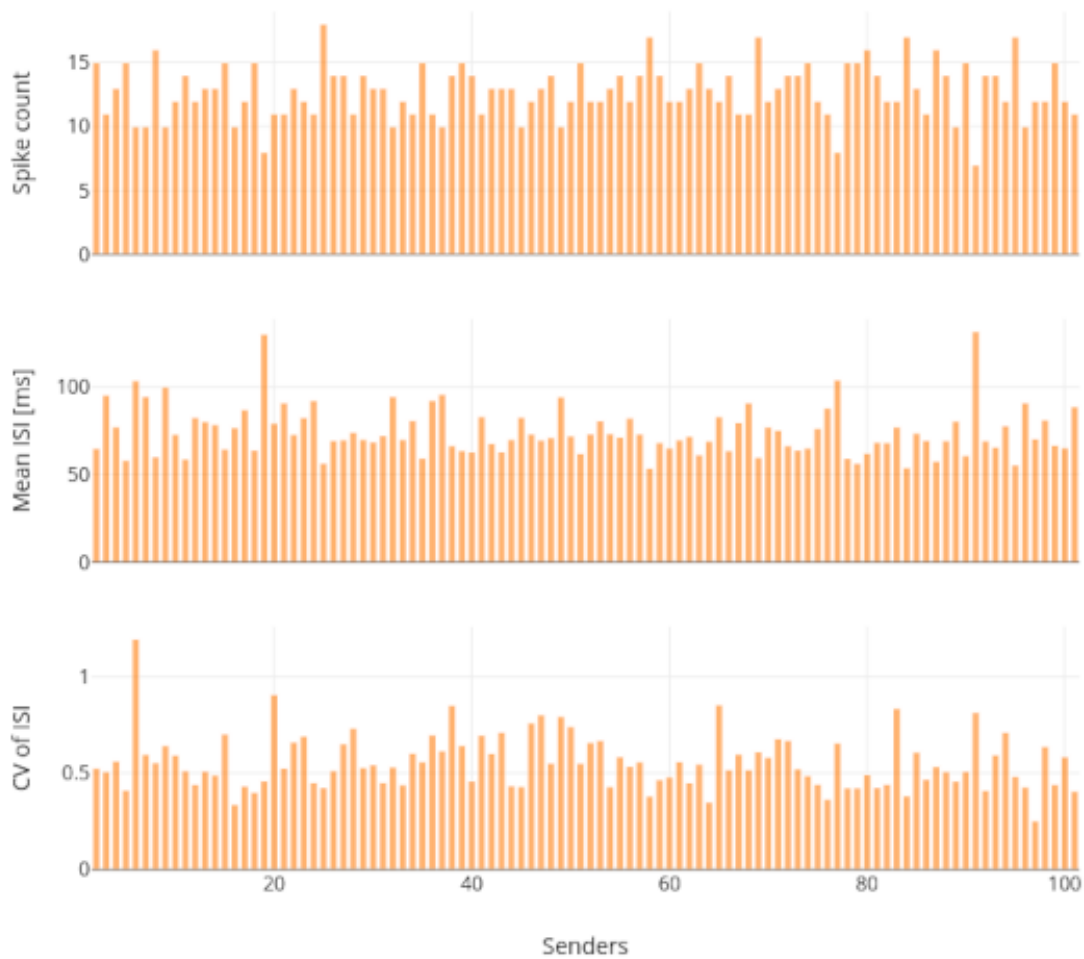


By default, it displays a raster plot of the spike times as well as a time histogram of spikes.





It displays a value histogram of the inter-spike intervals (ISI) as well as of the coefficients of variation of the ISI (CV of ISI) for the population.



It displays spike count, average Inter-spike interval (ISI) and coefficient of variation (CV of ISI) for each sender, e.g. neuron.

### Activity animation graph

It displays an animated 3D graph for the spatial network forming layers in topology whose neurons have geographical positions.

#### See also:

- *Use controller for activity graph*

## Analog signals

Analog signals contain continuous quantities from the recording devices (voltmeter or multimeter).

It is possible to display an animated 3D graph for the spatial network forming layers in topology whose neurons have geographical positions.

Each box represents a neuron in its geographical position. Values of the analog signals can be visualized using the colors of recorded event (here, it shows the color map spectral).

## Spike activity

Spike events contain times and ids of the senders collected by the spike recorder.

Spikes can be visualized as transient blobs appearing in the animated 3D graph. To follow the spike activity better, the trail length can be increased.

Optionally, trails can be faded after the spike time, and a growing or shrinking mode can also be applied.

## Controller sidebar

## Network controller

The screenshot displays the NEST Desktop Network Controller interface. At the top, there are tabs for different model types: ALL, NEURON, STIMULATOR, RECORDER, MODEL, and CUSTOM. The main area shows three configuration cards:

- DC1 DIRECT CURRENT**: Includes sliders for population size (set to 1), amplitude of current (pA) (set to 1), start time (ms) (set to 100), and stop time (ms) (set to 600).
- N1 IAF PSC ALPHA**: Includes sliders for population size (set to 1) and capacity of the membrane (pF) (set to 250).
- VM1 VOLTMETER**: Includes a slider for time interval of recording (ms) on a logarithmic scale from 0.01 to 10, with the value set to 0.1.

At the bottom, there are connection cards showing DC1 connected to N1 and VM1 connected to N1. On the right side, there is a vertical toolbar with icons for Network, Kernel, Code, Activity, and Stats.

The network controller displays cards of models, nodes and connections.

## See also:

- *Copy models*
- *Compartmental model*
- *Synapse model*

## Kernel settings

**SIMULATION KERNEL**

local number of threads: 1, 2, 4, 8

simulation resolution (ms): 0.01, 0.1, 1, 10

seed of the random number generator: 1

☐ randomize seed

**SIMULATION**

simulation time (ms): 1000

Network

Kernel

Code

Activity

Stats

The simulation parameters can be adjusted in the right sidebar. They are contained in the NEST Simulator code (more information below), so they will be passed to the NEST Simulator whenever a simulation is started.

In the Kernel settings, the slider local number of threads allows to set the number of processes used by the NEST Simulator.

It is possible to edit the simulation resolution.

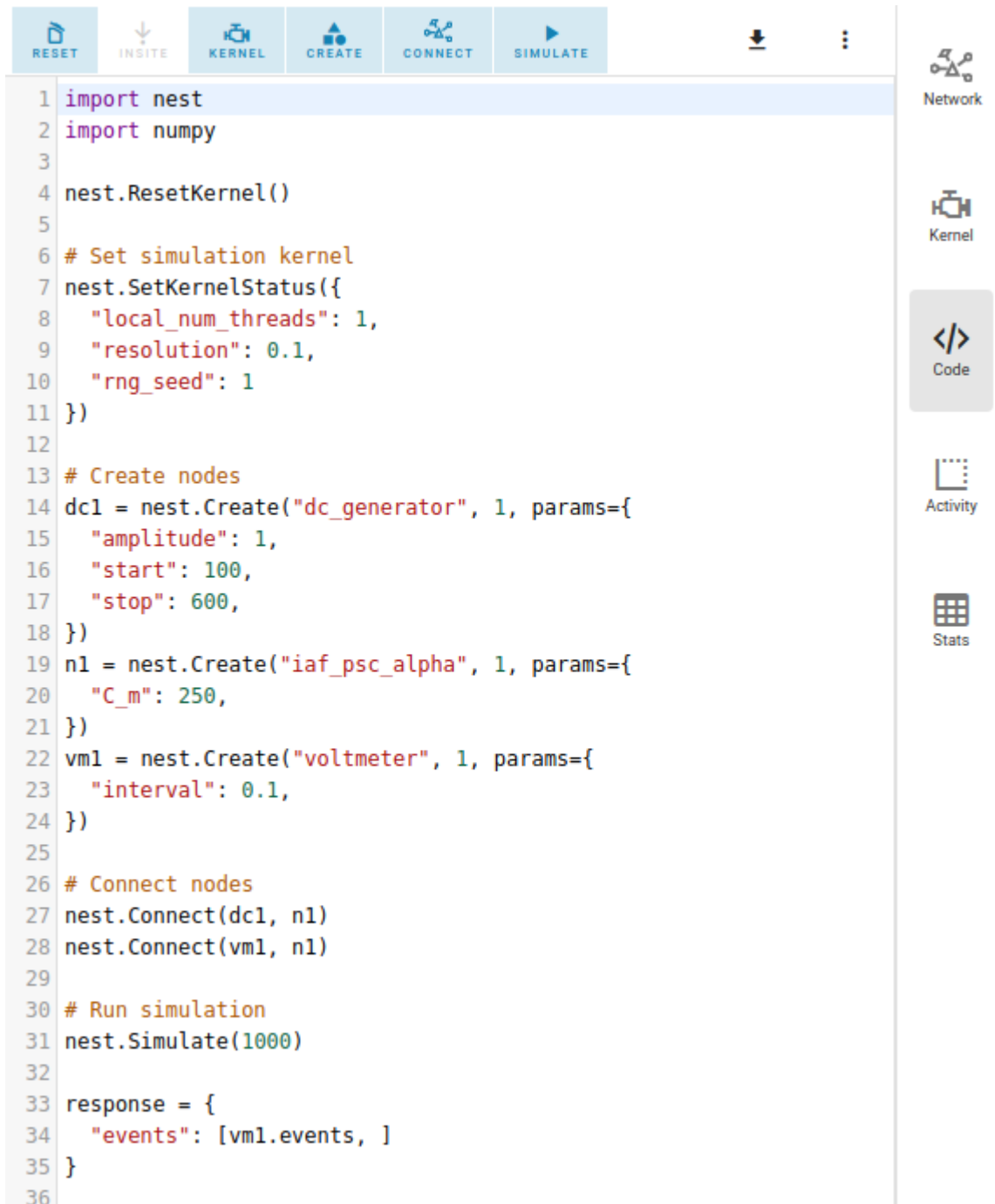
**Note:** Here, you should be aware of the created load on the NEST Simulator as well: Small values for the resolution size create many calculations and data points. Therefore, selecting small values for the simulation resolution can lead to freezes and lags, so please be patient when you choose a small number.

The seed of the random number generator can also be modified. The same seed produces the same simulation output.

It is possible to activate randomized seed that it generates new seed before each simulation.

The simulation time can be set as well (in Milliseconds).

## Code editor



```

1 import nest
2 import numpy
3
4 nest.ResetKernel()
5
6 # Set simulation kernel
7 nest.SetKernelStatus({
8     "local_num_threads": 1,
9     "resolution": 0.1,
10    "rng_seed": 1
11 })
12
13 # Create nodes
14 dc1 = nest.Create("dc_generator", 1, params={
15     "amplitude": 1,
16     "start": 100,
17     "stop": 600,
18 })
19 n1 = nest.Create("iaf_psc_alpha", 1, params={
20     "C_m": 250,
21 })
22 vm1 = nest.Create("voltmeter", 1, params={
23     "interval": 0.1,
24 })
25
26 # Connect nodes
27 nest.Connect(dc1, n1)
28 nest.Connect(vm1, n1)
29
30 # Run simulation
31 nest.Simulate(1000)
32
33 response = {
34     "events": [vm1.events, ]
35 }
36

```

NEST Desktop generates textual code from the constructed network. The generated code can be executed in any Python interpreter. This way, the code semantics of the NEST Simulator is understandable and easy to learn.

