NEST Desktop

Dec 13, 2021

Contents

CHAPTER 1

NEST Desktop

Hello there! :-)

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

The app enables the rapid construction, parametrization, and instrumentation of neuronal network models.

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

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

CHAPTER 2

Abstract

NEST Desktop - 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 for process for students/autodidacts and for teachers. Our new open source tool, NEST Desktop[1], 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[2].

A preprint for NEST Desktop is available on bioRxiv.

Acknowledgements

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).

Cite NEST Desktop 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 3.0.0. You can use the reference to the preprint for NEST Desktop (DOI: 10.1101/2021.06.15.444791) 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 bioRxiv.

[1] https://github.com/nest-desktop/nest-desktop

[2] https://ebrains.eu/service/nest-desktop

References

CHAPTER 3

History

2021

16 June Preprint on bioRxiv

03 - 07 May MSc course “Biophysics of Neurons and Networks” at [BCF], online (Freiburg, Germany).

08 - 09 April “NEST Desktop insitufication” on In-Situ Hackathon, online (HCI).

08 - 19 February BSc course “Simple Neuron Models” at [BCF], online, (Freiburg, Germany).

2020

30 Sep - 01 Oct Hand-on Session and Poster at Bernstein Conference 2020, online (Berlin, Germany).

18 - 22 July Tutorial with NESTML, presented by Dr. Linssen, at CNS 2020, online (Melbourne, Australia).

29 - 30 June Talk “NEST Desktop” at NEST Conference at [NMBU], online (As, Norway).

02 - 17 June MSc course “Biophysics of Neurons and Networks” at [BCF], online (Freiburg, Germany).

16 April Presentation and demo at NeuroMat, online (Sao Paulo, Brazil).

03 - 06 February Talk and Demo/Hand-on session at HBP Summit and Open Days , Athene, Greece.

2019

28 November 2nd HPAC Platform Training, Heidelberg, Germany.

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

18 - 20 September Poster/Live presentation at Bernstein Conference, Berlin, Germany.

22 July Talk and Tutorial/Hand-on session at [INM-6], Julich, Germany.

18 July NESTML/NEST-desktop integration workshop, [BCF], Freiburg, Germany.

24 - 25 June Talk and Tutorial/Hand-on session “NEST Desktop” at NEST Conference at [NMBU], As, Norway.

16 April Kick-Off workshop at [HCI], Trier, Germany.

25 - 31 March Tutorial workshop for IICCSSS at [BCF], Freiburg, Germany.

11 - 22 February BSc course “Simple Neuron Models” at [BCF], Freiburg, Germany.

2018

26 - 27 September Poster/Live presentation NEST Desktop at Bernstein Conference, Berlin, Germany.

27 - 28 August Technical meeting at [BCF], Freiburg, Germany.

25 - 26 June Talk “NEST Web API” at NEST Conference at [NMBU], As, Norway.

23 - 27 April MSc course “Biophysics of Neurons and Networks” at [BCF], Freiburg, Germany.

12 - 23 February BSc course “Simple Neuron Models” at [BCF], Freiburg, Germany.

2017

19 - 20 December Talk “NEST Desktop” at NEST Conference, Jülich, Germany.

20 - 22 November Live presentation with Ad Aertsen at Neural networks mini school, Strasbourg, France.

02 - 05 May MSc course “Biophysics of Neurons and Networks” at [BCF], Freiburg, Germany.

24 January Talk (Informal Seminar) “NEST Desktop” at [BCF], Freiburg, Germany.

2016

December

- The start of the development of NEST Desktop.

CHAPTER 4

Releases

Jun 23, 2021	v3.0.0
Mar 4, 2021	v2.5.1
Oct 23, 2020	v2.5.0
Jul 15, 2020	v2.4.1
Jun 28, 2020	v2.4.0
May 23, 2020	v2.3.2
May 22, 2020	v2.3.1
May 22, 2020	v2.3.0
Jan 27, 2020	v2.2.15
Jan 27, 2020	v2.2.14
Jan 20, 2020	v2.2.13
Jan 16, 2020	v2.2.12
Dec 30, 2019	v2.2.11
Dec 4, 2019	v2.2.10
Dec 4, 2019	v2.2.9
Dec 3, 2019	v2.2.8
Nov 27, 2019	v2.2.7
Nov 27, 2019	v2.2.6
Nov 27, 2019	v2.2.5
Nov 24, 2019	v2.2.4
Nov 24, 2019	v2.2.3
Nov 24, 2019	v2.2.2

Nov 21, 2019	v2.2.1
Nov 8, 2019	v2.2.0
Nov 5, 2019	v2.1.3
Nov 5, 2019	v2.1.2
Nov 4, 2019	v2.1.1
Oct 29, 2019	v2.1.0
Oct 2, 2019	v2.0.7
Sep 30, 2019	v2.0.6
Sep 25, 2019	v2.0.5
Sep 25, 2019	v2.0.4
Sep 23, 2019	v2.0.3
Sep 16, 2019	v2.0.2
Sep 15, 2019	v2.0.1
Sep 13, 2019	v2.0.0
Sep 13, 2019	v2.0.0
Jul 23, 2019	v1.5
May 31, 2019	v1.4
Apr 19, 2019	v1.3
Mar 18, 2019	v1.2
Dec 18, 2018	v1.0
Apr 22, 2018	v0.15.3
Apr 17, 2018	v0.15.1
Feb 28, 2018	v0.15.0
Feb 28, 2018	v0.14.0
Feb 7, 2018	v0.13.0
Jan 29, 2018	v0.12.0
Nov 22, 2017	v0.11.0
Oct 6, 2017	v0.10.0
Jun 20, 2017	v0.9.3
Jun 20, 2017	v0.9.2
Apr 29, 2017	v0.9.1
Apr 28, 2017	v0.9.0
Apr 18, 2017	v0.8.2
Apr 12, 2017	v0.8.1
Apr 10, 2017	v0.8.0
Mar 23, 2017	v0.7.2

Mar 15, 2017	v0.7.1
Mar 15, 2017	v0.7.0
Mar 4, 2017	v0.6.3
Feb 28, 2017	v0.6.2
Feb 27, 2017	v0.6.1
Feb 24, 2017	v0.6.0
Feb 9, 2017	v0.5.5
Feb 9, 2017	v0.5.4
Feb 7, 2017	v0.5.3
Feb 6, 2017	v0.5.2
Jan 23, 2017	v0.5.1
Jan 20, 2017	v0.5.0
Jan 9, 2017	v0.4.0
Jan 9, 2017	v0.3.12
Jan 5, 2017	v0.3.11
Jan 4, 2017	v0.3.10
Jan 4, 2017	v0.3.9
Jan 3, 2017	v0.3.8
Dec 29, 2016	v0.3.7
Dec 21, 2016	v0.3.6
Dec 21, 2016	v0.3.5
Dec 21, 2016	v0.3.4
Dec 20, 2016	v0.3.3
Dec 19, 2016	v0.3.1
Dec 19, 2016	v0.3.0
Dec 14, 2016	v0.2.1
Dec 14, 2016	v0.2.0
Dec 8, 2016	v0.1.0

CHAPTER 5

How to use NEST Desktop

Quick start

The video shows the few steps to construct a network and explore its activity. For more information, please read *detailed guides of usage*.

Getting started in Terminal with Docker-compose

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

For more information, please read the *detailed setup guides*.

Guides for NEST Desktop

- *How to setup NEST Desktop*
- *How to use NEST Desktop*
- *Troubleshooting*

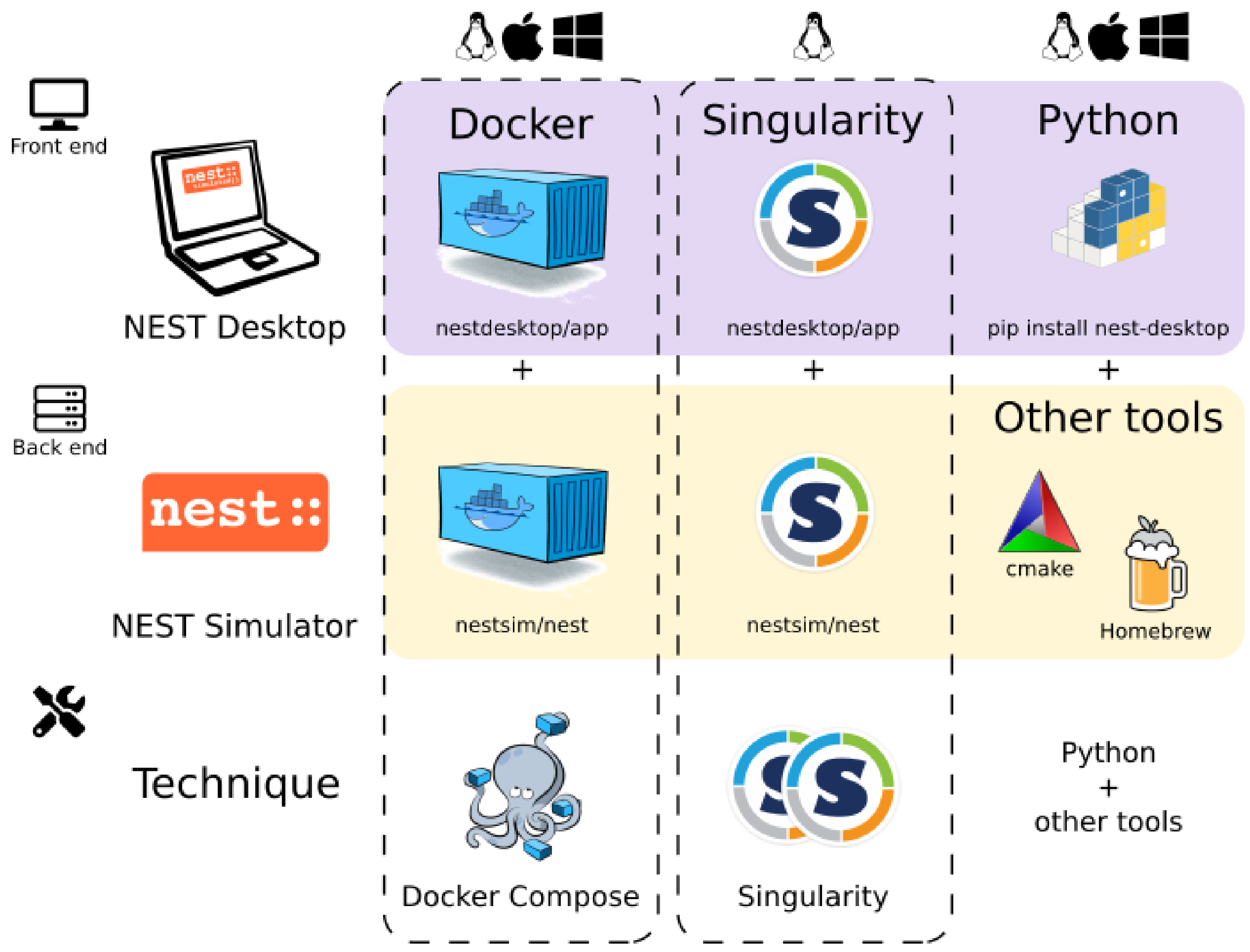
Guides for the infrastructure

- *Use NEST Desktop on EBRAINS*

CHAPTER 6

Setup Guide

This guide provides a detailed documentation on how to install NEST Desktop with NEST Simulator. The front end NEST Desktop requires NEST Simulator as the back end for the simulation. NEST Simulator has an API Server which can forward requests to the simulation engine. For more information, please have a look here.

