Funded by the European Union
NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA
Consiglio Nazionale delle Ricerche

Realistic modelling of brain microcircuits: The Brain Scaffold Builder

Filippo Marchetti, PhD - UniPV

Napoli, 03 December 2024

UNIVERSITÀ DI PAVIA
EBRAINS - Italy
Training and Innovation Centre

Brain Scaffold Builder

A black-box component framework for multiparadigm neural modeling, designed to offer the essential tools needed for network reconstruction, while allowing the user to configure the parameters required for the specific use case.

BSB is available through **PyP**I and requires Python **3.9** or higher.

It is recommended to create a Python environment and to install the BSB within:

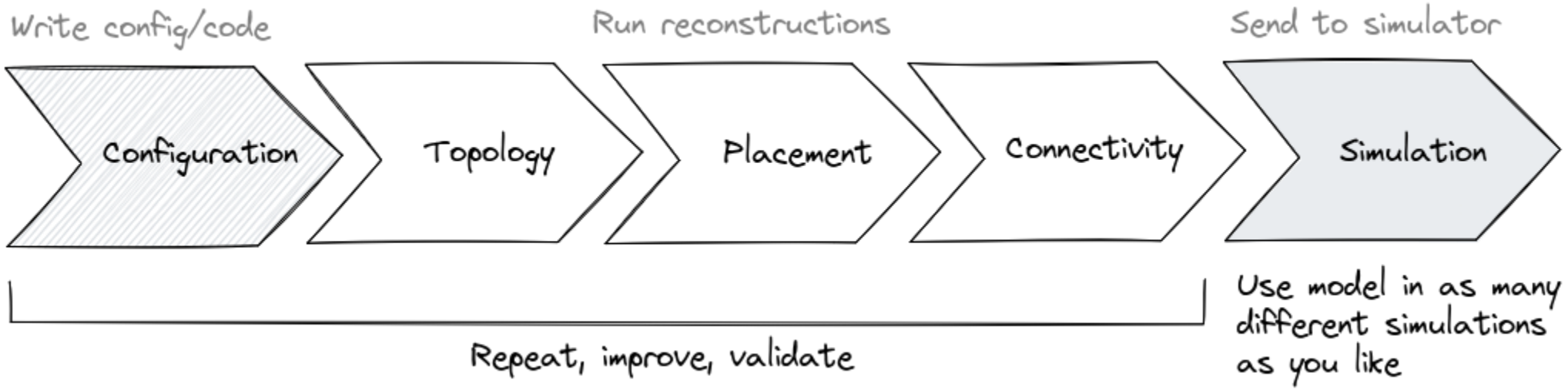
```
python3  -m venv  bsb_env
source bsb_env/bin/activate
```

Installation

```
pip install bsb
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS - Italy Training and Innovation Centre

Tipical BSB workflow



The configuration allows you to specify the shape and size of the network, as well as the arrangement of cells and their connections. BSB also offers an interface to simulate the activity of your model using various simulators

For most of your reconstructions you may not have to code!

Configuration Files

The main input file contains a description of the model, rather than its implementation or data. However, it can be easily interpreted by BSB. Supported file formats include JSON and YAML.

This file is divided into 8 primary configuration blocks, each of which defines a specific aspect of the network.

Configuration blocks:

```
{
  "storage": {},
  "network": {},
  "regions": {},
  "partitions": {},
  "cell_types": {},
  "placement": {},
  "connectivity": {},
  "simulations": { }
}
```

Funded by the European Union NextGenerationEU

Initial configuration

The first block set the name of the configuration object,
in our case: **Starting example**

Then in the ‘storage’ block we define the file where the network will be stored: **network.hdf5** and its estension (‘hdf5’)

Now we have to set our network dimensions: it will be contained in a parallelepiped of size 200 * 200 * 200 μm

```
{
    "name": "Starting example",
    "storage": {
      "engine": "hdf5",
      "root": "network.hdf5"
    },
    "network": {
      "x": 200.0,
      "y": 200.0,
      "z": 200.0
    },
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Give a shape to your network

Your network model needs a description of its shape, which is called the topology of the network. The topology consists of 2 components: **Regions** and **Partitions**.
Regions combine multiple partitions and/or regions together, in a hierarchy, all the way up to a single topmost region, while partitions are exact pieces of volume that can be filled with cells.