The script code starts with importing required modules (`import ...`) and resetting the simulation kernel (`nest.ResetKernel()`). It is necessary, because the old settings/imports of previous simulations have to be removed. Otherwise, errors may occur, e.g. with duplicate imports.

The simulation kernel can be configured by `nestSetKernelStatus(...)`.

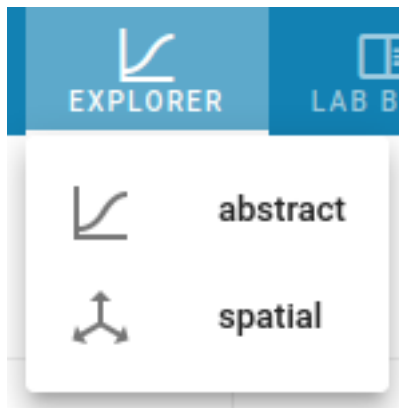
The graphical representatives of the nodes deliver arguments to the block of the `nest.Create(...)` function.

Furthermore, the properties of connections are integrated in the block of the `nest.Connect(...)` function.

The function `nest.Simulate(...)` triggers the simulation of your constructed network.

All recording nodes fill a block to collect activities containing neuronal properties, e.g. node ids and positions, and activity events.

### Activity controller

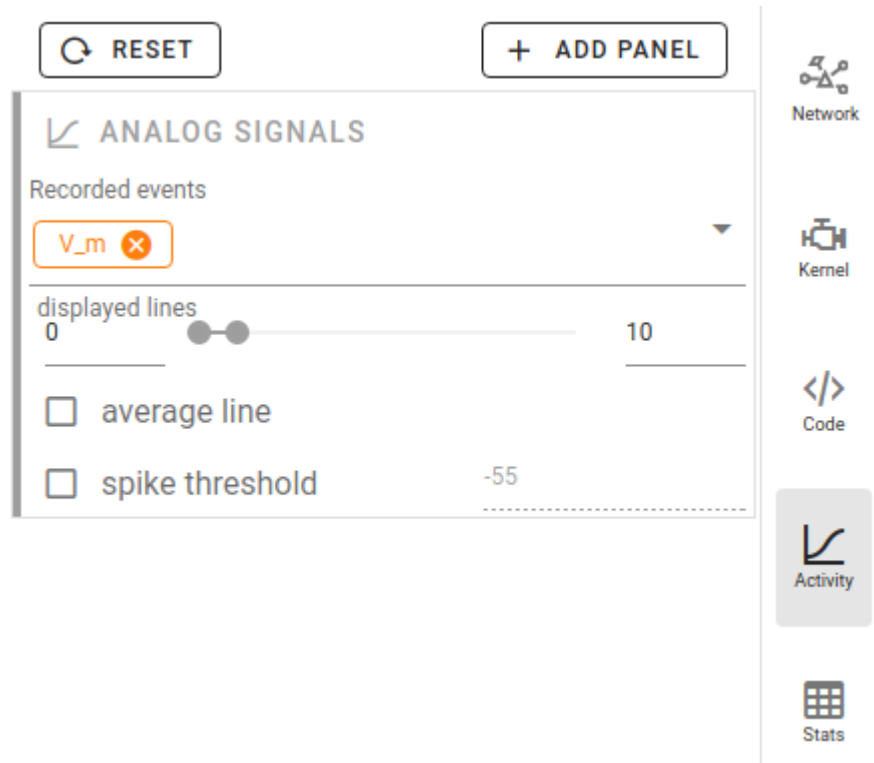


The activity controller displays different content depending on the visualization mode (abstract or spatial) of the activity graph.

### Activity chart controller

Every chart panel has an own controller card for individual customization. Other chart models can be chosen individually for each panel by clicking on the card toolbar in the activity controller.

## Analog signals



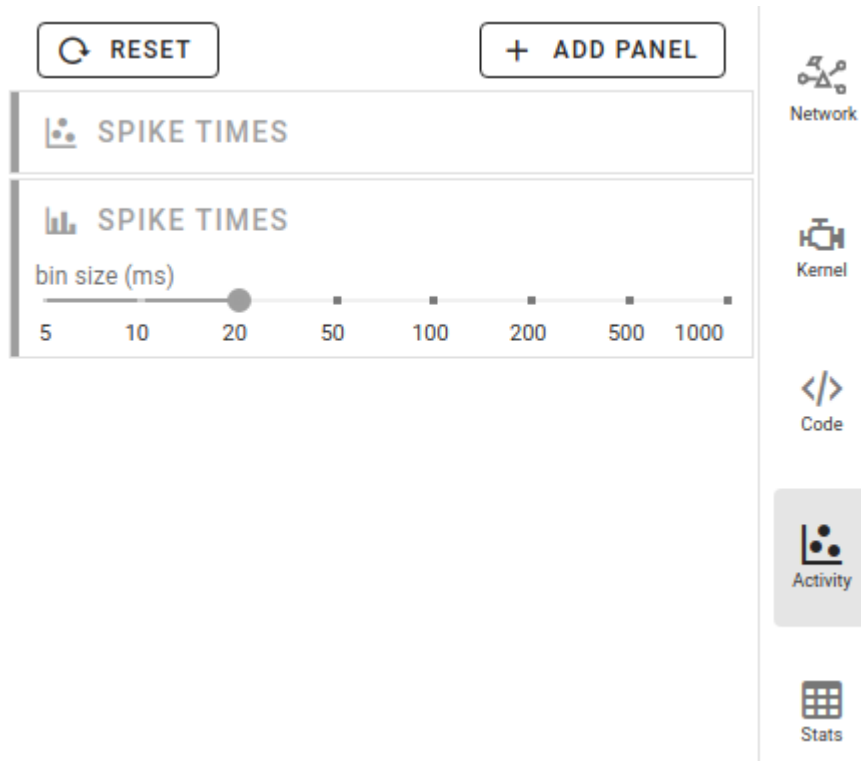
By default, NEST Desktop shows traces of the analog signals as a function of time. A panel with a histogram of values can be added when you select it in the + ADD PANEL dropdown menu.

When something doesn't work properly, you can reset the panels to default by clicking on RESET.

You can add more recorded signals to the panel when it comes from multimeter. Node records appear as chips in the cards, which allow you to change the colors of the corresponding traces and bars.

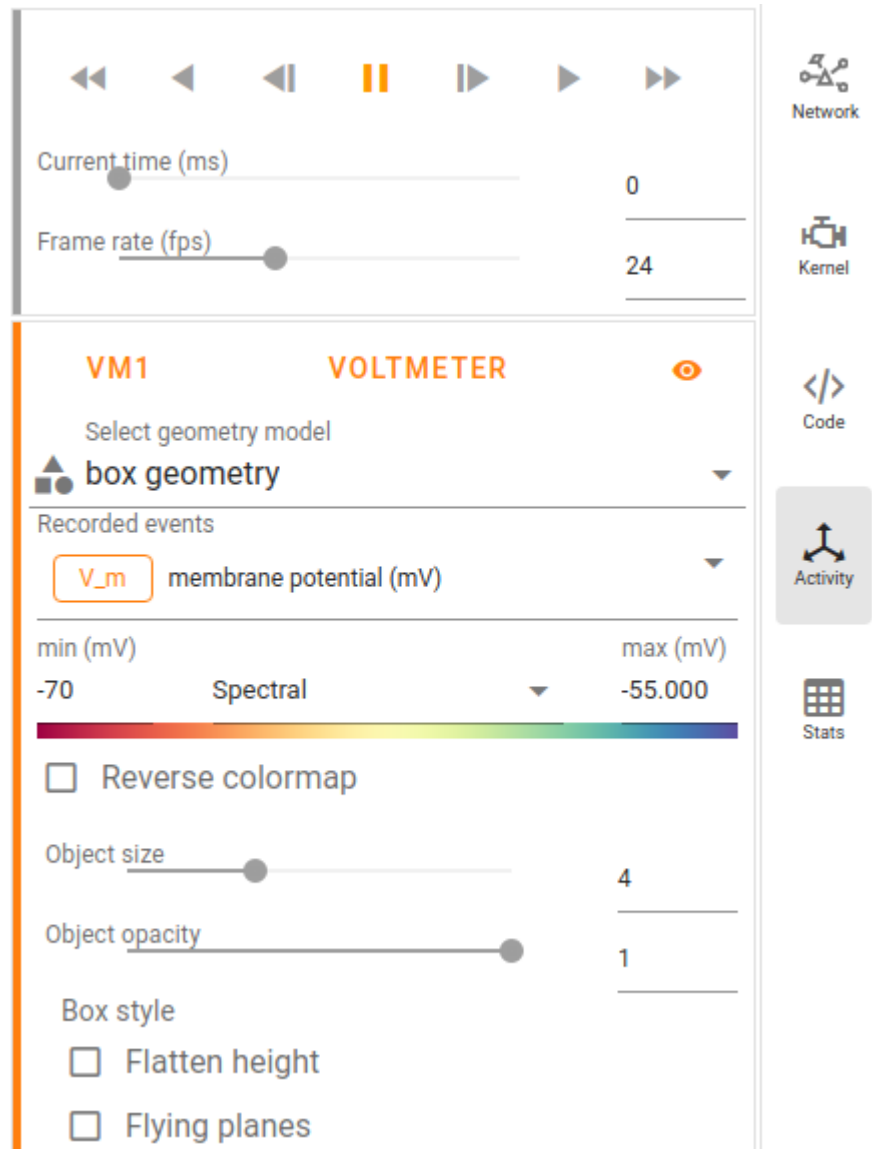


## Spike activity



By default, a raster plot of the spike times as well as a histogram for spike times is shown.

## Activity animation controller



The animated graph displays activity (analog signals or spikes) for the spatial network forming layers in topology whose neurons have geographical positions.

Values of the analog signals can be visualized using the colors of recorded targets. Here, it shows the color map spectral for the value scales (from min to max). You can change the color map in the dropdown menu between the input fields of the min and max values.

Additionally, an other geometry model (box or sphere) can be chosen.

We recommend to try out many different options for the animation graph to find the best representation, as the optimal ones depend heavily on the simulation data and the intended use of the visualization.

## Activity statistics

SR1

SPIKE RECORDER

^

ID ↑

Spikes

ISI mean

ISI std

CV (ISI)

2

16

63.19

28.40

0.45

3

10

82.43

29.71

0.36

4

14

69.37

34.49

0.50

5

12

83.26

44.81

0.54

6

11

78.70

37.35

0.47

7

15

59.65

35.09

0.59

8

16

58.25

33.33

0.57

9

12

81.25

52.61

0.65

10

13

72.83

37.73

0.52

11

15

66.66

40.91

0.61

12

11

92.80

77.31

0.83

13

12

82.98

57.30

0.69

14

14

71.84

35.00

0.49

15

13

74.65

51.23

0.69

16

14

62.48

44.78

0.72

All

Σ = 1271

μ = 75.60

μ = 45.38

μ = 0.59

Network

Kernel

Code

Activity

Stats

Rows per page:

15

1-15 of 100

<

>

It displays multiple panels for each recording device. In each panel a table shows the activity statistics of recorded elements (rows) of a node (population).

In spike events, the columns show the spike counts, mean and standard deviation of *ISI* (inter-spike interval) as well as *CV<sub>ISI</sub>* (coefficient of variation in inter-spike intervals).