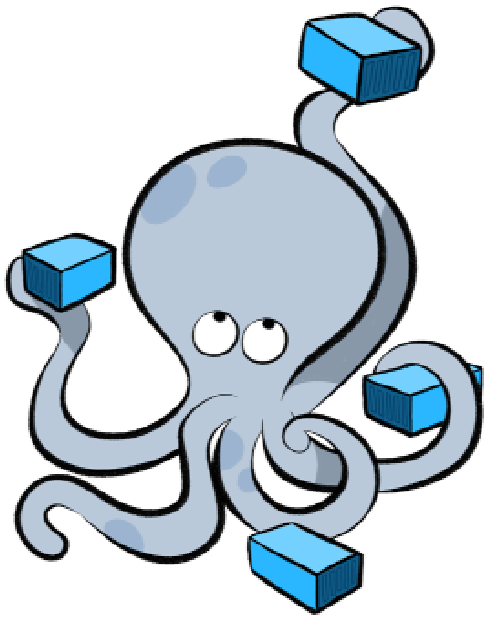


Docker (+ Docker Compose) and Singularity provide both NEST Desktop and NEST Simulator, so you have everything you need to run NEST Desktop.

Alternatively, you can install NEST Desktop with the pip command. 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. To enable the full functionality of NEST Desktop, you also need to install NEST Simulator on your computer and run NEST Server.

See instructions below for details.

6.1 Via Docker Compose



Docker is a virtualization software packaging applications and its dependencies. Docker Compose is a tool for running multi-container applications on Docker defined using the Compose file format. To get more information, see the official page of 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.

Please keep in mind that on Windows you can also use the Windows Subsystem for Linux (WSL) version 2 or higher, which allows you to run Docker without emulation. This offers performance advantages and more features. Detailed instructions on how to use Docker on WSL (version 2+) are provided by Docker and Microsoft.

Quick setup in Linux

1. Install Docker and Docker Compose

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

2. Get configuration file for Docker-compose (docker-compose.yml)

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

3. Build and start NEST Desktop and NEST Simulator with a single command:

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

Now NEST Desktop is started. You can use NEST Desktop in the web browser at http://localhost:8000.

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

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

6.2 Via Singularity

Singularity is an application container for **Linux** systems. For more information read the full documentation of Singularity here.

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

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

2. Register the bash command for NEST Desktop Singularity:

```
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.

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

```
nest-desktop-singularity 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-singularity* folder.

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

```
nest-desktop-singularity start
```

Now NEST Desktop is started. You can use NEST Desktop in the web browser at http://localhost:8000.

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

For more information read the full documentation of NEST Desktop Singularity.

6.3 Via Python Package

NEST Simulator cannot be installed via pip (maybe soon). Therefore, we need to install it in another way.

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 uwsgi
```

3. Start NEST Server as the back end:

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