In this example we build a stack of two layers called **base_layer** and **top_layer**. Each layer has a thickness of 100 µm, therefore filling the whole network volume.

```
"regions": {
  "brain_region": {
    "type": "stack",
    "children": ["base_layer", "top_layer"]
  }
},
"partitions": {
  "base_layer": {
    "type": "layer",
    "thickness": 100
  },
  "top_layer": {
    "type": "layer",
    "thickness": 100
  }
},
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Define the cells to use

Cell Types define cell populations.

In its simplest form, a cell type can be defined by its soma radius and the number of cells to be placed, either using a density value or a fixed count.

```
"cell_types": {
  "base_type": {
    "spatial": {
      "radius": 2.5,
      "density": 3.9e-4
    }
  },
  "top_type": {
    "spatial": {
      "radius": 7,
      "count": 40
    }
```

Funded by the European Union
NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Fill the volume

The placement blocks are in charge of placing cells in the partitons using cell type indications. For each placement component, you should specify the strategy to use, the list of cell types names to place and a list of partitions in which you want the placement to happen.

In this case we use the RandomPlacement strategy, which assign a random position to each cell's soma, for both base_layer (example_placement) and top_layer (top_placement).
But other stragegy are available in BSB.

```
"placement": {
  "example_placement": {
    "strategy": "bsb.placement.RandomPlacement",
    "cell_types": ["base_type"],
    "partitions": ["base_layer"]
  },
  "top_placement": {
    "strategy": "bsb.placement.RandomPlacement",
    "cell_types": ["top_type"],
    "partitions": ["top_layer"]
  }
},
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Connect the neurons

This component contains the blocks that specify the connections between population of cells. For each block you have to define the **connection strategy** and the list of cell types for pre and post synaptic groups.

In this example we connect **base_type** cells to **top_type** cells with a simple **AllToAll** strategy, where every pre-synaptic cell is connected to all the members of the post-synaptic cell population.

```
"connectivity": {
  "A_to_B": {
    "strategy": "bsb.connectivity.AllToAll",
    "presynaptic": {
      "cell_types": ["base_type"]
    },
    "postsynaptic": {
       "cell_types": ["top_type"]
    }
  }
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Compile the configuration file

Once the configuration file is ready is possible to reconstruct our network:

```
bsb compile --verbosity 3  network_configuration.json
```

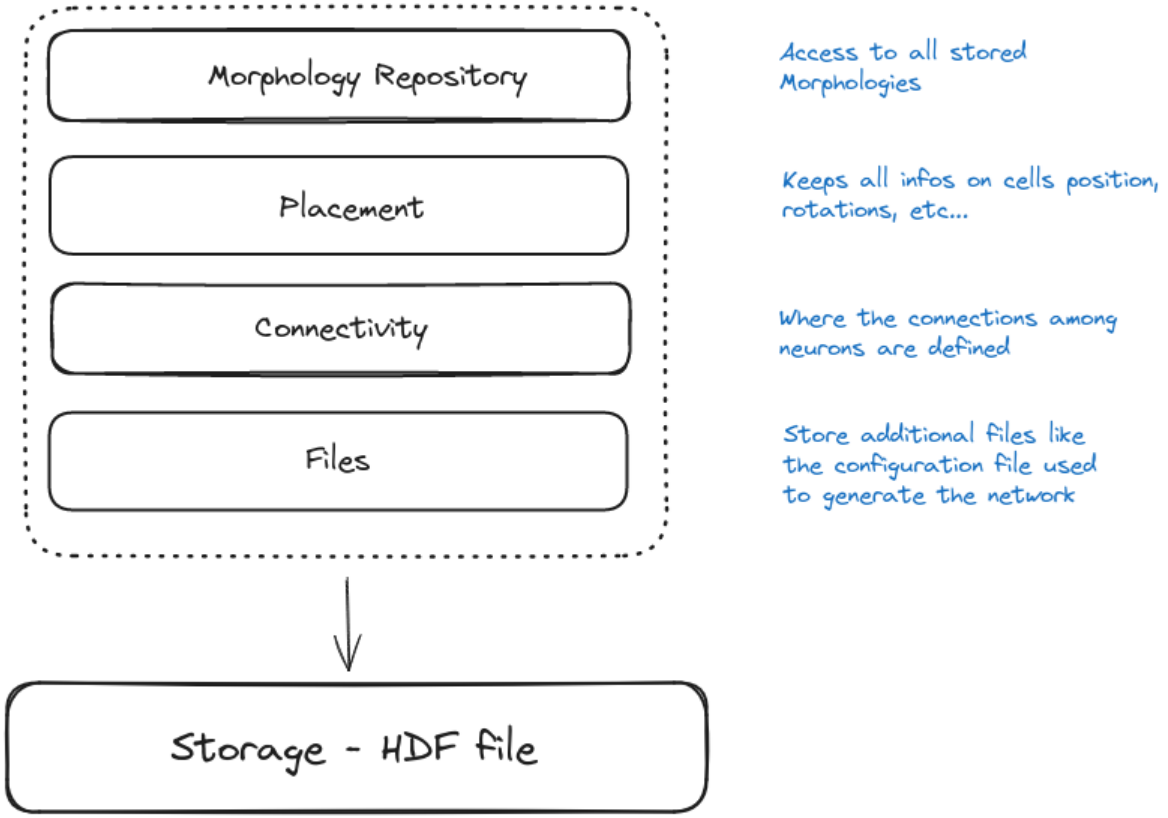
When BSB compiles a scaffold, it extracts the blocks defined in the configuration and runs the reconstruction pipeline and for every step it stores the results in the storage. All the results are accessible in the *network.hdf5* file.

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

In network.hdf5 file are stored relevant data

Use the Hierarchical Data Format

In the storage data are organized in four main blocks.



Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

How to access scaffold data

After you reconstruct successfully the network with BSB all the information can be retrieved through Python code.

So create a new script and import the **from_storage** method:

```
from bsb import from_storage

scaffold = from_storage('network.hdf5')
```

Once you have loaded the scaffold object, you have the access to its configuration and data:

- Placement Set: contain information about location of the cells
- Connectivity Set: list all the connection established for the connection strategy

With BSB you can use different simulators

Once the network is built you can use BSB to interface with different simulators, you only have to provide configuration parameter for the simulation setup.

The neuron simulator that are supported:

- The NEURON simulator
- The Arbor simulator
- The NEST simulator

Detailed neuron simulations

Point-neuron simulations

Funded by the European Union NextGenerationEU

Running point-neuron simulations

Requires to install NEST, but unfortunately it is not available in the PyPI library.

Please follow instruction on the website to install NEST:

https://nest-simulator.readthedocs.io/en/stable/installation/index.html

If the installation is successful, you should be able to import both bsb and nest within your environment:

```
import bsb
import nest
```

Then, you need to install the bsb-nest package which contains the BSB interfaces for NEST:

```
pip install bsb-nest
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Configurate simulation parameters

Now we want to introduce a simulation block in the previous configuration file,

Introducing in our network a basal stimulation.

We will call this simulation «basal_activity».

Here we define:

- The *simulator* to use
- The *resolution* (time step of the simulation, in ms)
- The *duration* (total lenght of the simulation, in ms)

```
"simulations": {
    "basal_activity": {
      "simulator": "nest",
      "resolution": 0.1,
      "duration": 5000,
      "cell_models": {
      },
      "connection_models": {
      },
      "devices":{
      }
}
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

NEST point-neuron models

The simulator needs to know which neuron model to use for describing the behavior of our cell types. NEST offers a wide range of neuron and synapse models in its library.

https://nest-simulator.readthedocs.io/en/stable/neurons/index.html

For this simulation, we will use the simple conductance-based leaky Integrate-and-Fire model.

Integrate-and-Fire equations:

$$\frac{cdV}{dt} = -I_{leak} - I_{spike} - I_{syn}$$

$$I_{leak} = \frac{c(V - V_p)}{\tau_p}$$

Funded by the European Union NextGenerationEU

Insert cell models in the configuration

We add the integrate-and-fire neuron model (iaf_cond_alpha) and set its parameters in the ‘cell_models’ field.

In our case both ‘top_type’ and ‘base_type’ cells will have the same model

With the ‘constants’ attribute is possible to tune the parameters, for example we change the refractory period to 1.5 ms and the initial membrane potential value to -62 mV

All neuron parameter and units are described in the NEST documentation

```
"cell_models": {
    "base_type": {
     "model": "iaf_cond_alpha"
    },
    "top_type": {
     "model": "iaf_cond_alpha",
     "constants": {
      "t_ref": 1.5,
      "V_m": -62.0
     }
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

NEST synapse models

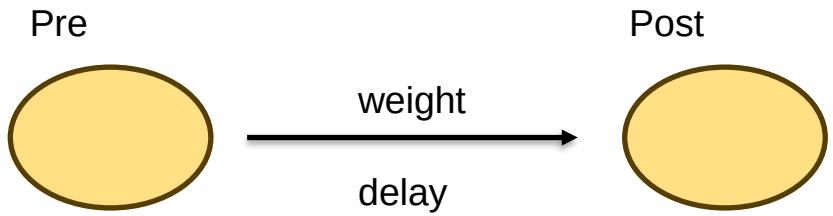
At this point is necesary to establish the connection between cells by defining a synapse model that transform a presynaptic spike event into a postsynaptic input current.