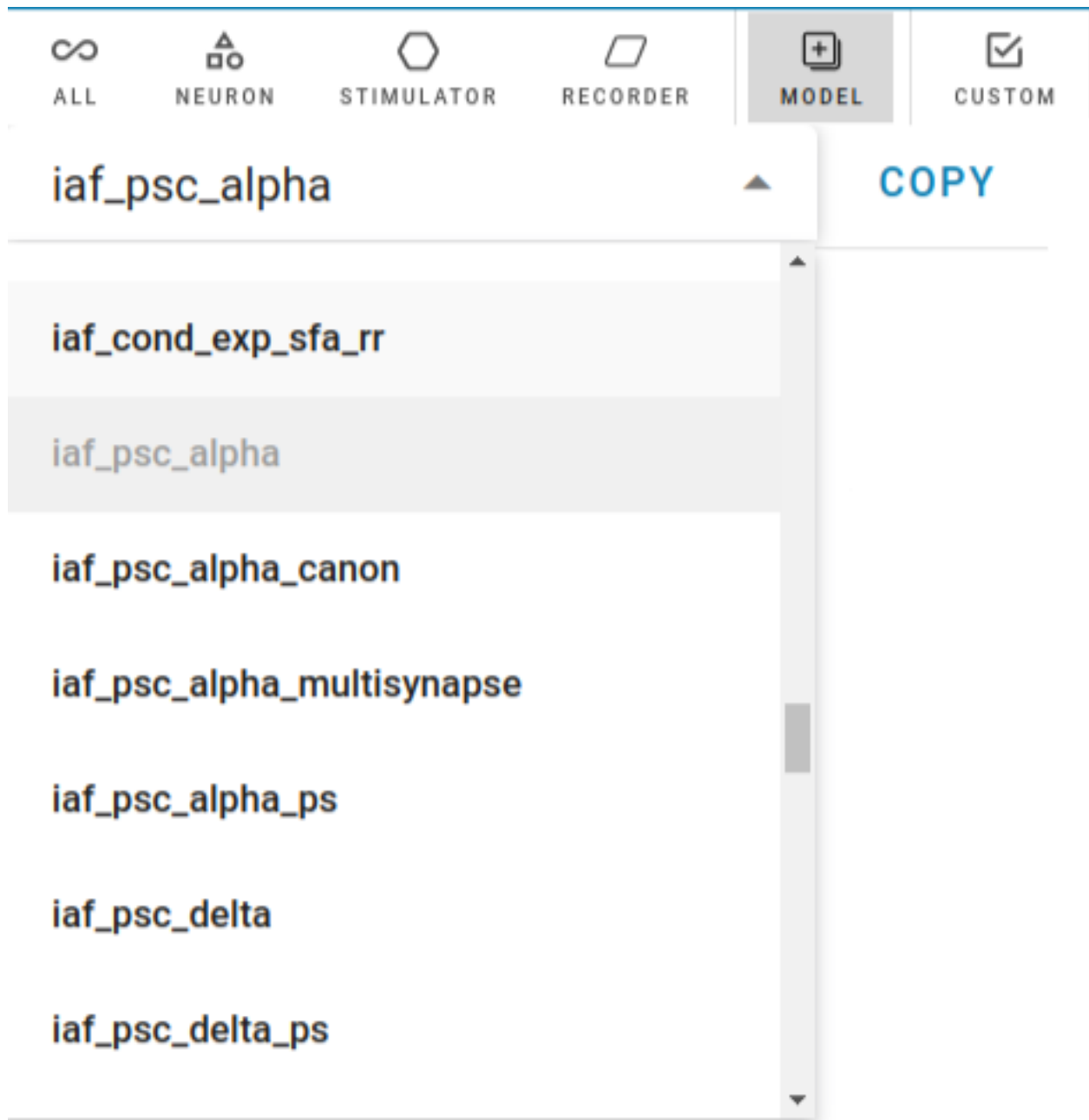
In analog signals (e.g. membrane potentials), the columns show the mean and standard deviation of the values.

## Copy models

NEST Simulator provides a function to copy a model together with its set of parameters. The `nest.CopyModel()` function is useful when multiple populations or synapses should be created with the same set of parameters. This simplifies the work a lot, as you can see in the example below:

Code with CopyModel	Code without CopyModel
<pre> # Copy node models nest.CopyModel("iaf_psc_alpha", "brunel",   ↪params={     "C_m": 250,     "E_L": 0,     "I_e": 0,     "V_m": 0,     "V_reset": 0,     "V_th": 20,     "t_ref": 2,     "tau_m": 20,     "tau_syn_ex": 0.5,     "tau_syn_in": 0.5,   })  # Create nodes n1 = nest.Create("brunel", 100) n2 = nest.Create("brunel", 25) </pre>	<pre> # Create nodes n1 = nest.Create("iaf_psc_alpha", 100,   ↪params={     "C_m": 250,     "E_L": 0,     "I_e": 0,     "V_m": 0,     "V_reset": 0,     "V_th": 20,     "t_ref": 2,     "tau_m": 20,     "tau_syn_ex": 0.5,     "tau_syn_in": 0.5,   }) n2 = nest.Create("iaf_psc_alpha", 25,   ↪params={     "C_m": 250,     "E_L": 0,     "I_e": 0,     "V_m": 0,     "V_reset": 0,     "V_th": 20,     "t_ref": 2,     "tau_m": 20,     "tau_syn_ex": 0.5,     "tau_syn_in": 0.5,   }) </pre>

## How to copy models - step by step



Click on the MODEL tab in the network controller and then select a model to copy. Then confirm with a click on COPY.

IAF\_PSC\_ALPHA

New label

brunel

capacity of the membrane (pF)

250

leak reversal potential (mV)

+

0

constant external input current (pA)

0

initial membrane potential (mV)

0

reset potential of the membrane (mV)

0

spike threshold (mV)

20

duration of refractory period (ms)

2

membrane time constant (ms)

20

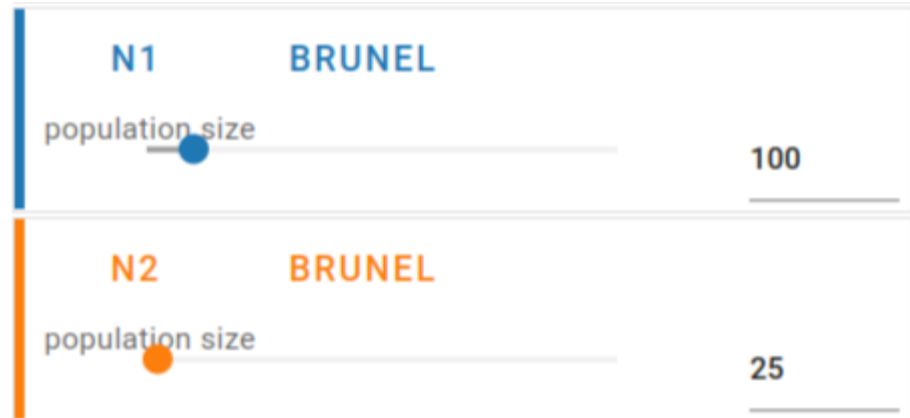
time constant of the excitatory synapse (ms)

0.5

time constant of the inhibitory synapse (ms)

0.5

Enter the name of the new model. If you like to have other model parameters than the default one, just click on the model title and select the parameters you want to change. This opens the sliders and fields to edit their values.



Choose the copied node model for your node (e.g. in the nodes list).

**Note:** Copied synapse models can also be applied for synapses (analogously as above).

### Compartmental model

NEST Simulator is actually a simulation tool of point-neurons but it also provides a model `cm_default` which is a neurons with compartments. Here, the guide shows the steps to create a simple neuron with compartments.

## Step by step guide

## Compartments

soma 1 

dendrite 1 

dendrite 2 

+

capacitance of the compartment	10 pF	<input checked="" type="checkbox"/>
--------------------------------	-------	-------------------------------------

coupling conductance with parent compartment	0 nS	<input checked="" type="checkbox"/>
--	------	-------------------------------------

leak conductance of the compartment	1 nS	<input checked="" type="checkbox"/>
-------------------------------------	------	-------------------------------------

leak reversal of the compartment	-70 mV	<input checked="" type="checkbox"/>
----------------------------------	--------	-------------------------------------

sodium maximal conductance	0 nS	<input type="checkbox"/>
----------------------------	------	--------------------------

sodium reversal	50 mV	<input type="checkbox"/>
-----------------	-------	--------------------------

potassium maximal conductance	0 nS	<input type="checkbox"/>
-------------------------------	------	--------------------------

potassium reversal	-85 mV	<input type="checkbox"/>
--------------------	--------	--------------------------

## Receptors in soma 1

GABA 

+

GABA reversal	-80 mV	<input type="checkbox"/>
---------------	--------	--------------------------

GABA rise time	0.2 ms	<input type="checkbox"/>
----------------	--------	--------------------------

First import `cm_default` from GitHub and create a node with `cm_default`. Then open node selection popup, add compartments () and select compartment parameters to modify. Add receptors () in each compartment and select receptor parameters to modify.

Click on the chips (soma 1, dendrite 1, ...) of a compartment to see its content.



N1

COMPARTMENTAL NEURON

population size

1

spike threshold (mV)

-50

Compartments

soma 1

dendrite 1

dendrite 2

capacitance of the compartment (pF)

10

coupling conductance with parent compartment (nS)

0

leak conductance of the compartment (nS)

1

leak reversal of the compartment (mV)

-70

Receptors in soma 1

GABA

You can modify the values in each compartment and its receptors.

**Note:** Use multimeter to record events from various compartments.

## Synapse model

NEST Desktop is able to apply synapse models to the connections between neurons. Here, we show the steps how to observe neuronal activity in aspect of short-term plasticity and how to measure synaptic weights.

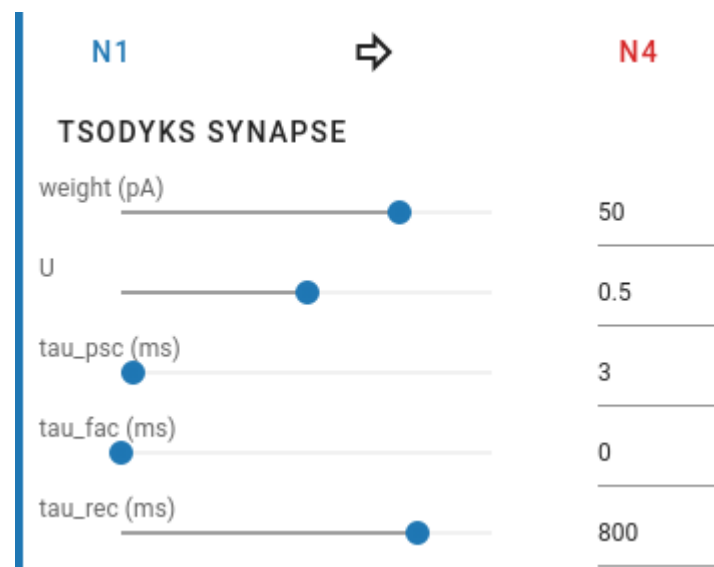
### Import synapse model

First, you have to import synapse models.