```
nest-server start
```

Now, NEST Server is running at http://localhost:5000.

You can find the detailed information on NEST Server here.

4. 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 read the complete installing guide *here*.

5. 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:8000.

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

For more information read the full documentation of the command API *here*.

CHAPTER 7

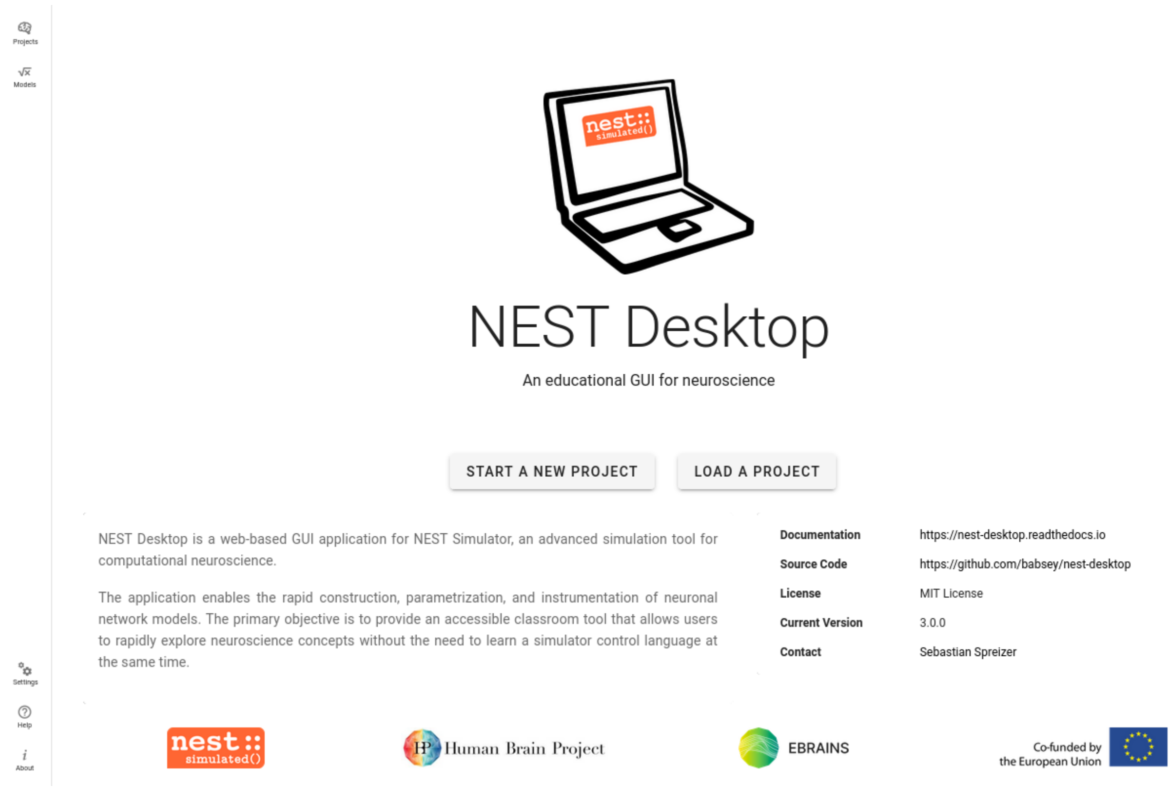
Usage Guide

This guides provides detailed documentation 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.

Getting Started

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

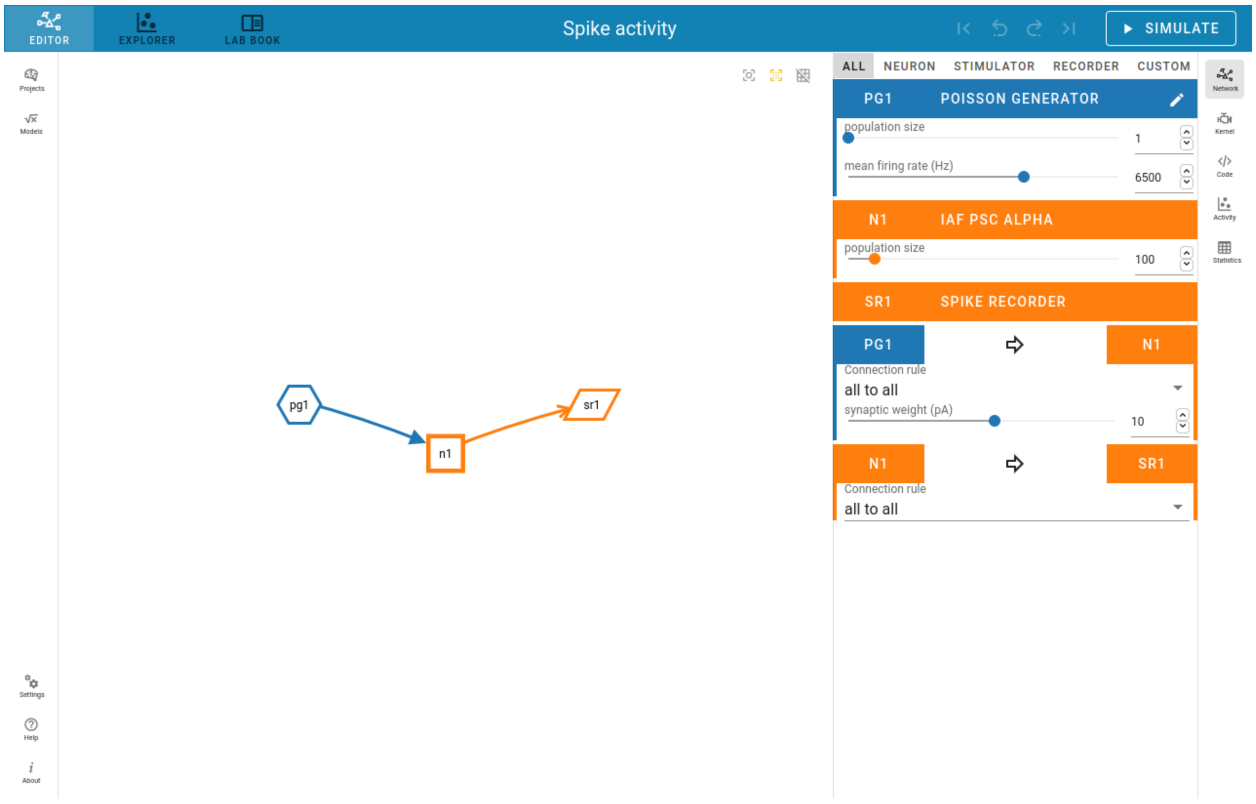
The application enables the rapid construction, parametrization, and instrumentation of neuronal network models. The primary objective is to provide an accessible classroom tool that allows users to rapidly explore neuroscience concepts without the need to learn a simulator control language at the same time.

Documentation	https://nest-desktop.readthedocs.io
Source Code	https://github.com/babsey/nest-desktop
License	MIT License
Current Version	3.0.0
Contact	Sebastian Spreizer

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

7.1 Construct neuronal networks

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



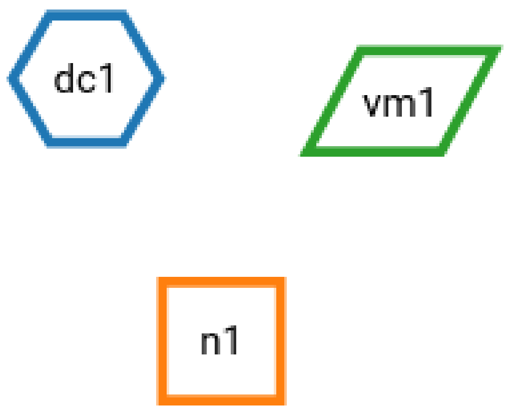
Here, we explain steps to create and connect nodes.

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.

Node labels

Each node graph is labeled to identify the model of the node. By default, it creates 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

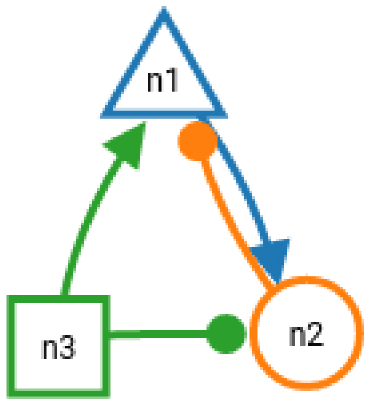
Nodes and connections contain parameter configurations which are displayed in the controller panel in the side navigation. The color of nodes helps you to associate the network graph with the controller as well as the corresponding visualization of the network activity. The color of lines is defined 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.
- **Square/Triangle/Circle:** 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.

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)

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 anther node and finally click on a target node. It creates a connection between source and target nodes.

Note: 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

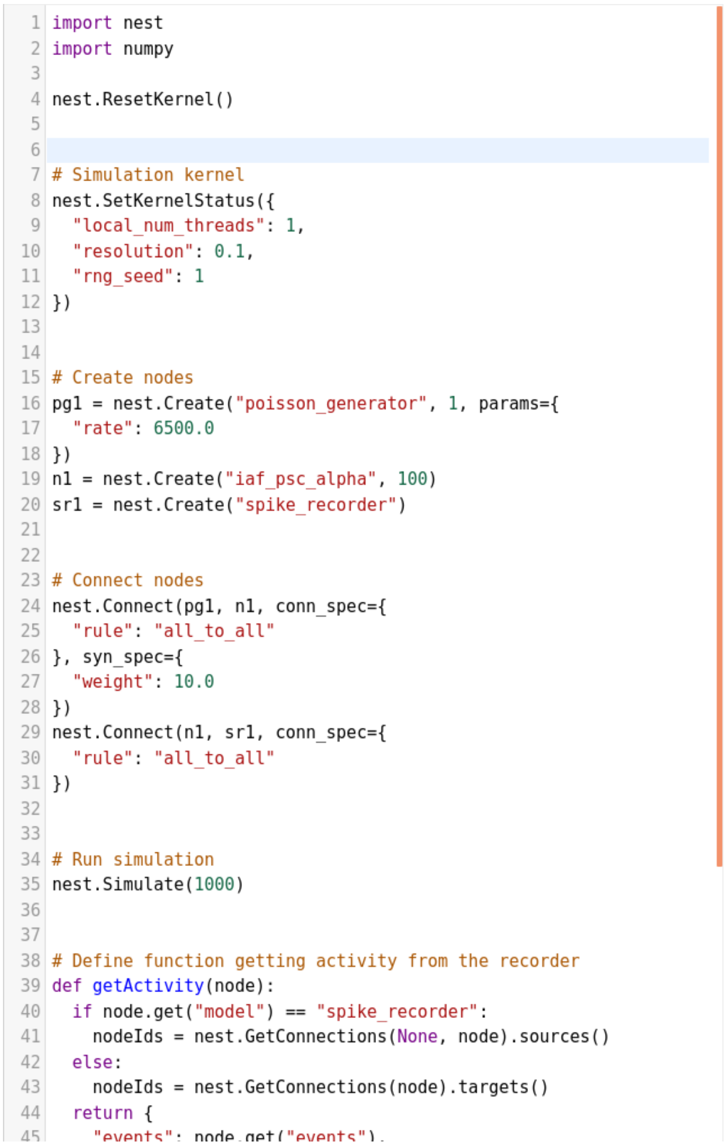
You are able to select the model of a node in the network controller. Then it shows a list of parameters which you might want to work on. Finally, you are able to change the values of visible parameters.

Network history

After every network changes, it automatic snapshot of the network it created and pushed to the network history list. With this network history you can undo or redo the network changes. Loading a snapshot from this history is called *checkout network*.

7.2 Simulate neuronal networks

You can click on the *Simulate* button to start the simulation of your network. In the code editor you can have an insight into the generated script code (see below for further information).



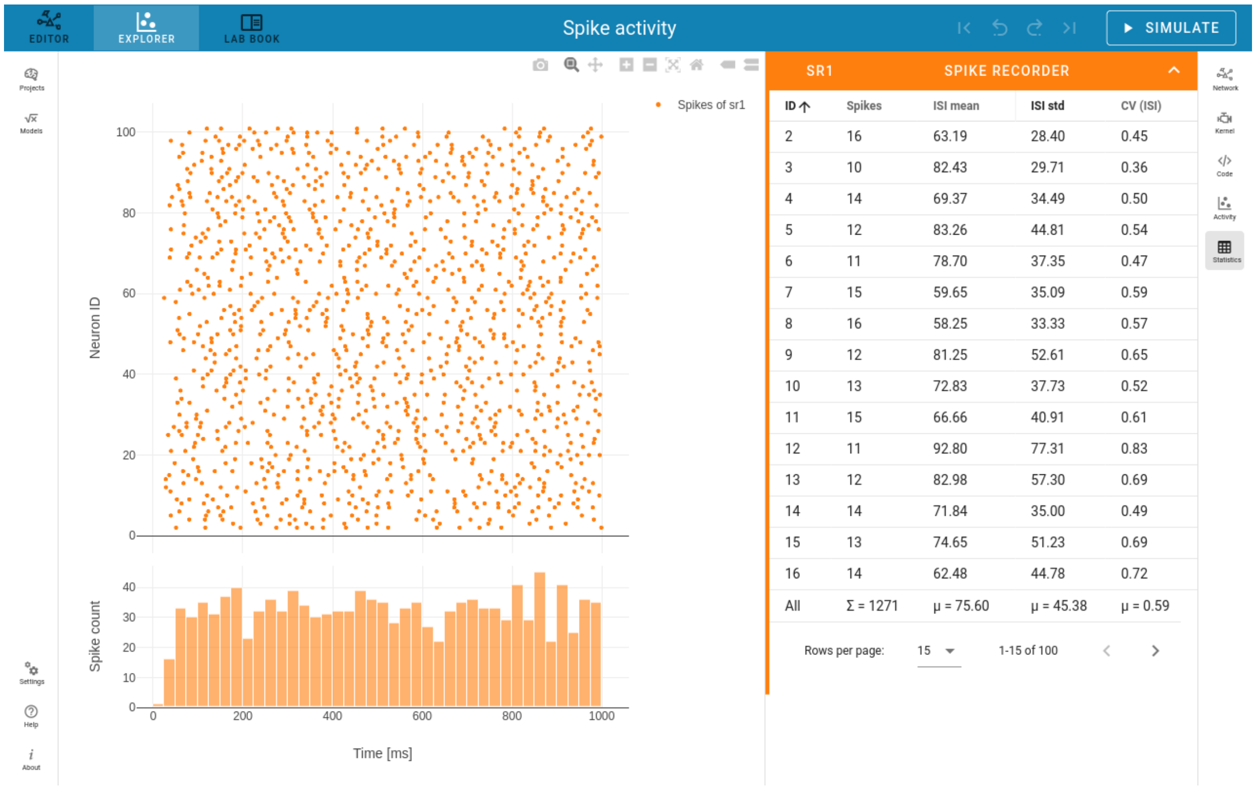
```python
import nest
import numpy

nest.ResetKernel()


# Simulation kernel
nest.SetKernelStatus({
  "local_num_threads": 1,
  "resolution": 0.1,
  "rng_seed": 1
})


# Create nodes
pg1 = nest.Create("poisson_generator", 1, params={
  "rate": 6500.0
})
n1 = nest.Create("iaf_psc_alpha", 100)
sr1 = nest.Create("spike_recorder")


# Connect nodes
nest.Connect(pg1, n1, conn_spec={
  "rule": "all_to_all"
}, syn_spec={
  "weight": 10.0
})
nest.Connect(n1, sr1, conn_spec={
  "rule": "all_to_all"
})