See also:

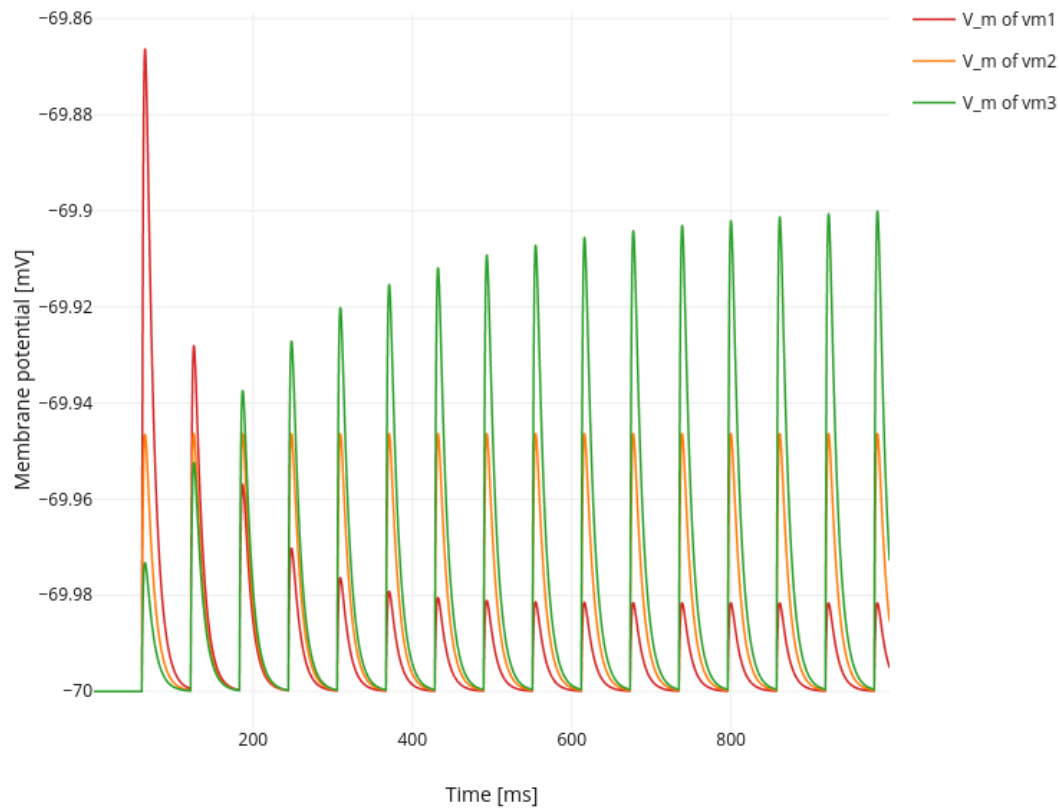
- *Import models in dialog*

### Connect neurons with non-static synapse



After you have initially built the neurons with their connections, you can select another synapse model (Here: Tsodyks synapse). Configure the parameter values for facilitating or depressing the synapse.

### Observe effects of short-term synapses



After the simulation you might register changes of PSP of neurons receiving spike inputs from other neurons via non-static synapses.

## Measure synaptic weights

**TSODYKS\_SYNAPSE**

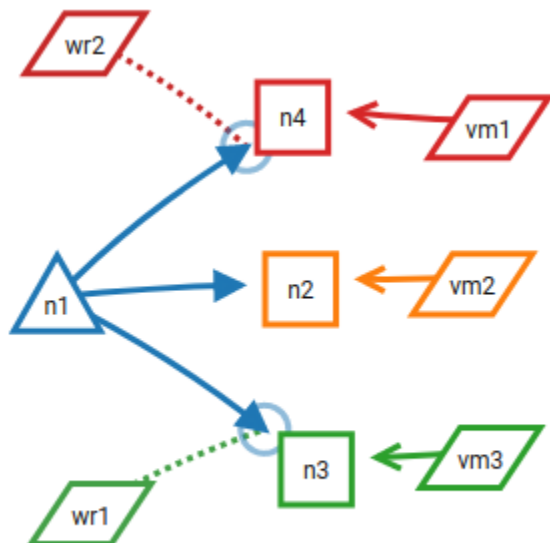
New label  
**tsodyks\_fac**

weight (pA)	<input type="range"/>	250
U	<input type="range"/>	0.03
tau_psc (ms)	<input type="range"/>	0.3
tau_fac (ms)	<input type="range"/>	530
tau_rec (ms)	<input type="range"/>	150
weight_recorder	<input type="text" value="wr1"/>	▼

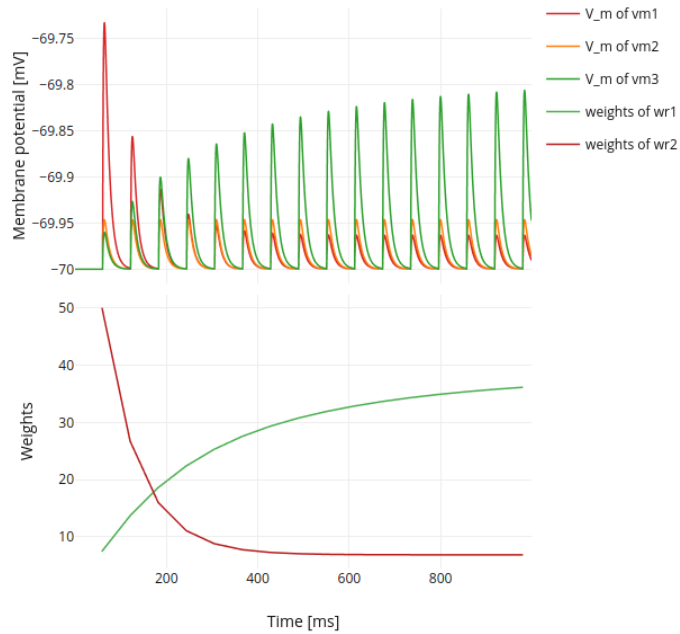
A weight recorder is not a typical recorder like others. It can only be assigned to a synapse model to measure its weight.

First, import WEIGHT RECORDER from GitHub. You need to copy the synapse model whose weight should be recorded. Select the copied synapse model for an existing connection between neurons.

Create a node with WEIGHT RECORDER and connect it to a connection (use the connection as the target instead of a node). You can see in the copied synapse model that it is assigned to WEIGHT RECORDER.



After the simulation, add a new panel showing only weights.



```

33
34 # Copy synapse models
35 nest.CopyModel("tsodyks_synapse", "tsodyks_fac", params={
36     "weight": 250,
37     "U": 0.03,
38     "tau_psc": 0.3,
39     "tau_fac": 530,
40     "tau_rec": 150,
41     "weight_recorder": wr1,
42 })
43 nest.CopyModel("tsodyks_synapse", "tsodyks_dep", params={
44     "weight": 100,
45     "U": 0.5,
46     "tau_psc": 3,
47     "tau_fac": 0,
48     "tau_rec": 800,
49     "weight_recorder": wr2,
50 })
51
52 # Connect nodes
53 nest.Connect(n1, n4, syn_spec="tsodyks_dep")
54 nest.Connect(n1, n2, syn_spec={
55     "weight": 10,
56 })
57 nest.Connect(n1, n3, syn_spec="tsodyks_fac")
58 nest.Connect(vm1, n4)
59 nest.Connect(vm2, n2)
60 nest.Connect(vm3, n3)
61
62 # Run simulation
63 nest.Simulate(1000)
64

```

### 3.3.3 External software

#### Simulate with Insite

This is a guide to show how to use NEST Desktop with Insite.

Insite is a recording backend module which is also usable with NEST Simulator. Basically, with Insite, neuronal or network activity can be observed during the simulation.

**Note:** Simulations with Insite need to be run with the Insite docker images `nest-module` and `access-node`. The best method is to use Docker Compose, which also deploys NEST Desktop and Insite. For more information, please read the [deployment guide of Insite](#).

## Check if Insite is running

## Insite

In-situ recording backend

URL of Insite access

`http://localhost:52056`

Please enter the URL where the server of Insite can be found at (including protocol!).

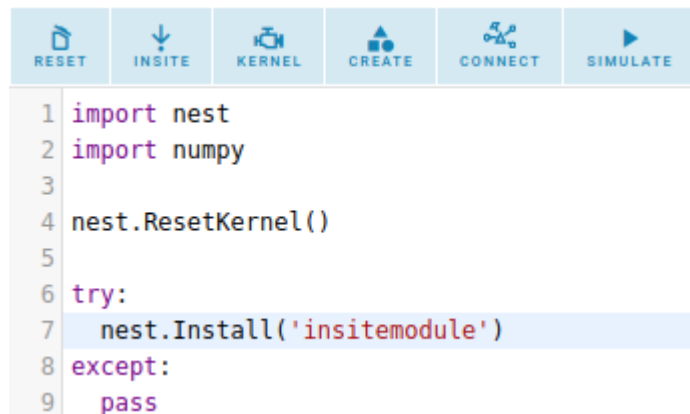
CHECK

Response:

✓ Insite version: 1.1

In the settings page you can check whether the Insite backend is running. When it is disabled, you can toggle the slide to enable it.

## Enable simulation with Insite



After successfully receiving a ping from the Insite access node of the backend, you can activate the button Insite (second from left) in the toolbar of the code editor. Then, NEST Desktop generates the script such that Insite is used during the simulation.

## Script code for simulation with Insite

The Insite module has to be loaded into the NEST kernel. Preferentially load the `insitemodule` after importing NEST:

```
nest.Install('insitemodule')
```

Next, check whether the parameter `record_to` of any recording device (e.g. `spike recorder`, `multimeter` or `voltmeter`) has to be modified:

```
...
recorder.set({"record_to": "insite"})
...
```

Now, the Insite recording module collects activity events from the recording devices in the NEST Simulator. NEST Desktop receives activity from the Insite access node on another port (default: 52056).

#### See also:

For more information about Insite, please visit the official [documentation of Insite](#) from the VR Group of RWTH Aachen.

## Acknowledgements

Thanks for integrating Insite in NEST Simulator and NEST Desktop:

- Simon Oehrl (Conceptual design for Insitufication in NEST Desktop)
- Marcel Krüger (Collaboration on Insitufication in NEST Desktop)

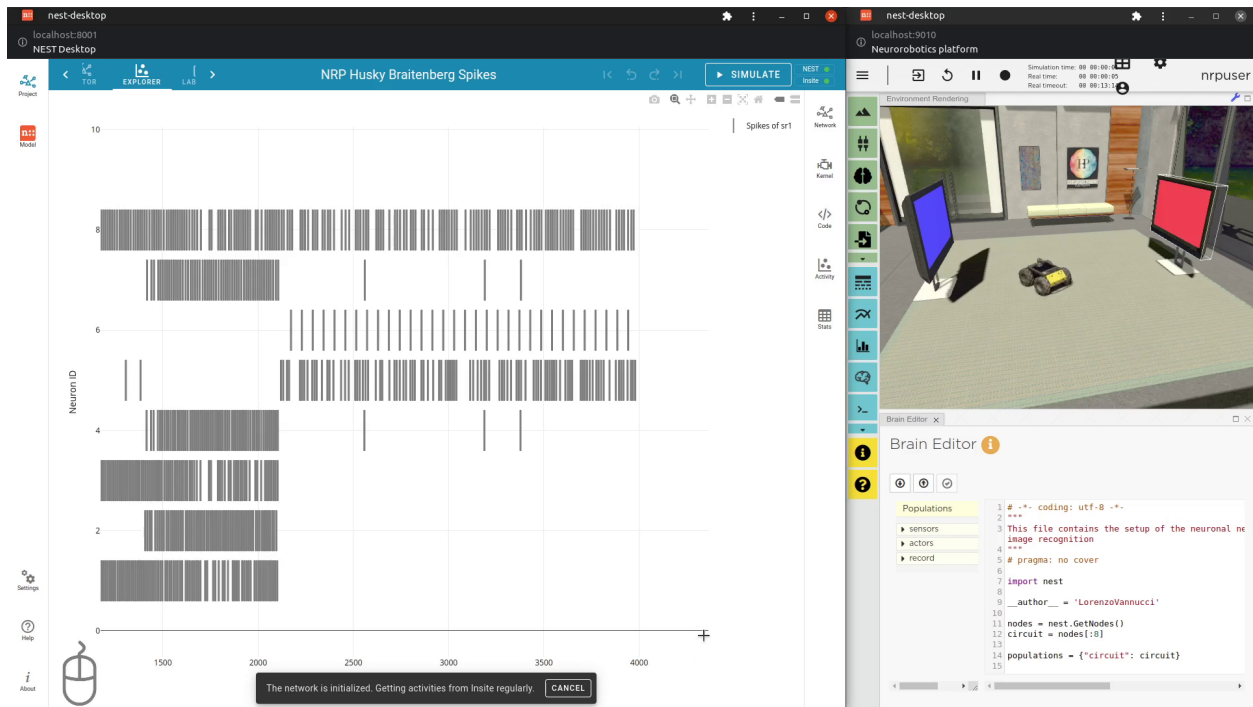
## Use NEST Desktop with NRP

This is a guide to show how to use NEST Desktop with NRP (Neuro-Robotic Platform).