# Run simulation
nest.Simulate(1000)


# Define function getting activity from the recorder
def getActivity(node):
  if node.get("model") == "spike_recorder":
    nodeIds = nest.GetConnections(None, node).sources()
  else:
    nodeIds = nest.GetConnections(node).targets()
  return {
    "events": node.get("events"),
```

Code editor

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 easily to learn.

The graphical representatives of the nodes deliver arguments to the block of the `nest.Create(*)` function. Next, connections supply a specification for 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.

7.3 Explore network activity



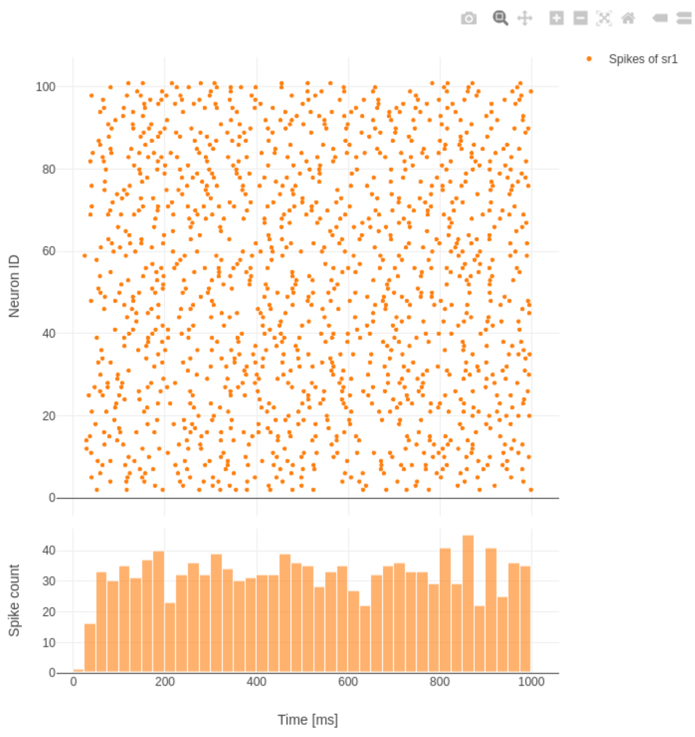
SR1 SPIKE RECORDER

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

Rows per page: 15 1-15 of 100

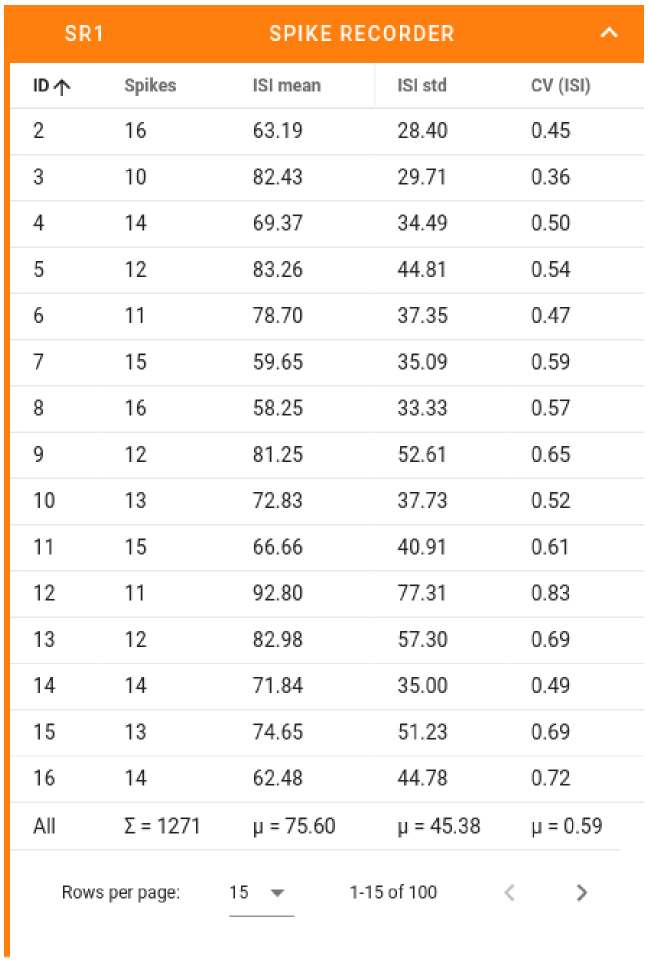
The network activity is composed of neuronal properties (positions and ids of neurons) and recorded events from recording devices. Events can be subdivided in two groups: spike events and analog signals. Spike events contain times and ids of the senders emitting events to the recording devices which can be considered as collectors (`spike detector`). 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 a graph or table.

Activity graph



It displays either a chart graph or an animated 3D graph for the spatial network forming layers in topology whose neurons have geographical positions. The chart graph contains graphical panels organized in vertical stacks. Chart panels are introduced specifically to explore the network activity by mouse interaction. It uses a scatter graph to visualize spike activity and a line graph to visualize analog signals. When you add one or more sub-panels for spike activity it shows a histogram graph of spike times or of inter-spike intervals.

Activity table



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

Rows per page: 15 1-15 of 100

You can go to the table by clicking on the *Activity statistics* button in the right side navigation. A table shows simple statistics of recorded elements (rows) of a node (population) connected to a recording device.

In spike events, the columns show the spike counts, mean and standard deviation of ISI (inter-spike interval) as well as CV_{ISI} (Coefficient of variation in inter-spike interval).

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

7.4 Manage projects

NEST Desktop has a project management helping you to organize your networks and network activity. If you want to explore the network activity of the project, you will have to start the simulation (see *Simulate neuronal networks*).

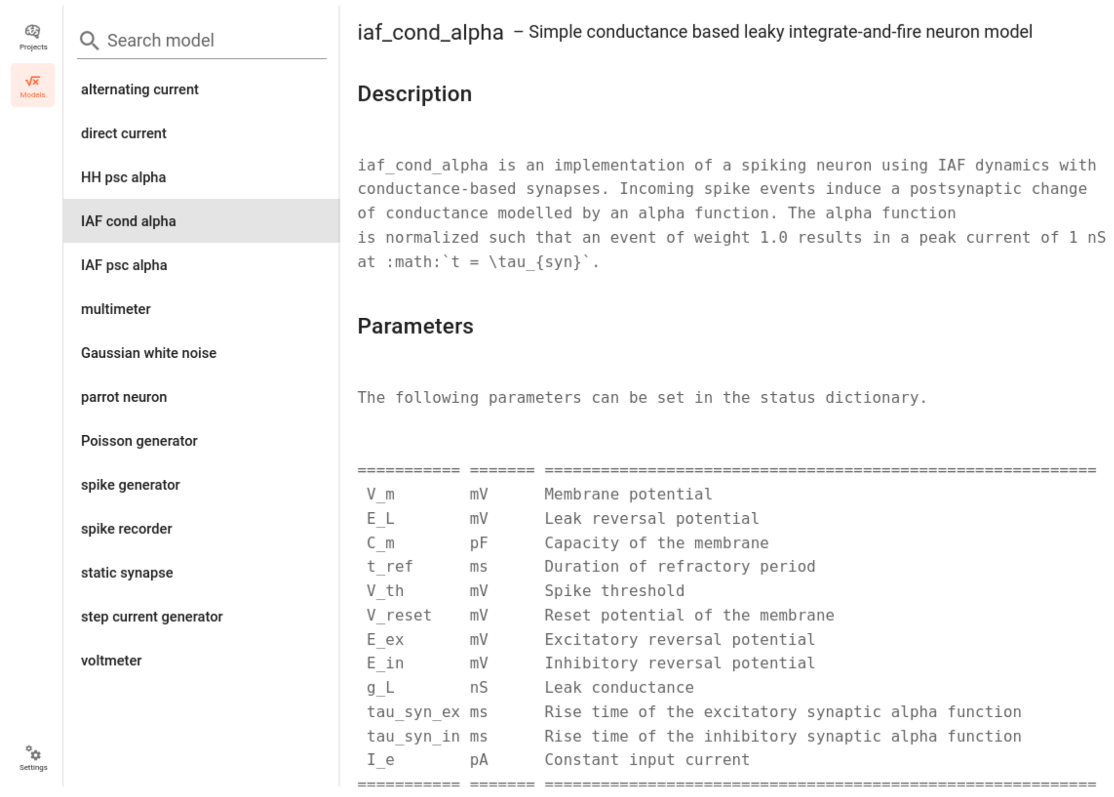
Clicking on *New project* creates a new project where you can construct a network from the scratch (see *Construct neuronal networks*). It is useful to give project a proper name so that you can recognize your projects.

Below the search field it shows a list of the projects. Clicking with right mouse button on a project item shows a menu with options to reload, duplicate, download or delete the project. In the projects menu, you can find methods to reload, download, upload, delete or reset projects.

Note: Unless you click on the save button, the project is not stored in database and is lost when you reload the page!

An important remark is that it stores only projects with neuronal networks in database, but all activity will be lost after page reload!

7.5 Explore neuron models and devices



iaf_cond_alpha – Simple conductance based leaky integrate-and-fire neuron model

Description

```
iaf_cond_alpha is an implementation of a spiking neuron using IAF dynamics with
conductance-based synapses. Incoming spike events induce a postsynaptic change
of conductance modelled by an alpha function. The alpha function
is normalized such that an event of weight 1.0 results in a peak current of 1 nS
at :math:`t = \tau_{syn}`.
```

Parameters

```
The following parameters can be set in the status dictionary.