The NRP enables users to perform virtual experiments on virtual objects (e.g. robots, robotic arms) or on virtual animals (e.g. rats).

In our case we use NRP to learn neuronal activity dynamics of the “robot brain” according to a thought experiment by the cyberneticist [Valentino Braitenberg](#).

His thought experiment demonstrated a simple concept of the brain interacting with the environment with a simple machine. The simplest concept of this so-called *Braitenberg vehicle* <[https://en.wikipedia.org/wiki/Braitenberg\\_vehicle](https://en.wikipedia.org/wiki/Braitenberg_vehicle)> shows direct connections from two sensors to two individual wheels. This implies, that the signal strength controls the rotation speed of the wheel.



In our experiment, our robot (“Husky”) acts similar to the Braitenberg vehicle. Additionally to the basic setup in “Husky”, we implanted a “robot brain” receiving electrical signals from the sensors, processing them and transferring signals to motors/engines. In summary, the robot brain controls the movement and thus our robot “Husky” is able to react to the environment.

In the NRP experiment, we see our robot “Husky” and two monitors showing a screen color, e.g. blue, red, green. The user can change the screen color. During the live experiment, the Husky rotates itself during a non-attractive situation (blue and green screen). However, when one of these monitors shows an attracting color (red) and the camera sees it, the Husky moves towards to it.

In NEST Desktop we can observe the spike activities of the “Husky brain”.

### How to perform a simulation with NRP and NEST Desktop

See also:

- [Deploy NEST Desktop with NRP.](#)

First, open two browser windows/tabs, one for NEST Desktop and another for the NRP page. You can place two windows side-by-side to see both at the same time.

In NEST Desktop import the project “Husky experiment” (from GitHub). Investigate the network and prepare the simulation.

In the NRP you have to clone and launch the Husky experiment. Then, you can start the experiment in the virtual environment. Here, you can see that the robot with the “Husky” network (the “Husky”) is rotating. Now, switch the screen color of a monitor to red. Watch until the “Husky” moves toward the red screen.



Observe the spike activity in NEST Desktop (recorded by Insite).

**See also:**

For this approach, we need to run the simulation with Insite as recording backend.

When you want to learn how to use NEST Desktop with Insite, please read [Simulate with Insite](#).

## Acknowledgements

Thanks for the collaboration on NRP and NEST Desktop:

- Viktor Vorobev (Collaboration on NRP and NEST Desktop)
- Marcel Krüger (Insite as recording backend)
- Fabrice Morin (Contact person of NRP)
- Jochen M. Eppler (NEST Server MPI)
- Benedict Feldotto (Implementation of NEST Server in NRP)

## Use NEST Desktop with ViSimpl

ViSimpl visualizes neural activity from brain simulation data. It displays spike activity in space and can be co-used with NEST Desktop.

**See also:**

For this approach, we need to run the simulation with Insite as recording backend.

When you want to learn how to use NEST Desktop with Insite, please read [Simulate with Insite](#).

## Preparation

First, download the Docker Compose configuration file for NEST Desktop and Insite.

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop-docker/main/examples/  
↪insite/docker-compose.yml
```

Then, pull the docker images using Docker Compose.

```
docker-compose pull
```

Afterwards, you can start NEST Desktop (with Insite).

```
docker-compose up
```

---

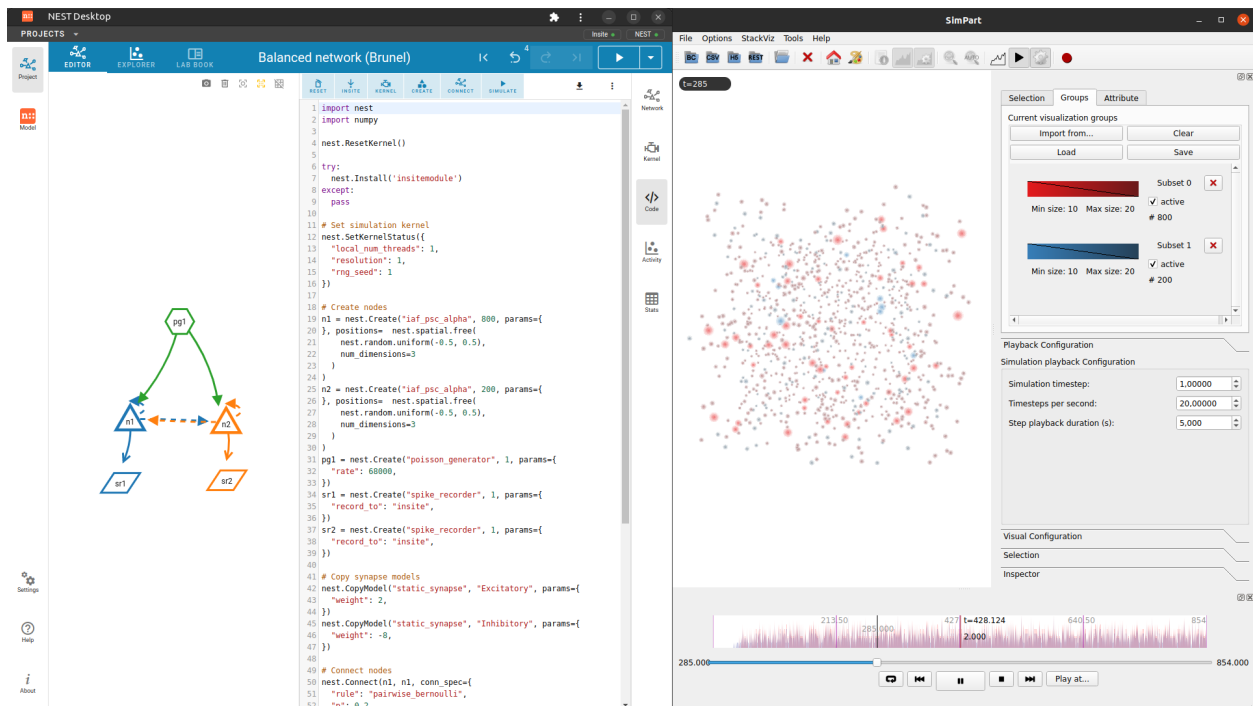
For ViSimpl, download the (binary) AppImage from the [page](#), make it executable and then open it.

VERSION=1.8.3