=========== ======= ==========================================================
 V_m        mV      Membrane potential
 E_L        mV      Leak reversal potential
 C_m        pF      Capacity of the membrane
 t_ref      ms      Duration of refractory period
 V_th       mV      Spike threshold
 V_reset    mV      Reset potential of the membrane
 E_ex       mV      Excitatory reversal potential
 E_in       mV      Inhibitory reversal potential
 g_L        nS      Leak conductance
 tau_syn_ex ms      Rise time of the excitatory synaptic alpha function
 tau_syn_in ms      Rise time of the inhibitory synaptic alpha function
 I_e        pA      Constant input current
=========== ======= ==========================================================
```

The model page provides you detailed documentation of the models.

CHAPTER 8

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.

8.1 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. See the FAQ for NEST Simulator below.
- The server is down. Contact the server operators and enter the URL of the NEST Simulator, e.g. `localhost:5000` in a web browser to check if it is running again.

Note: For advanced users: Check in a terminal whether NEST Simulator is running (`curl localhost:5000`).

Internal server error

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

NEST error

NEST Simulator produces a value error, e.g. `The value cannot be negative.`. Please have a look at the official NEST documentation to obtain the correct syntax for the commads.

8.2 Frequently asked questions (FAQ)

NEST Simulator

How can I change the URL of the NEST Simulator? On the settings page you can find the URL of the NEST Simulator.

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

Project

How can I make a new project? In the navigation sidebar you will find a list item *New Project* to create a new project.

How can I duplicate a project? In the project toolbox you will find a button to clone project.

How can I rename a project? In the navigation sidebar you will find the input field of the project name. There you are able to change the name of the project.

How can I save a project? In the left navigation of the page you will find a floppy disc icon to save a project.

How can I delete a project? In the context menu of the projects (by clicking with the mouse right button on a project) you will find a method to delete a project.

How can I download a project? In the context menu of the projects (by clicking with the mouse right button on a project) you will find a method to download a project.

How can I delete multiple projects at once? In the context menu of the projects icon in the navigation side bar you will find a dialog to delete projects. Then select several projects and delete them.

How can I download multiple projects at once? In the context menu of the projects icon in the navigation side bar you will find a dialog to download projects. Then select several projects and download them.

How can I upload projects? In the context menu of the projects icon in navigation side bar you find a dialog to upload projects. Then select a file or URL, it shows a list of projects. Select wanted project and then upload them.

Where can I find the data for the project in JSON format? When the development mode is enabled, clicking on the button *{}* (on which the prominent brackets used in JSON are shown) you will find the JSON data of the current project.

Network

Where can I find the network controller? You will find the network controller by clicking on the icon (*Network*) in the right controller. Nodes and connections are stacked as card panels in the network controller.

How can I empty a network? In the network graph you will find top right a trash button that empties the network.

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

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.

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 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 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 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.

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 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 simulation is not implemented.

How can I activate "simulation after change"? In the context menu of the *SIMULATE* button (by clicking it with the right mouse right button) you will find an option to activate *simulation after change*.

How can I activate "simulation after load"? In the context menu of the *SIMULATE* button (by clicking it with the right mouse button) 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 context menu of the *SIMULATE* button (by clicking it with the right mouse button) 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.

Where can I set the simulation time? You will find the 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 resolution of the recorders are larger than the resolution in the kernel!

Where can I change the seed? You can find the seed value 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.

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.

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 the 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 the activity of projects.

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, simple 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 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 activity controller? The controller for the animated activity is placed in 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 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.

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.

How can I add a "trailing" effect for dots? It only works with the animation of the spikes.

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

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.

Settings

Where can I find the settings? You will find settings by clicking on the cog icon on the left in the navigation side bar. The settings are stored in the 'local storage' of the browser.

How can I change settings? You can change settings in the settings section by clicking on cog icon next to navigation side bar.

Where can I find the databases? The databases are stored as the 'Indexed DB' of the browser.

How can I switch to the development view? In the settings of the application you will find an option to switch to the development view.

CHAPTER 9

Use NEST Desktop on EBRAINS

EBRAINS is a platform infrastructure for neuroscience. NEST Desktop is available as a prototype online.

Note: You need an **EBRAINS** account to access NEST Desktop.

Trouble shootings

Why cannot I find NEST Simulator? Sometimes the issue is resolved when you check NEST Simulator.

If not, the problem lies in the cookies and site data for the user authentication of the *EBRAINS* platform. That cookie expires after a session. This happens when you re-access to `https://nest-desktop.apps.hbp.eu` after the browser is closed.

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

A persistent solution is to set the browser configuration so that it deletes cookies and site data when the browser is closed (It works in Firefox), but avoid to accidentally delete cookies you want to keep.

CHAPTER 10

How to lecture with NEST Desktop

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 (*How to use NEST Desktop*).

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 simulating brains*
- *How to prepare a course protocol*

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.

Select the level of the course that you plan to conduct with NEST Desktop:

Acknowledgments

Thanks for the help:

- Stefan Rotter (course instructor)
- Jeanine Laturner (tutor)

CHAPTER 11

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.

CHAPTER 12

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.

CHAPTER 13

First steps toward simulating brains

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 need to learn how to . . .

- select a neuron model
- construct a neuronal network model
- apply a stimulus to the network
- record the activity of neurons
- run a simulation
- analyze the activity data
- visualize the results
- document a simulation for the protocol

CHAPTER 14

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 most 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.

CHAPTER 15

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.

Topics

15.1 Direct current injection into single neurons

Intracellular recordings give access to the membrane potential of single neurons. If cells are studied *in vitro* in an acute slice preparation, the synaptic input, which a neuron normally receives under *in vivo* conditions, is then missing. It must be replaced by artificial input applied to the neuron through the electrode. One can use direct current (DC) of various amplitudes to systematically explore the response properties of a neuron. This represents, in fact, a simple and efficient method to characterize biological neurons in electrophysiological experiments. We will here apply it to model neurons as well.

You will use a simple linear integrate-and-fire (LIF) point neuron model to devise a single-neuron current injection experiment. In the LIF neuron model, spikes are generated each time the membrane potential reaches a predefined threshold. Spike waveforms are not modeled explicitly, and you recognize a spike only by the reset (strong downward deflection) of the membrane potential that follows a threshold crossing.

1. What is the membrane potential response for both negative and positive values of the applied current? You can measure the membrane potential by performing an intracellular recording, using a voltmeter. Try different amplitudes of the applied current and describe the phenomena you observe.
2. Explore how the membrane potential response depends on the biophysical neuron parameters. In particular, describe the influence of the time constant of the membrane, the spike threshold and the absolute refractory time on the membrane potential trajectories.

Video tutorial

15.2 Noise current injection into single neurons

To account for additional effects seen in recordings from real nerve cells, we add some *noise* current to the input mimicking spontaneous channel openings, or *synaptic bombardment* originating from a large pool of presynaptic neurons. If the fluctuations of a physical variable are due to the combined influence of a large number of small contributions, the concept of White Noise can provide an adequate descriptive model. In this idealized type of signal, all frequencies are present in equal proportions, but with random phases (*wind*). A well-known example is *white* light, which comprises all colors from the visible spectrum. In our simulations, we use a sequence of independent identically distributed Gaussian random variables to approximate a particular type of white noise, Gaussian White Noise (GWN).

1. First look at the effect of noise in the subthreshold case. While you explore the membrane potential response for (weak) noise current input, note that GWN, like any Gaussian random variable, has two parameters: mean and variance. Perform systematic simulations with different combinations of mean and variance (or standard deviation) and find a way to document the results.
2. Now, consider a spiking neuron with "strong" noise current as input. If the strength of the input (mean), or the amplitude fluctuations of the input (variance), are large enough, the firing threshold is crossed and action potentials are generated. In contrast to the case of DC input, the fluctuations of the membrane potential will now affect the timing of the action potentials to some extent. Document this phenomenon with a suitable set of experiments, where you run the simulations with different seeds of the random number generator.
3. Play with the parameters of the noise current over a certain range of values when there is

   (i) no action potential generated, and

   (ii) several spikes generated.

   What exactly is the "threshold" now? To judge the effect of random fluctuations, it is important to look at multiple repetitions of the same experiment. What happens to the frequency of spikes and the irregularity of spike trains as mean and/or variance of the noise is increased? The irregularity of neuronal spiking can be assessed, for example, by the coefficient of variation (CV) of the interspike intervals.
4. Now you should systematically measure two types of input-output curves of the neuron:

   (i) Keep the variance of the noise at a fixed level and systematically change the mean of the noise. What is the difference to the curve you obtained with pure DC input?

   (ii) Now keep the mean of the noise at a fixed level and systematically change the variance of the noise. What is the minimal variance ("threshold") that leads to a non-zero response rate?

15.3 Excitatory and inhibitory synaptic input into single neurons

In a biological neuronal network, communication between neurons typically relies on synaptic input. Whenever the presynaptic neuron generates a spike, a chemical neurotransmitter (e.g. glutamate or GABA) is released and bound to postsynaptic receptors. This leads to a transient activation of the synapse, and a transient inward or outward postsynaptic current (PSC). As a consequence, the membrane potential of the postsynaptic neuron experiences a small deflection, the postsynaptic potential (PSP). Typically, for an excitatory synapse using glutamate as a transmitter, this deflection is depolarizing (towards threshold). In contrast, for an inhibitory synapse using GABA as a transmitter, it is hyperpolarizing (away from threshold). The superposition of many PSCs of either polarity represents the effective input to a neuron, which may, or may not, lead to an output spike to be sent to other neurons in the same circuit. You should now explore all these aspects by performing the following numerical experiments:

1. Devise a simulation method to study single postsynaptic potentials. Which simulation parameter reflects the "strength" of a synapse? Systematically explore the effect of activating single excitatory and inhibitory synapses. Under which conditions can activating a single synapse elicit an output spike?

2. Multiple PSPs elicited in rapid succession at one and the same synapse add up to a compound signal if they sufficiently overlap in time. This phenomenon is called "temporal integration". Multiple PSPs elicited at different synapses are also superimposed, all contributing to the membrane potential of the postsynaptic neuron. This is called "spatial integration". Design a set of experiments to illustrate the phenomena of temporal and spatial synaptic integration in the subthreshold regime. Which parameters of the neuron are mainly responsible for the temporal overlap between individual PSPs?
3. A hallmark of the LIF model, which is shared by many biological neurons, is the linearity of temporal and spatial integration. The membrane potential response to a combined input is just the sum of the individual responses to the individual inputs, as long as all of them remain subthreshold. Design an experiment that demonstrates the linearity of synaptic integration for the LIF neuron model. Does linearity also hold for superthreshold inputs that lead to action potential firing?

NEST Desktop allows you to study how neurons are activated by synaptic input. You can use the so-called `spike_generator` for experiments with maximal control. In essence, you specify the time point of each spike explicitly. Don't forget to specify the amplitude of the post-synaptic potential. This way, you can explore the effect of "spatio-temporal integration". You can also study, under which conditions synaptic input can trigger an output spike.

Video tutorial

15.4 Poisson input into single neurons

In models, we often assume that presynaptic action potentials are arriving randomly at a certain rate. Synapses on the dendrite of the postsynaptic neuron are then activated in a random fashion. The stochastic process that reflects such random arrival of point-like events is called a Poisson process. Its only parameter is the rate at which events occur. In view of the linearity of temporal and spatial dendritic integration, all synapses that have the same amplitude and sign can be treated as a single compound source of input. You can now employ the Poisson generator functionality of NEST to explore the response behavior of neurons:

1. Devise a method to display simulated Poisson spike trains, which will later be used as a source of input to our model neuron. Verify the randomness of the Poisson generator by repeating the same simulation multiple times. A raster display with lines corresponding to "trials" rather than "neurons" represents an adequate tool to illustrate this.
2. Consider an LIF neuron that receives Poisson input of a constant rate using a synapse of a specific amplitude. Analyze how the input rate influences the membrane potential and the spiking response of the neuron. The parameters of interest are the mean and the variance of the membrane potential, as well as the output firing rate and the irregularity of the output spike train. What happens if you change the strength of the synapse?
3. Now we consider the more realistic situation that a neuron receives input from two different and independent presynaptic populations, one consisting of excitatory, the other one consisting of inhibitory neurons.

   Note: The presynaptic population of a cortical nerve cell can be quite large, comprising up to 10 000 neurons, say.

   What matters for the postsynaptic neuron is the accumulated spike rate for each type of input, so these input rates will also be large. The model has two parameters, the rate λ_E of the excitatory Poisson process and the rate λ_I of the inhibitory Poisson process. Begin your simulation experiments by fixing $\lambda_E = \lambda_I$ assuming exactly the same firing rate for excitatory and inhibitory inputs. Start with small rates (subthreshold) and jointly increase them step by step until output spikes are generated (superthreshold). Describe your observations for weak and for strong input, both on the level of the membrane potential and on the level of output spike trains.
4. Considering synaptic bombardment from a large pool of presynaptic neurons, the mathematical model of shot-noise is appropriate to describe membrane potential fluctuations. Generally, the two relevant parameters λ_E and λ_I are fixed independently, and combinations with $\lambda_E \neq \lambda_I$ may arise. Previously, we have considered

Gaussian White Noise input, which was described by the two parameters mean μ and variance σ^2 . No specific assumptions were then made about the biophysical origin of membrane potential fluctuations. Shotnoise can also be described in terms of the mean μ and variance σ^2 of the membrane potential. As long as the input remains subthreshold and no output spikes are generated, it holds that $\mu \sim \lambda_E - \lambda_I$ and $\sigma^2 \sim \lambda_E + \lambda_I$. (The symbol $\sim$ means "is proportional to".) Perform some experiments that illustrate this relation.

5. If input rates are large enough, output spikes are generated. There is a loose correspondence between the mean membrane potential and the mean output rate, as well as between the membrane potential fluctuations (variance) and the variability of output spike trains (irregularity). One speaks of the "mean-driven regime" and the "fluctuation-driven regime", depending on whether spikes are predominantly generated by a depolarizing drive (mean), or by membrane potential fluctuations (variance), respectively. Explore the meaning of these two terms, and illustrate the two regimes by suitable simulations. Develop criteria that allow you to classify neuronal activity recorded in experiments accordingly.

NEST Desktop does not only offer direct current (DC) stimulators, but also noise current stimulators. In principle, they are used in the same way as a DC stimulator. "Poisson input" is just one specific form of "noise input". Technically, this is correct, and this immediately explains how to use it in a simulation: Just replace the direct current stimulator by a Poisson stimulator.

Biologically, however, we are now talking about an input that works with spikes that activate synapses, and it does not just inject electrical charge into the cell. Therefore, changing the properties of synapses on the target neuron also changes the properties of the "noise" input. This may be confusing, but the concept of "shotnoise" exactly reflects this. You should also keep in mind that the spikes generated by a Poisson source typically originate from a large set of presynaptic neurons. In the neocortex, this set could comprise hundreds or thousands of presynaptic neurons, and the rate parameter can assume very large values (rate of individual neurons number of presynaptic neurons).

Noise and fluctuations in our simulations are based on so-called "pseudo-random" numbers. They look like "true" random numbers for all practical purposes, but they are generated by a perfectly deterministic algorithm, one after the other. Using the same starting point (`seed`), you get exactly the same stream of random numbers. However, if you want a different stream of random numbers each time you perform the simulation, select *Randomize seed* in the "Simulation" controller.

CHAPTER 16

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.

Topics

16.1 The Hodgkin-Huxley theory of the action potential

16.1.1 Current clamp simulation of the free-running membrane

The leaky integrate-and-fire neuron model features a very simple mechanism of action potential generation: a fixed threshold on the membrane potential. The upstroke of the spike is not part of the model at all, whereas the voltage reset and the absolute refractory time following each spike are imposed on the membrane potential trajectories. To address the biophysical underpinnings of action potential generation and spike aftereffects, however, a different model must be considered. It turns out that specific membrane components, so-called voltage-dependent ion channels, are responsible for spike generation. The Hodgkin-Huxley model of the squid giant axon (published in 1952) accounts for these additional components. It explains why action potentials dynamically arise, and which membrane parameters and other circumstantial factors matter during this process. You will now perform simulation experiments on Hodgkin-Huxley (HH) neurons and this way characterize this model:

1. Verify that the subthreshold properties of the HH neuron model are similar to the properties of the LIF neuron model. To address this issue, inject a depolarizing or hyperpolarizing current into a LIF neuron and into a HH neuron and perform intracellular recordings to document the membrane response in both cases. Make sure the current is weak enough to not elicit a spike.
2. For strong-enough DC input current the HH model neuron will fire a train of action potentials. Inspect the spike waveform carefully and relate it to the spikes generated in a LIF neuron. The following keywords might guide your discussion: threshold, upstroke, downstroke, spike width, voltage reset, after-hyperpolarization, absolute refractory time, relative refractory time. Is the spike waveform of the HH neuron really the same for different input scenarios (e.g. weak vs. strong current)?
3. Use a spike recorder to characterize the spiking response to superthreshold current input. The goal is again to characterize the neuron by a curve that depicts the firing rate response as a function of the applied current.

NEST Desktop offers different neuron models. Here you should select `hh_psc_alpha` and compare its behavior to the simpler `iaf_psc_alpha` model that we have studied previously. As these two models have an entirely different spike generation mechanism, any intracellular recording of the membrane potential will look very different in both models. In the LIF model, spikes are just threshold crossings. In the HH model, in contrast, their waveform is explicitly simulated.

16.1.2 Sodium and potassium currents under current clamp conditions

It is the joint action of both voltage-dependent sodium (Na^+) channels and voltage-dependent potassium (K^+) channels which underlies action potential generation in excitable membranes.

In particular, sodium channel activation (i.e. opening) due to some initial depolarization leads to an influx of sodium ions, which depolarizes the membrane even further and opens even more sodium channels. This self-amplifying avalanche of sodium channel activation drives the action potential to a point of no return, from where it continues until the sodium reversal potential is reached. Sodium channel inactivation and potassium channel activation finally terminate the avalanche and repolarize the membrane. The ongoing outflux of potassium ions through (non-inactivating) potassium channels eventually hyperpolarizes the membrane towards the reversal potential of potassium.

1. Action potential generation based on voltage-dependent ion channels takes place in several stages, as described above verbally. Use the membrane potential recording of a single spike to illustrate the dissection of this process. Which of the contributing factors determines the temporal width of the spike? Can one record the underlying ion-specific membrane currents in biological neurons?
2. It is difficult to disentangle the effect of sodium channel inactivation and potassium channel activation. The former helps terminating the upstroke, the latter causes a downstroke of the membrane potential during a spike. In a simulation, you can record the concentration of `h`, `m` and `n` particles of the Hodgkin-Huxley model for a more precise view on these two processes. Is such a recording also possible in a biological neuron?
3. Explore the effect of the specific neurotoxins TTX and TEA on neuronal spiking. You can easily achieve this in your simulations by setting the peak conductances of either sodium channels or potassium channels, respectively, to zero. What happens if you only partially knock-out these channels by setting them to a non-zero, but reduced value? Formulate and explain your expectations before you perform the experiments.

NEST Desktop allows you to record the activation of `h`, `m` and `n` particles directly, using a `multimeter`. In a biological neuron, this would never be possible. To assess their dynamics, multiple separate experiments have to be performed.

16.1.3 Exploring the "spike threshold" and the "depolarization block"

Action potential generation follows an "all-or-nothing" principle. Whenever favorable conditions are met, a stereotyped waveform is kick-started and cannot be stopped any more. But what exactly are those "favorable" conditions? You should perform some numerical simulations to address this question:

1. Identify the "spike threshold" of a HH neuron based on the input-output function you generated above. Zoom into the threshold regime by performing more simulations just below and just above threshold. Does the HH neuron exhibit "type I" behavior or "type II" behavior?
2. Use now brief current steps to induce single spikes only. The two parameters of interest are the width of the current pulse, and its amplitude. In a nutshell, brief steps need to have a high amplitude, longer steps can be somewhat weaker to successfully generate a spike. Explore the threshold behavior with regard to both parameters, along the lines explained in the lecture.
3. You can use a continuous current ramp, as opposed to a sharp current step, to stimulate the neuron. If the rise in amplitude is shallow enough, it is possible to depolarize the membrane beyond any threshold voltage. No spike is generated then, and this phenomenon is called the "depolarization block". Try out different slopes of

the ramp. Can you explain it in terms of sodium channel inactivation? Perform a simultaneous recording of `h`, `m` and `n` particles to back up your explanation.

16.2 Point neuron models with conductance-based synapses

The term "point neuron" refers to a nerve cell, in which spatial variations of the membrane voltage are negligible. Such cells are then fully described by a single membrane potential variable. Most small neurons share the property of being "electrotonically compact" in this sense. Spatially extended cells, in contrast, must be conceived as a physical cable. In such a cable, the voltage is attenuated with distance due to leakage of electric charges. Moreover, the cable itself acts as a spatio-temporal filter and alters the signal transmitted by it. Such a cable is often approximated by a chain of electrically coupled single compartments, leading to so-called multi-compartment neuron models.

16.2.1 Point neurons with conductance-based synapses

In contrast to what we assumed for the leaky integrate-and-fire neuron model, biological synapses operate on the basis of ion channels, and should not be considered as fixed current sources. When a neurotransmitter is released at the presynaptic axon terminal, it diffuses through the synaptic cleft and binds to the receptors sitting in the membrane of the postsynaptic neuron. As a result, ion channels linked to the receptors open transiently and enable a brief postsynaptic current flow. The duration of this transient inflow or outflow is described by the so-called synaptic time constant. Depending on the concentration gradient of the ions involved (Na^+ for excitatory synapses, Cl^- for inhibitory synapses), this current is either depolarizing or hyperpolarizing, respectively.

If many of these synaptic channels are simultaneously open due to massive synaptic bombardment from a pool of presynaptic neurons, this may change the integration properties of the neuron as a whole. Its total input resistance R is decreased, and therefore the membrane time constant $\tau = R \cdot C$ is decreased as well, affecting the width of individual postsynaptic potentials (PSPs), which may also have a reduced amplitude due to the strong membrane leak. In addition, postsynaptic potentials have a shorter duration, as the membrane time constant is smaller. This scenario has been described as the "high-conductance state". Neurons then assume nonlinear properties, as their input integration becomes state-dependent: PSPs at rest are different to PSPs sitting on a background of many activated synapses.

1. Consider an isolated, conductance-based point neuron, with synapses that have alpha-functions as post-synaptic current transients, and a non-zero synaptic time constant. Simulate an individual excitatory postsynaptic potential, and explore its dependency on the driving force (distance to the excitatory reversal potential). This can be achieved by injecting additional subthreshold depolarizing currents of different strengths. Perform the same experiment for an inhibitory postsynaptic potential and discuss the differences to the case of excitation.
2. Now replace the DC input by an equivalent synaptic input, called synaptic background activity. It can be conveniently provided by a Poisson source, which is coupled to the neuron by an excitatory synapse. By systematic experimentation, you can now determine which firing rate of the Poisson source leads to the same mean depolarization of the postsynaptic neuron as a given DC injection. On top of the background activity, simulate again an individual excitatory postsynaptic potential. Describe how it changes its shape (amplitude and width) in the high-conductance state.
3. You can emphasize the high-conductance state even more, if you apply a combined excitatory and inhibitory Poisson input. In order to arrive at the same mean depolarization, the inhibitory input must be overcome by some extra excitatory input. The excessive excitatory and inhibitory synaptic bombardment, however, will reduce the effective input resistance and time constant even more.
4. Consider now synapses with different synaptic time constants. The reduction of synaptic strength in the high-conductance state is more prominent for slow synapses (large synaptic time constant) than it is for fast synapses (small synaptic time constant). Perform an experiment that demonstrates this surprising effect. You might just repeat the experiment from the second assignment with a different synapse that has a smaller or larger synaptic

time constant. Then you compare the attenuation of synaptic transmission due to background activity in both cases.