```
wget https://vg-lab.es/apps/visimpl/latest-release/visimpl-$VERSION-x86_64.AppImage
chmod +x visimpl-$VERSION-x86_64.AppImage
./visimpl-$VERSION-x86_64.AppImage
```

**Hint:** You can place NEST Desktop and ViSimpl side by side to see them both.

## How to use NEST Desktop with ViSimpl



### Steps

- In NEST Desktop, make sure that both backends (NEST Simulator and Insite) are running.
- Run the simulation of the network with Insite as recording backend.
- In ViSimpl, click on the REST button to get data from Insite (check that you use the correct URL).

### Hint:

- Increase the Request size to 10 000 spikes in the REST dialog that it collects spikes faster.

- It shows spatial dots representing neurons and spikes are displayed in glowing mode.

### Hint:

- Increase the Simulation timestep to 1ms in Simulation Playback Configuration.

- Increase the Delay to 5ms in Visual Configuration.
- 

## Acknowledgements

Thanks for the collaboration on ViSimpl and NEST Desktop:

- Félix De Las Pozas Álvarez (Collaboration on ViSimpl and NEST Desktop)
- Marcel Krüger (Insite as recording backend)
- Óscar David Robles Sánchez (Lead developer of ViSimpl)

## 3.4 Lecturer guide

This section gives directions for lecturers who want to teach computational neuroscience with NEST Desktop. Using computer simulations, students are able to explore models ranging from single neurons to large neuronal networks. Numerical experiments of increasing complexity help to understand how brain networks function and what the features of their dynamic behavior are.

---

**Note:** This section assumes that you have prior knowledge of how to use NEST Desktop. If you have not used NEST Desktop before, please read the User Documentation first ( *User guide* ).

---

### 3.4.1 Course organisation

To support the organization of a course, we provide some hints for course instructors:

- *Course design*
- *Didactic concept*

Additionally, we provide course materials to be handed out to participants:

- *First steps toward brain simulation*
- *How to prepare a course protocol*

### Course design

A course with NEST Desktop can be organized for students at different levels of university education. The course can be held in a computer lab, allowing for personal interaction with other course participants and tutors. Alternatively, the course can be held online, where students use their own computers at home and interact by video chat. For both options, NEST Desktop is deployed in a virtual machine that is controlled by the students via a standard web browser.

A typical course could be scheduled as a one-week block course with fixed hours for complementary lectures, technical tutorials and consultation hours. The complementary lectures provide the basic background information needed to understand the course assignments. The technical tutorials explain step by step the operation of NEST Desktop. The consultation hours give students an opportunity to ask the tutors questions about the course assignments. Students can also post their questions and comments to the teaching staff by email.

Specific sample assignments are provided for students with different background knowledge.

### Didactic concept

NEST Desktop focuses on teaching university students, for whom significant programming skills cannot be assumed. The goal is to give them an introduction to computational neuroscience, and illustrate how computer simulations are used in the field. NEST Desktop provides an intuitive graphical user interface to NEST, which is actually a script-based simulation tool widely used in research. The idea is that students approach and understand important concepts in neuroscience by means of interactive construction, simulation and analysis of neuronal network models. This functionality is enabled by visual elements that can be manipulated with the computer mouse. No script-based programming is required at this stage. Thanks to this intuitive approach, learning is fast, and students can devote their time and attention to neuroscientific content rather than code syntax and data structures.

NEST Desktop still implements the standard workflow in the computational sciences: model creation, numerical simulation, statistical analysis and visualization of the simulation outcome. This logic represents another didactic dimension of NEST Desktop, beyond lowering the threshold for novices to use complex and powerful simulation engines like NEST. In addition, it is possible to inspect the automatically generated NEST code and even change it before simulation. This way, the students get some insight into the script-based interface of NEST, which enables more complex simulations.

For a typical course in computational neuroscience, the following combination of three course elements has proven to be effective:

1. **A theoretical introduction to computational neuroscience** using slide-based presentations, possibly enhanced by the interactive use of NEST Desktop as a demonstrator during the lecture.
2. **An interactive tutorial** explaining how NEST Desktop can be employed to work on course assignments.
3. **Structured lab reports** exploiting the capabilities of NEST Desktop with regard to creation, simulation and analysis of models, potentially prepared in small working groups.

Please refer to the various examples of specific assignments offered in this documentation.

### First steps toward brain simulation

In this course, students will learn how computer simulations are employed in brain research. Sometimes, these simulations are very close to experiments, and a viable goal may be to mimic the known biological reality as good as possible. In other cases, the motivation is rather to devise a strategy for entirely new experiments. In this case, a computer simulation is more like a thought experiment, leading to a new hypothesis how things might work. This is the promise and the potential of computational neuroscience.

In experiments with animals or humans, brain activity must be recorded with suitable advanced instruments (e.g. electrodes, optical instruments, or brain scanners). This is exactly what researchers also do in simulations. For that reason, simulated data typically looks very much like experimental data, and they have to be analyzed like experimental data. To some degree, therefore, this is also a course on neuroscientific data analysis. In contrast to experiments with biological brains, in a simulation one has access to much more detailed information about the inner workings. As a consequence, computer simulations can sometimes provide insight that goes beyond what is achieved by experiments, even with latest techniques. This sets the stage for what students can expect from this course, and what students need to do to get the most out of it.

Specifically, students learn how to ...

## How to prepare a course protocol

The course focuses on the function of synaptically coupled neuronal networks in the brain. Step by step, simulation experiments of increasing complexity will be performed. The students' task is to document what they did, why they did it, and what was the outcome, using precise language and appropriate terminology. In addition, students will need to design meaningful illustrations and parameter tables describing the simulation.

## How to achieve this?

Simulations with NEST Desktop can be exported to *PDF*, *PNG* or *SVG* files. For PDF export, students can increase the zoom factor of the browser before printing the page. Image files can be imported to a document processor, e.g. LibreOffice or LaTeX for annotated illustrations of the simulation. It is important, however, that the writing is focused, and images to import into the protocol are carefully selected. As a final step, the protocol will be exported to a single PDF file and submitted by email to the course instructors.

Protocols must be concise, complete and correct. For each assignment, students must separately address the following four aspects:

---

1. **What is the scientific question or problem that is approached by the simulation?** Use correct terminology in descriptions. Keep neurobiological aspects and mathematical modeling concepts apart.
  2. **Which exact simulation setup was used to answer this question?** List and describe all its components (e.g. neurons, devices), specify how they are connected together (e.g. electrodes, synapses), and provide tables of all relevant parameters. Default settings need to be stated at the beginning of each section. The goal is to make simulations fully reproducible.
  3. **What was the outcome of the simulations performed?** A meaningful set of figures should be selected to underpin the outcome of the simulations. In many cases, figures are helpful to illustrate the text-based description of the role of a specific parameter for the behavior of the system in question (e.g. a neuron, a network).
  4. **What are the conclusions in view of the simulation results?** Based on the simulation results obtained, formulate appropriate answers to the original questions. Distinguish aspects of biology/biophysics and possible issues (e.g. shortcomings) of the numerical model.
-

### 3.4.2 Course topics

This guide shows how you can teach the biophysics of neurons, synapses and large networks of the brain to students using NEST Desktop. Video tutorials illustrate important aspects of the course work. The provided material could be used to prepare handouts for students.

#### Bachelor students

This section provides sample assignments for students with little prior knowledge. The focus lies on the activity dynamics of single neurons. In all assignments we use `iaf_psc_alpha` as our neuron model. It is studied how a neuron responds to different types of input.

- `/lecturer/bachelor/single-neuron-direct-current-injection`
- `/lecturer/bachelor/single-neuron-noise-current-injection`
- `/lecturer/bachelor/single-neuron-synaptic-input`
- `/lecturer/bachelor/single-neuron-poisson-input`

#### Master students

This section covers advanced topics for students with previous knowledge in neurobiology. Here, we cover the activity dynamics of single neurons and of neuronal networks.

- `/lecturer/master/hodgkin-huxley-action-potential`
- `/lecturer/master/point-neuron-conductance`
- `/lecturer/master/network-dynamics`

#### Doctoral students

This section illustrates how NEST Desktop might be used for research in computational neuroscience. A typical example covers the activity dynamics of neuronal networks with multiple interacting populations.

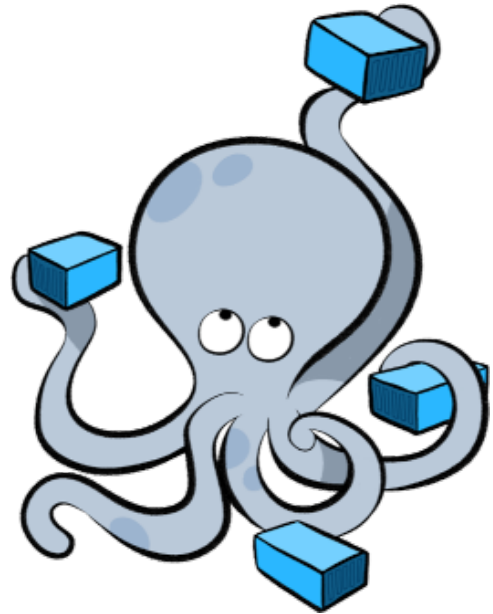
- `/lecturer/doctorate/networks-decision-making`

## 3.5 Deployer guide

This guide provides detailed documentation on how to deploy NEST Desktop. You can read the deployment instructions by clicking one of these images below.

### 3.5.1 Deploy NEST Desktop

#### Deploy with Docker Compose



Docker is a virtualization software packaging applications and its dependencies in a virtual container that can run on any Linux server. It is available for a variety of the operating systems, e.g. Linux, Mac and Windows. For more information follow the link [here](#).

NEST Desktop and NEST Simulator are prepared in different containers, but you can use docker-compose to start multiple containers, e.g. NEST Desktop, NEST Simulator. Docker Compose needs the configuration file (`docker-compose.yml`).

Here, the guide shows you how to build containers with `docker-compose`.

#### Requirements:

- [Docker Compose](#)

#### Preparation

Prepare your local environment by installing Docker (if you have not installed it yet).

```
apt install docker-compose
```

### Get the configuration file

The configuration file *docker-compose.yml* contains all setup steps executed by Docker. Fetch this file from GitHub:

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop/main/docker-compose.yml
```

It will pull images of NEST Desktop from <https://docker-registry.ebrains.eu/harbor/projects/6/repositories/nest-desktop> and NEST Simulator can be started from within the official NEST image (<https://docker-registry.ebrains.eu/harbor/projects/6/repositories/nest-simulator>).

### Getting started

Build and start the NEST Desktop and NEST Simulator containers.

```
docker-compose up --build
```

NEST Desktop and NEST Simulator are now serving at <http://localhost:54286> and <http://localhost:52425>, respectively. With CTRL + C you can shutdown these services.

### Configurations in *docker-compose.yml*

Here, you can find the details of the configuration file.

image	Get docker image from Docker Hub
container_name	Set container name
ports	Bind host ports to container ports
command	Execute command on container start
environment	Set environment variables

Alternatively, you can clone the source code so that you can change the Dockerfile and build custom docker images on your machine. For more information, visit the page <https://github.com/nest-desktop/nest-desktop-docker>.

### Upgrade images

First stop the containers and shut down all services “nest-desktop” and “nest-simulator”.

```
docker-compose stop  
docker-compose down
```

Then pull images from docker hub.

```
docker-compose pull
```

Afterwards, you can start the services and containers.



```
docker-compose up --no-start
docker-compose start
```

## Useful commands

In the following you can find some useful commands for docker-compose.

List containers.

```
docker-compose ps
```

If there are no services (`nest-desktop` and `nest-simulator`) in the displayed list, it means that no containers can be started. You can attach a container for services without starting it using `--no-start`.

```
docker-compose up --no-start
```

Then start all services `nest-desktop` and `nest-simulator` as daemon.

```
docker-compose start
```

Stop all services, here `nest-desktop` and `nest-simulator`.

```
docker-compose stop
```

Shutdown all services, here `nest-desktop` and `nest-simulator`.

```
docker-compose down
```

## Set environments

### Custom port of NEST Simulator

For some reason the port 52425 is already occupied and thus starting the server instance of NEST Simulator might cause conflicts. To resolve this issue, you can change the port to 54321 for NEST Simulator server instance.

You have to change three lines:

- Set the environment `NEST_SIMULATOR_PORT`: 54321 in `nest-desktop` service.
- Set the environment `NEST_SERVER_PORT`: 54321 in `nest-simulator` service.
- Change the port binding to "54321:54321" in `nest-simulator` service.

An example configuration for docker-compose would be:

```
version: "3"

services:
  nest-desktop:
    image: docker-registry.ebrains.eu/nest/nest-desktop:3.2
    environment:
```

(continues on next page)

(continued from previous page)

```
NEST_SIMULATOR_PORT: 54321
ports:
  - "54286:54286"

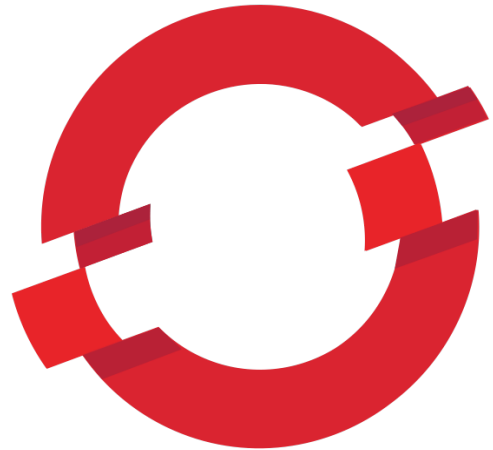
nest-simulator:
  image: docker-registry.ebrains.eu/nest/nest-simulator:3.4
  environment:
    NEST_CONTAINER_MODE: "nest-server"
    NEST_SERVER_PORT: 54321
  ports:
    - "54321:54321"
```

## Acknowledgments

Thanks for the help:

- Steffen Graber (Docker Hub for NEST Simulator)
- Jochen Martin Eppler (API Server for NEST Simulator)

## Deploy on OpenShift



This part of the documentation shows how to deploy NEST Desktop on OpenShift resources. In the following, we will use the deployment process of NEST Desktop on the OpenShift resources of EBRAINS as an example of practice.

### Requirements:

- OC Client Tools

## Deploy NEST Desktop on EBRAINS

EBRAINS provides two OKD infrastructures:

- <https://okd-dev.hbp.eu> for the development and
- <https://okd.hbp.eu> for the production.

---

**Note:** I strongly recommend to use the development page for testing.

---

## Register client for authentication on EBRAINS

To access to NEST Desktop on EBRAINS infrastructure, an authentication is requested. You find the codes on <https://github.com/nest-desktop/apache2-oidc>.

Here are the steps how to setup the authentication for NEST Desktop properly.

```
bash get-dev-token.sh
```

Change the configuration file and then create a client for your application.

```
bash create-client.sh
```

Keep `client_id` and `client_secret` for the **okd** infrastructure.

## Build NEST Desktop on EBRAINS

First, copy the command line from the web console of <https://okd-dev.hbp.eu> and enter in terminal to login via `oc`:

```
oc login https://okd-dev.hbp.eu:443 --token=<TOKEN>
```

Get the status of the current project:

```
oc status
```

You can find the configurations on <https://github.com/nest-desktop/nest-desktop-ebbrains>. Therein, you have to modify the environment for EBRAINS authentication, i.e. `OIDC_CLIENT_ID` and `OIDC_CLIENT_SECRET` of NEST Desktop (which is printed after setting up the client for NEST Desktop).

Execute the bash script to deploy the `nest-desktop`, `nest-server` and `apache2-oidc` containers:

```
bash setup-nest-desktop.sh
```

### Further usage

Scaling up the replicas (pods or nodes):

```
oc scale --replicas=2 dc nest-desktop
```

### Acknowledgements

Thanks for the help to integrate NEST Desktop on EBRAINS resources:

- Alberto Madonna (Conceptual design of the user authentication)
- Collin McMurtrie (Conceptual design of the user authentication)
- Fabrice Gaillard (Conceptual design of the user authentication)
- Jonathan Villemaire-Krajden (Conceptual design of the user authentication)
- Martin Jochen Eppler (For the contacts)

### Deploy on OpenStack

The guide provides a step-by-step documentation on how to deploy NEST Desktop on OpenStack resources. For more information on OpenStack, please follow this link: <https://www.redhat.com/en/topics/openstack>.

As an example of an OpenStack infrastructure, we show the deployment on bwCloud, which is assigned to the universities in Baden-Württemberg, Germany. For more information bwCloud, follow the link: <https://www.bw-cloud.org/>.

Deployers can build an OpenStack image via Packer and Ansible.

#### Requirements:

- [Packer](#)
- [Ansible \(2.3.2.0 or newer\)](#)

## Deploy NEST Desktop on bwCloud

You can find the source code on <https://github.com/nest-desktop/nest-desktop-bwCloud>.

1. Download the OpenStack RC File from [bwCloud dashboard](#):

Project -> API Access -> Download OpenStack RC File

2. Source the RC file to login:

```
source Project_<userID>-openrc.sh
```

3. Modify the Ansible configurations in `infrastructure/bwCloud/nest-desktop.json`.

Set `image_name`. Values for `source_image` and `networks` are taken from bwCloud dashboard.

4. Build an image on bwCloud:

```
packer build nest-desktop.json
```

5. Start an instance on the bwCloud dashboard and it will have a public IP of the virtual machine.

## Acknowledgements

Thanks for the help to integrate NEST Desktop on *bwCloud*:

- Bernd Wiebelt
- Jonathan Bauer
- Michael Janczyk
- Manuel Messner
- Christopher Ill

## 3.5.2 Deploy with external software

### Deploy NEST Desktop with Insite

The Insite system can be served as a used for NEST Desktop. It allows to visualize activity of the live simulation.

### How to setup NEST Desktop and Insite

First, get the configuration file for Docker Compose.

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop-docker/main/examples/  
↪insite/docker-compose.yml
```

For more information about Docker Compose, please read the corresponding [documentation](#).

Next, start all services of the Docker Compose file.

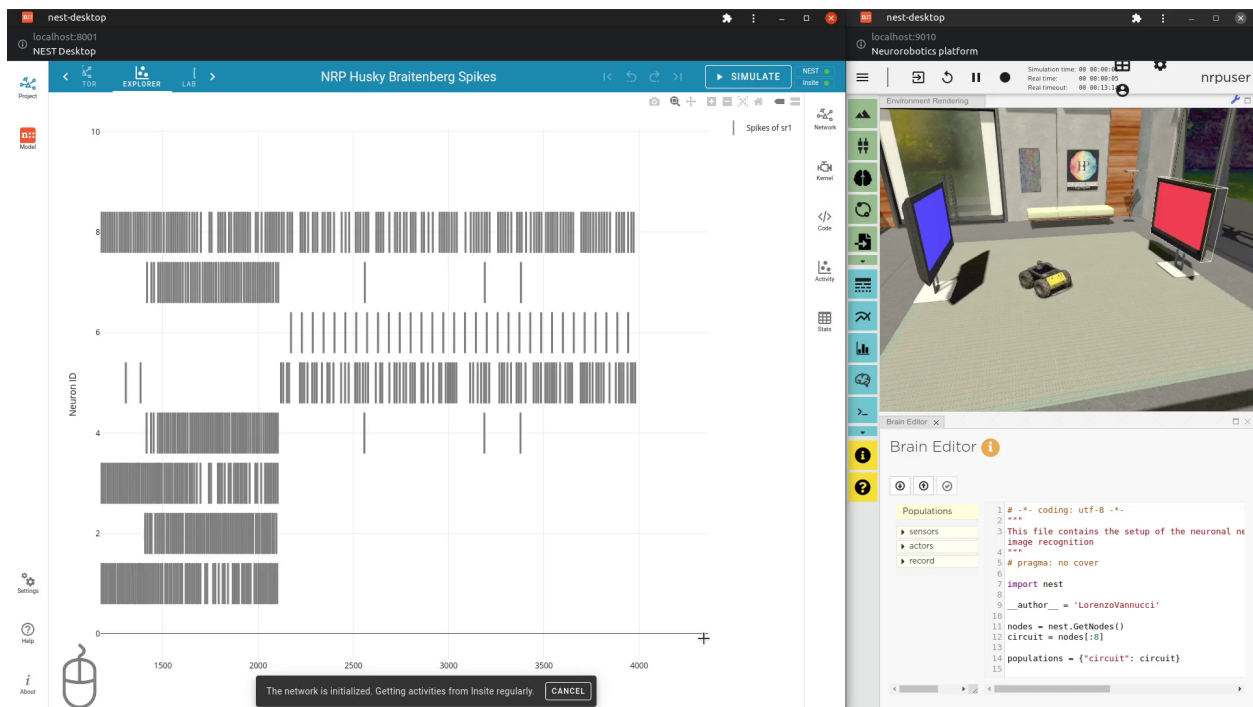
```
docker-compose up
```

NEST Desktop is now served at <http://localhost:54286>, whereas Insite NEST Module and Insite Access Node are served at <http://localhost:52425> and <http://localhost:52056>, respectively.

See also:

- [Simulate with Insite](#)

### Deploy NEST Desktop with NRP



The Insite system can be served as a backend for NEST Desktop. It allows to visualize activity of the live simulation.

## How to setup NEST Desktop and NRP

First, download the configuration file of Docker Compose from GitHub:

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop-docker/main/examples/  
↩️nrp/docker-compose.yml
```

Then start all services (*nest-desktop*, *insite-nest-module*, *insite-access-module*, *nrp-backend* and *nrp-frontend*):

```
docker-compose up
```

It takes a few minutes to pull all five docker images and start all containers.

### See also:

- Be sure that NEST Desktop runs two backends: NEST Simulator and Insite Access Node. For more information, please read *Simulate with Insite*.
- For the usage, please read *Use NESET Destkop with NRP*.

## 3.6 Developer guide

The developer guide provides more detail on how to develop NEST Desktop.

### 3.6.1 Get the source code

The source code of NEST Desktop is hosted on [GitHub](#). You can clone NEST Desktop from the GitHub repository:

```
git clone https://github.com/nest-desktop/nest-desktop  
cd nest-desktop
```

### 3.6.2 Development guideline

#### Coding conventions

Coding conventions help to generate good code. Therefore, we use some recommendations regarding the coding style. Most of them follow the conventions used by Git, Linux and [other projects](#). Some more central ones will be mentioned in the following.

### General coding conventions

- The TypeScript and ESLint options can be inspected in their respective config files.
- An `.editorconfig` for the basic settings is also available.
- In `.rst` files, the line length should not exceed 120 by much. This is of course not a fixed rule, but the `.editorconfig` seems *not to be able to guarantee this* and we have not found a convincing alternative. A bigger problem is that there are cases, where longer lines make sense (e.g. when they contain very long URLs or within bullet points, the latter with the possibility to use backslashes, which is sometimes quite cumbersome).
- The general coding conventions for Vue, TypeScript and for Python ( [PEP 8](#) ) should be followed.
- For `.ts` files, the type `any` should be used as rarely as possible (and might be removed in the future).
- The general recommendations for good (Vue) code should be applied, e.g. that Variables should be typed whenever possible.
- Coding and naming styles which are not explicitly mentioned here should be kept similar to the already existing code parts in that language.

### Git conventions

#### Commits

- Commits should have a title containing not more than 72 characters.
- The subject line of commit messages should be a short and informative summary of the pull request in the imperative (e.g. ‘Fix bug’ instead of ‘Fixed bug’) and should not end with a full stop.
- The body should focus on the *What* (the changes in comparison to the original version) and *Why*, very detailed explanations on the *How* should be included in the source code as comments.
- If the commit addresses an issue from the issue tracker, at least the issue ID should be mentioned in the commit body (or even in the name).
- This helps to generate release notes and to maintain a Git history where a `git log` produces helpful output.

#### Pull requests

- The rules for commits also apply here.
- Please have a look at the [GitHub keywords in issues and pull requests](#) .

### Naming conventions

- Names should be self-explanatory.
- We use camel case for custom `.class` names in *Vue* files.
- Kebab case is used for standard `.class` names in *Vue* files.
- The coding style overrides other capitalization rules (e.g. `ID` can become `Id`).



## Sphinx conventions

- For headings, we use the following items: = (with overline!) for parts and for sections: First-level: =, second level: -, third level: ^. Please note that all section headings should not have an overline!
- We usually capitalize only the first letter of a title (or heading), except programming expressions like class names, proper names, etc. We also recommend the Python documentation conventions, as suggested in [the official documentation](#).

## The semantic versioning

During the course of development, the implementation of (new) features in NEST Desktop will be reviewed for compatibility. In this concept a general rule of the semantic versioning NEST Desktop was introduced that the operational capability of the application can be guaranteed.

**Warning:** Please be aware of the differences to the official [Semantic versioning](#) standard!

The formal convention of the version releases for specifying compatibility in NEST Desktop uses a three-part number:

### A major number

It is incremented for the compatibility with the NEST Simulator. It implies that the major version of the NEST Desktop (2.x.x) has to match with the one of NEST Simulator (currently 2.x.x).

### A minor number

It is a breaking feature such as a new library or a minor changes of the data structure. It means that this version could cause the compatibility issues and the user might reset the data of the page.

### A patch number

It is a bugfix and non-breaking features were added to the code. The user is able to work with different patch versions of NEST Desktop and NEST Simulator.

## Concept of the interface

### General layout concept of the interface

NEST Desktop consists of three segments with different purposes.

The left column (1) shows the navigation to route pages. The center area (2) renders the main content of the page, whereas the right column (3) displays the controller for the modification of the content.