NEST Desktop enables strategies of analysis that cannot easily be adopted in a biological experiment. In order to display weak effects under noisy conditions, researchers have to perform many repetitions in several recording sessions and compute the average outcome afterwards. This is very time consuming. In a simulation, in contrast, you might consider performing simultaneous recordings from several neurons at the same time and directly compute the average. This is easily achieved in a simulation by setting the population size to a large enough value.

16.3 Network dynamics

In the central nervous system, neurons never act in isolation. Rather, they are bound to communicate with other neurons using both electrical and chemical signals. Fast electric signaling mostly relies on specific and precise synaptic transmission based on neurotransmitters. Depending on the transmitter system the presynaptic neuron is using, synapses come in different flavors: Synaptic communication is either excitatory or inhibitory. Although polarity is the most salient property, other parameters of synaptic transmission are also important, depending on the type of information that is being processed. We mention here the strength of the synapse, the transmission delay, the rise time of the postsynaptic potential, and different aspects of synaptic plasticity. Synapses are highly important building blocks of networks, determining the properties and the function of brains in essential ways.

A typical task of behavioral control rarely involves just one step of signal transduction. In most cases, several processing stages are needed. To achieve a complex task (e.g. produce spoken language) many neurons at different places in the brain eventually make their contribution. There is little agreement among researchers, however, how the communication is organized on the system level. In fact, different parts of the brain seem to employ very different strategies of collective signal processing. This is at least what the microanatomy of synaptic connectivity in brain circuits suggests. Whereas the cerebellum has a clear feed-forward architecture, neuronal communication in the large recurrent networks of the neocortex is dominated by feedback. It is really a daunting task to characterize these networks of different types of neurons in the brain to help improving our understanding of their role for the control of behavior. Numerical simulations of prototypic circuits help us exploring and refining theoretical ideas, and aligning them with biological design principles.

16.3.1 Recurrent networks of excitatory neurons

The most numerous cell type in the neocortex are pyramidal neurons (approximately 80%). They generally use glutamate as a transmitter and are, therefore, excitatory. Local circuits are highly recurrent, and there is intense synaptic communication among neurons. External inputs make an important contribution as well. In fact, each neuron receives a large number of inputs from other neurons, and contributes a large number of outputs to other neurons in the same network. Individual synapses are rather weak, though. Only the joint action of many excitatory inputs can bring the postsynaptic neuron to fire a spike. The synaptic connectivity of the network, to the degree to which it is known at all, is statistically consistent with the topology of a random graph. A typical value for the connection probability in local cortical networks is 10%.

1. Set up a large-enough population of excitatory neurons, based on the standard LIF model. Establish random synaptic connections among neurons.

   Note: There are several different options to wire up a network randomly (see https://www.nest-simulator.org/connection-management).

   Find a meaningful way to monitor neuronal activity in the network, both on the level of individual neurons (membrane potentials and spike trains), and on the level of the whole population (PSTH, as a proxy for the EEG). Characterize the type of activity that you observe.

2. Explore now the role of external input for the dynamics of the recurrent network. As this external input is normally provided by other neurons that are not part of the local network in question, a Poisson generator represents an adequate model for it. Try out different options how to connect it to the recurrent network, and play with the intensity of the input. Networks in the neocortex are sparse (10% connectivity). How does the input influence the response of such a sparse network?

3. If the Poisson input is neither too weak nor too strong, the activity in the recurrent network looks random itself, both on the level of spike trains and on the level of membrane potentials. What are the reasons for this? You can test the influence of the random Poisson input by replacing it by a deterministic DC source. Make sure that the amplitude of the injected DC is roughly equivalent to the Poisson input applied previously. As a criterion, you can either compare mean membrane potentials in an intracellular recording of one of the neurons in the network, or you can base your calibration on the response rate of one or several neurons in the network.

NEST Desktop offers several different methods to wire up a network randomly. Make sure you understand how they work, and which of them allows you to generate sparse networks. Your networks should be not too small and not too large. Networks with 100 neurons may be a good start. Working with a larger number of neurons is possible, but the initial wiring and the simulation of activity can take quite long then. Also make sure that each neuron receives enough external input, as the input from other neurons in the network will not suffice to push the membrane potential above threshold and elicit any activity.

16.3.2 Recurrent networks of inhibitory neurons

There are networks in the brain that consist of only inhibitory neurons. In particular, several nuclei in the basal ganglia (e.g. the striatum, and both parts of the globus pallidus) are comprised of GABAergic neurons. It is clear that such networks depend on excitatory drive from outside to become active.

1. Use a similar setup as for the all-excitatory networks studied before, except that all neurons should now be inhibitory. Again explore how the properties of the input determine the properties of the network response. Using the same random connectivity and external inputs that lead to the same mean firing rates, what is the main difference between all-inhibitory and all-excitatory networks?

2. Additional parameters that are relevant in a network, like the synaptic transmission delay, have a rather strong influence on the activity dynamics in recurrent networks. Systematically vary this particular parameter and describe the consequences you observe. Make sure you vary it only in small steps, as the network might react quite sensitively.

NEST Desktop generally allows to set up big networks with weak recurrent synapses, or smaller networks with strong recurrent synapses. To see the impact of the transmission delay of recurrent synapses on the network activity, the recurrent contribution to the network activity needs to be high enough. Neurons communicate with other neurons in the central nervous system using both electrical and chemical signals depending on the transmitter system.

16.3.3 Recurrent networks of excitatory and inhibitory neurons

We will now get back to the neocortex and add the missing 20% of inhibitory neurons to the recurrent network. Each of the two subpopulations is now conceived as a random recurrent network in its own respect, and the two subnetworks are mutually coupled with each other in a random fashion. This means that we now have to track four connectivity parameters, for the following types of synapses: $E \to E$, $E \to I$, $I \to I$, $I \to E$. Although it is interesting and relevant to vary them independently, it is recommended to use the same connection probability of 10% throughout all types of synapses, and to use exactly the same strength $J > 0$ for all excitatory synapses ($E \to E$, $E \to I$) and same strength ˘$gJ < 0$ for all inhibitory synapses ($I \to I$, $I \to E$). The number $g > 0$ is a unit-less factor describing how dominant inhibition is in the network. The value $g = 4$ is special, because in this setting the relatively small number of inhibitory neurons is exactly compensated by an increased strength of inhibitory synapses.

1. Set up a random recurrent network according to the prescription given above. Fix a value of $g = 5$ while you search for good values of the other parameters. First of all, the strength J of excitatory synapses must be

matched to the typical input a neuron gets. What is your criterion? As for the other networks considered before, external excitatory drive will be necessary to induce meaningful activity in this network. Fix a good value for the rate of the external drive, just above threshold. The goal should be to establish stable activity in the network, which is characterized by low firing rates, irregular (Poisson-like) spike trains, and a low degree of synchrony across neurons. Describe your experiences during the parameter search, and formulate your recommendations how to make this a reproducible and joyful procedure.

2. Whatever configuration you are now working with, the activity should be stable against external perturbations. In fact, such dynamic stability would be a highly desirable property of any biological system. For example, you can use an additional DC input and apply a strong depolarizing perturbation to all neurons, mimicking the effect of a flash of transcranial magnetic stimulation, TMS. After the perturbation is turned off, the network should return to its previous activity. Is this "return to the fixed point" a fast or a slow process? Can you estimate a time constant for it?

3. Stable "fixed point activity" is characterized by a tight temporal balance between excitation and inhibition. This balance can be demonstrated by comparing the time-resolved PSTH fluctuations of the excitatory population to the inhibitory population. A "scatter plot" may come handy to display the observations made "by eye" in a more objective way: Simultaneous bin counts of excitatory activity x and inhibitory activity y make the coordinates (x, y) of data points in a two-dimensional display. What is the relation of individual spike trains with the population activity measured by the PSTH?

4. You should now vary the parameter g and document all important changes. Changing this parameter has the potential to alter the balance between excitation and inhibition. Describe how the balance is affected, and what the consequences of this for the recurrent network dynamics are.

Video tutorial

CHAPTER 17

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.

Topics

17.1 Network models of decision making

Deciding between two alternatives is a very basic, but also a very essential task the brain has to perform. However, not much is known about how this process is actually implemented in biological brain networks. The stable fixed point dynamics considered in the previous section does not seem to suggest anything that looks like a decision making system. The conceptual model that has been suggested for decision making, however, is surprisingly simple: Dividing the excitatory neuron population into two halves, and arranging the strength of synapses within each population to be a little stronger than the synapses across populations, the two excitatory populations start to compete with each other. Depending on the external input each of them gets, one of the two population "wins" by suppressing the other one with the help of the inhibitory population. If this happens, the symmetry is broken and a "decision" has been made. This kind of "winner takes all" dynamics in EEI networks is considered as a candidate mechanism for decision making, e.g. for classification tasks in sensory systems of the neocortex ("cat/dog", "left/right", "good/bad"). A very similar mechanism, based however on splitting the inhibitory population, has been postulated for subcortical brain areas. In the basal ganglia, for example, decisions are made whether a planned action, for example a movement, is executed or terminated ("go/nogo").

To explore decision making with computer simulations, we recommend that you start with the network models that were prepared for the course. In the navigation sidebar on the left, click with the right mouse button on the projects icon (contains a brain symbol). Then choose the option "Upload projects" (icon: arrow pointing upwards) from the dropdown menu (three vertical dots) and upload the file provided in the repository (tba). This is, by the way, a general method with which you can share models with others, including all settings and parameters.

17.1.1 Decision making in EEI Networks

The provided network model comprises three (instead of just two) subpopulations: two of them are excitatory (a, b), and one is inhibitory (c). This was achieved by dividing the excitatory population of the model considered in the previous chapter into two equal halves. In a fully symmetric setting, the two excitatory populations (a, b) of course do not behave any different. However, if the symmetry is broken, one of the populations may take over and dominate the other one. There are different possibilities how this can be achieved.

1. Figure out which manipulations of the network configuration lead to an unequal activation of the two excitatory populations, for example a stronger activation of population (b). You may test the number of neurons, the connectivity (synaptic weights or transmission delays) within or across populations, or external input. It is suggested that you change only one parameter at a time, and reset all the other parameters to their default values. To account for statistical fluctuations, you should perform repeated simulations for each parameter setting, using different seeds of the random number generator.
2. What is the exact role of the inhibitory population in the competition process? How does the activation of inhibitory neurons reflect a decision?

17.1.2 Decision making in EII networks

A decision making network can also be established by dividing the inhibitory population into two halves. In this scenario, the two inhibitory subpopulations (b, c) are acting as competitors. Very much like in the EEI scenario considered in the previous section, the goal is to understand which factors contribute a bias in the activation of one population (b, light blue).

1. Consider now an alternative decision making system, which is based on a EII scenario. You should devise a strategy to study its behavior similar to the one you developed for the EEI model.

   Note: Inhibitory synapses have opposite effects as compared to excitatory ones.

2. What is the exact role of the excitatory population in this process? How does the activation of excitatory neurons reflect a decision?

17.1.3 Decision making

Perceptual decisions have to be made when there are two conflicting interpretations of the input (e.g. leftward vs. rightward movement of a subtle stimulus). The brain will then not maintain both interpretations, but rather decides for one of them and adjusts its behavior accordingly. Typically, the “stronger” input wins, but additional factors (e.g. memory previous encounters, or random perturbations) might contribute as well. The same holds for action selection, which is a necessary component to resolve conflicts in the behavioral goals of an animal.

1. Interpret the phenomena observed in simulations of EEI and EII networks in relation to a hypothetical “decision making” process. What are the requirements to enable “decision making”? Which additional components would you need for a full-blown decision making system?

CHAPTER 18

How to deploy NEST Desktop

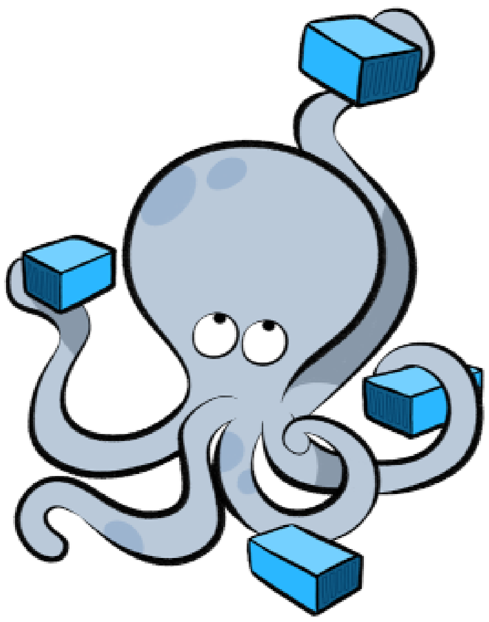
This guide provides detailed documentation on how to deploy NEST Desktop. You can read the deployment instructions by clicking one of these images below:

Guides to deploy NEST Desktop

- *Deploy with Docker Compose*
- *Deploy on OpenShift*
- *Deploy on OpenStack*

CHAPTER 19

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

```
apt install docker-compose
```

Get the configuration file 'docker-compose.yml' from GitHub

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

It will pull images of NEST Desktop from https://hub.docker.com/r/nestdesktop/app) and NEST Simulator can be started from within the official NEST image (https://hub.docker.com/r/nestsim/nest).

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:8000` and `http://localhost:5000`, 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 services of nest-desktop and nest-server.

```
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 for docker-compose

List containers.

```
docker-compose ps
```

If there are no services (`nest-desktop` and `nest-server`) 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 the services `nest-desktop` and `nest-server` as daemon.

```
docker-compose start
```

Stop the services `nest-desktop` and `nest-server`.

```
docker-compose stop
```

Shutdown the services `nest-desktop` and `nest-server`.

```
docker-compose down
```

Acknowledgments

Thanks for the help:

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

CHAPTER 20

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.
- 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-ebrains. 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)

CHAPTER 21

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

CHAPTER 22

How to develop NEST Desktop

This is the developer guide, providing more detail on how to develop NEST Desktop.

Get the source code

The source code of NEST Desktop is hosted on GitHub. Clone NEST Desktop from the GitHub repository:

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

Guides for the development

- *Prepare the development environment*
- *Working on the source code*
- *Build and publish packages*
- *User documentation workflow*

Guides for the developers

- *Command API*
- *The semantic versioning*
- *The concept of the interface*
- *References for images*

CHAPTER 23

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

- Python 3.6 or higher
- Node.js, Yarn
- NEST Simulator 3.0 or higher

You can install these requirements in the host system.

However, I prefer to use a Singularity container and leave the host system unchanged. For this, I prepared a Singularity recipe that build a container with the required packages for the development. You can find the definition file in `singularity/nest-desktop-dev.def` for building this Singularity container.

Build an environment with Singularity

The definition file `singularity/nest-desktop-dev.def` contains an adequate environment to develop and build NEST Desktop.

Build a singularity image:

```
singularity build nest-desktop-dev.sif singularity/nest-desktop-dev.def
```

Go to the shell of singularity container:

```
singularity shell nest-desktop-dev.sif
```

Node modules

Install node modules for NEST Desktop:

```
yarn install
```

Check if any node modules are outdated:

```
yarn outdated
```

Upgrade outdated node modules:

```
yarn upgrade
```

CHAPTER 24

Work on the source code

First, prepare the development environment with the required packages.

```
yarn serve
```

The Live Development Server is serving at `http://localhost:8080`.

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
```

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 commited, stage and commit them:

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

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

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

CHAPTER 25

Build and publish package

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
```

CHAPTER 26

Command API

This documentation guide provides detailed information about the command `nest-desktop`.

Show the usage of the `nest-desktop` command:

```
nest-desktop
```

Options for `nest-desktop`:

```
nest-desktop status|start|stop|restart [-h <HOST>] [-p <PORT>]
```

Commands

status display the status of NEST Desktop

start start a new server instance for NEST Desktop

stop stop a server instance running on <HOST>:<PORT>

restart restart (i.e. stop and start) a server on <HOST>:<PORT>

Arguments

-h <HOST> use hostname/IP address <HOST> for the server [default: 127.0.0.1]

-p <PORT> use port <PORT> for opening the socket [default: 8000]

CHAPTER 27

User documentation

We use Sphinx to generate the documentation and Read the Docs to publish it. Sphinx uses reStructuredText. To learn more about the syntax, check out this quick reference.

Requirements

- Sphinx
- Read the Docs Sphinx Theme

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

```
python3 -m pip install sphinx sphinx_rtd_theme
```

Development: Render HTML offline

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

```
rm -r _build/; make html
```

Publication: Push to ReadTheDocs

The documentation files for the main branch are automatically rebuilt (and updated) each time a push is made to the repository. The docs for other versions depend on the GitHub tags.

CHAPTER 28

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. 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.

CHAPTER 29

Concept of the interface

General layout concept of the interface.

NEST Desktop contains three columns with clear function. The left column shows the navigation to route pages. The center column renders the main content of the page, whereas the right column displays the controller content for the modification.

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.

CHAPTER 30

References

Images

nest.svg https://nest-simulator.org/

spike-transmission-carlos.svg Carlos Toledo Suárez, a former PhD of Abigail Morrison's lab at the Jülich Center, who passed away in 2013.

Bibliography

[BCF] Bernstein Center Freiburg, Faculty of Biology, University of Freiburg, Freiburg, Germany

[HCI] Department IV - Computer Science, Human-Computer Interaction, University of Trier, Trier, Germany

[INM-6] Institute of Neuroscience and Medicine (INM-6), Jülich Research Center, Jülich, Germany

[NMBU] Norwegian University of Life Sciences, As, Norway