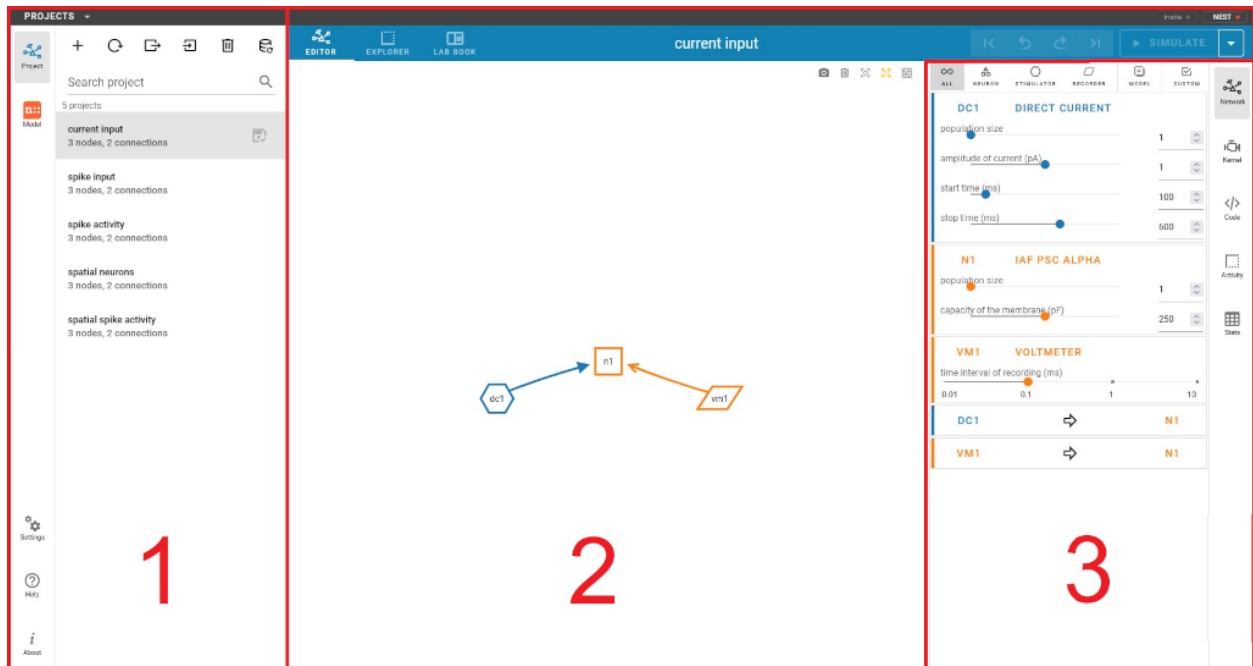


Fig. 1: The layout of NEST Desktop with the three main areas (1) - (3).

## Pages

NEST Desktop has three router views (Project, Model, Settings). The icons buttons on the left side navigate to these views.

## Page colors

The color code was taken from [Adobe](#). The colors of the pages are taken from the split complementary of the NEST default color (#ff6633).

## Navigation sidebar (left)

The navigation shows either projects or models.

## Router view (center)

The router view renders the page content via the URL. The project page displays a tab containing the network editor, the activity explorer and the lab book. The model page shows the model description which can be used in NEST Simulator. The setting page shows an overview of all settings for various components of the app.

### Controller sidebar (right)

The controller enables users to change values or configurations. The network controller displays a list of nodes and connections with their parameters.

## 3.6.3 Development stages

### Prepare the environment

NEST Desktop is written in *Vue.js* (a web framework written in TypeScript), and also in TypeScript. The Vue code is transpiled to HTML5 and JavaScript Code. There are multiple ways to develop Vue applications, but my preferred way (and probably the most common one) to develop NEST Desktop is to use *Node.js* (and optionally *Yarn*). Therefore, if you do not use any of the container systems mentioned below, you will need to [install Node.js](#) (for Windows, an easy installation guide can be found [here](#) ), which gives you also the possibility to install *Yarn*.

#### Requirements

- Node.js, Yarn
- NEST Simulator 3.0 or higher

You can install these requirements in the host system.

However, we prefer to use an Apptainer container and leave the host system unchanged. For this, we prepared a Apptainer recipe that builds a container with the required packages for the development.

### Build an environment with Apptainer

Get an Apptainer recipe:

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop-apptainer/master/
↪recipes/development/dev-node-16-alpine.def
```

The definition file `dev-node-16-alpine.def` contains an adequate environment to develop and build NEST Desktop.

Build an Apptainer image:

```
apptainer build dev-node-16-alpine.sif dev-node-16-alpine.def
```

Go to the shell inside the Apptainer container:

```
apptainer shell dev-node-16-alpine.sif
```

### Commands

Install node modules for NEST Desktop:

```
yarn install
```

Start a development server:

```
yarn serve
```

---

**Note:** The command `yarn serve` uses the configuration file `vue.config.js`. This file controls the threads used for the linting (the statical-syntactical code checks). With the default configuration, all available threads are used to minimize the build time. This might slow down other programs. There are cases where you cannot afford that and prefer a slightly longer execution time. In that cases, you can either adjust the number of threads in that file. This reduces the CPU load, but some CPU resources might stay unused. Alternatively you can execute the console in which you want to spawn the `yarn` command with a lower priority. On Linux (even on MacOS or Windows using WSL2 and an available shell command) this can be done using

```
nice -n 20 bash
```

This will spawn a new console inside your current console, but with the lowest processing priority possible, i.e. this console and its tasks do not block other tasks (like video conferences, etc.) significantly. Do not be confused that there will be no new window and no major visual cues that you are now in another process. In that console you can now execute the commands mentioned above.

---

### Useful commands

Check if any node modules are outdated:

```
yarn outdated
```

Upgrade outdated node modules:

```
yarn upgrade
```

### Work on the source code

First, prepare the development environment with the required packages.

```
yarn serve
```

The Live Development Server is now serving at `http://localhost:54286`.

---

**Note:** For more information on how to prepare the environment for the development, please check the [guide](#).

---

## Setup

It is possible to install NEST Desktop from source code on a local machine using `pip` (where it finds `setup.py`). The recommended method is to install it in the user's home directory using the command argument `--user`.

```
python3 -m pip install --user -e .
nest-desktop start
```

---

**Note:** Do not forget to start NEST Simulator.

---

## Commit changes

Go to the *dev* branch for the development.

```
git checkout dev
```

Fetch the data from GitHub (download it to your local directory):

```
git fetch
```

This command can be varied with options to e.g. fetch all branches (`git fetch --all`) or to discard unreachable content (`git fetch --prune`), even with multiple of them. If required, intergrate the changes from GitHub into your local repository:

```
git pull
```

It is recommended to create a new branch for an an implementation of a new feature/goal.

```
git checkout -b newBranch
```

If your changes are ready to be committed, stage and commit them:

```
git add ...
git commit -m 'This is my commit.'
```

### Push changes to GitHub

Finally, push all of them to repository on the internet (and create a merge request afterwards).

```
git push --set-upstream origin newBranch
```

A merge request will then be handled by the team: It will be reviewed and if it provides some nice additions, it will be merged.

---

**Note:** It is likely that the review contains some change requests which have to be addressed and committed by you before the merge can be made.

---

### Build and publish

Currently, we build NEST Desktop for multiple targets and publish them on various platforms.

---

**Note:** Please be aware that a lot of steps are already covered by our [GitLab CI process](#). Therefore, we recommend to inspect the `.gitlab-ci.yml` file together with this chapter. It might also be helpful to have a look at the commands defined in `package.json`.

---

## Python



Building and pushing NEST Desktop on [PyPI](#) is a required step for the production. After that, Docker Hub can upgrade NEST Desktop in the provided Docker image.

### Requirements

- `setuptools`, `wheel`, `twine`

The Python Package Index **nest-desktop** includes an executive command `nest-desktop` and a Python library `nest_desktop`.

## Build

The current working directory is `nest-desktop`.

The building phase contains two steps: First, build a package of NEST Desktop using `vue-cli-service`.

Initially, you have to upgrade the version of `nest-desktop` in:

- `packages.json`
- `nest_desktop/__init__.py`

Then generate the app package using `yarn`. It builds the folder `nest_desktop/app`:

```
yarn build
```

The second step is to build a pip package for PyPI:

```
rm -rf build/ dist/ nest_desktop.egg-info/
```

Then generate the distribution packages of *nest-desktop* for PyPI:

```
python3 setup.py sdist bdist_wheel
```

## Upload

Finally, the package is ready for the the publication. You can upload the pip-package of `nest-desktop` to PyPI:

```
python3 -m twine upload dist/*
```

Do not forget to commit the changes you made and set a new version tag in git.

```
git tag -a v3.0 -m 'v3.0.0'
git push --tags
```

### Conda



We have a [conda-smithy repository for nest-desktop](#). When a new Python package is released, we can change the version in `meta.yaml` (the meta content [online](#)):

```
{% set version = "3.x.y" %}
```

---

**Note:** It is also important to change the sha256 checksum of the source of `tar.gz` file.

---

Then make a pull request on the base branch of this repository.

### Electron (.deb package)

In `package.json`, there are also yarn commands configured to build an Electron app.

```
yarn electron:build
```

Then install the `.deb` file on your Linux system.

#### See also:

If you want to build other Electron packages, please have a look into `electron-builder.yml` file.

For Snap packages, you can find more information in the [Snap repository for nest-desktop](#).

### User documentation

We use reStructuredText for [Sphinx](#) to generate the documentation locally and online on [Read the Docs](#). To learn more about the syntax, check out [this quick reference](#). Please have a look at the [coding conventions](#), too.

#### Requirements

- [Sphinx](#)
- [Material Design Theme for Sphinx](#)

Use the working directory: `nest-desktop/docs`. To install Sphinx and the Read the Docs theme via `pip`:



```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop/main/docs/requirements.  
↪txt  
python3 -m pip install -r requirements.txt
```

### Development: Build HTML locally

Build the documentation which you created with Sphinx in the docs folder offline:

```
make clean; make html
```

Start the Python server to serve the documentation locally, i.e. available only on your personal machine.

```
python3 -m http.server --directory ./_build/html 8000
```

Then open the URL `http://localhost:8000` with your browser.

### Publication: Push to ReadTheDocs

The documentation files for the dev branch are automatically rebuilt (and updated) each time a push is made to the repository. The docs for other versions refer to the GitHub tags or branches. The latest tag is assigned to the latest release version.

### Optional: Use Apptainer

Build an Apptainer image file:

```
wget https://raw.githubusercontent.com/nest-desktop/nest-desktop-apptainer/master/  
↪recipes/development/doc-sphinx.def  
apptainer build doc-sphinx.sif doc-sphinx.def
```

Start the Apptainer container:

```
apptainer shell doc-sphinx.sif
```

Now you are in an Apptainer virtualization in which you can execute the `sphinx` command.

### Continuous integration (CI)

#### Mirror Action on GitHub

Since the NEST Desktop team has only restricted access to the CI resources of GitHub, the source code of NEST Desktop is mirrored (`.github/workflows/ebbrains-push.yml`) to the [EBRAINS GitLab](#), where we are able to use automated CI systems for the compilation and the deployment. We use only a small GitHub CI setup to transfer the code to the EBRAINS GitLab and execute the compute-intensive workloads there.

#### Jobs on GitLab

You can find the configuration in `.gitlab-ci.yml`. It consists of two stages, *build* and *deploy*.

In each stage, we prepared two parallel pipelines in which jobs will be executed when a specific branch is pushed:

- the development pipeline for the *dev* branch to check the functional operation of the CI (in the future with testing) and
- the production pipeline for the *main* branch when NEST Desktop is released.

In the build stage, the CI pipeline uses *Node.js* to produce NEST Desktop and to store it in the `nest_desktop/app` folder.

In the deploy stage, the CI deploys NEST Desktop as a Python package and as Docker images for [EBRAINS Harbor](#) (the Docker container registry of EBRAINS) and [Docker Hub](#). All jobs in the deploy stage depend on the job of the build stage being executed successfully.

Each executable job of the development and production pipelines has a base job, so that the job scripts in both cases seem to be identical, but they only differ in the version variable for the Python package and the Docker image.

---

**Note:** You can check these scripts also for commands if you want to execute single build steps manually on your machine.

---