NEST provides a set of synapse models:

https://nest-simulator.readthedocs.io/en/stable/synapses/index.html

We are going to use the most basic model: the static_synapse

This model does not have any plasticity dynamics, it will just forwards the events with a certain weight and delay



Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Define synapse parameter

A synapse model has to be defined for every connection, similar to the cell_model block the key used must match a connectivity set previously created, in our case is A_to_B.

Assign the static_synapse with a weight of 100 and a delay of 1 ms.

```
"connection_models": {
 "A_to_B": {
   "synapse": {
    "model": "static_synapse",
    "weight": 100,
    "delay": 1
    }
  }
},
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Add devices to stimulate and record

Finally we need to add an interface to stimulate our cells, calling the new device background_noise.
We will target the ‘base_type’ cells as they will stimulate the other cells in cascade.

We are going to use a ‘poisson_generator’ that will produce random spike trains at 10 Hz frequency.

Since the poisson generator is connected through a static synapse to the neurons you can also change its ‘weight’ and ‘delay’.

```
"devices": {
    "background_noise": {
     "device": "poisson_generator",
     "rate": 10,
     "targetting": {
      "strategy": "cell_model",
      "cell_models": ["base_type"]
     },
     "weight": 40,
     "delay": 1
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Add devices to stimulate and record

We also need to keep track of the behaviour of our network in rensponse of the stimulus, so we collect the spike events using a NEST ‘spike_recorder’.
Spike recorders also also connected to the cell so they will receive the spike after a short delay.

We add a recorder for the cells of the base layer and call it base_layer_record.
Setting the delay at 0.1 ms.

```
"base_layer_record": {
        "device": "spike_recorder",
        "delay": 0.1,
        "targetting": {
          "strategy": "cell_model",
          "cell_models": ["base_type"]
        }
      },
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Add devices to stimulate and record

To capture the activity in the top layer, we add a third device block to target the top_type cells. The delay remains set to 0.1 ms, and we name this block `top_layer_record`.

Now, all the simulation parameters have been defined.

```
"top_layer_record": {
        "device": "spike_recorder",
        "delay": 0.1,
        "targetting": {
          "strategy": "cell_model",
          "cell_models": ["top_type"]
        }
      },
```

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Run a simulation with BSB

Simulations are separated from the reconstruction pipeline so you do not need to recompile,

but you need to reconfigure the stored scaffold to update all the simulation block that we added.

```
bsb reconfigure network.hdf5 network_configuration.json
```

Now you can run your simulation using the simulate command:

```
bsb simulate network.hdf5 basal_activity
```

After few minutes it will generate a result file with a random assigned name.

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

How to retrieve data from nio files

To retrieve the information from your **.nio** file, you need to open it with python.

The content is read/written using the Neo Python package, a library designed for handling electrophysiology data. Therefore, add the following lines:

```
from neo import io

my_file_name = "simulation-results/NAME_OF_YOUR_NEO_FILE.nio"
block = io.NixIO(my_file_name, mode="ro").read_all_blocks()[0]
segment = block.segments[0]
my_spiketrains = segment.spiketrains
```

These files are organized in blocks and every block is divided in segments.
In our case, we have one **block** since we ran one simulation.
For this block, we have one **segment** since our data is recorded with the same time frame
Read the neo documentations for more info:
https://neo.readthedocs.io/en/latest/read_and_analyze.html

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

More on spike trains analysis

Various analyses can be perform on spiking data, and several tools facilitate these.

If you want to learn more about spike analysis, we recommend:

- the Analysis of Parallel Spike Trains, https://link.springer.com/book/10.1007/978-1-4419-5675-0
- the Neuronal Dynamics, https://neuronaldynamics.epfl.ch/index.html
- the Elephant python package which integrates Neo, https://elephant.readthedocs.io/en/latest/index.html

Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

To sum up

- **bsb_nest** interfaces NEST, allowing you to extend your configuration for simulations:
  - You can specify any NEST point neuron model (and its parameters) to your cell population
  - You can choose any NEST synapse model (and its parameters) to your connections
  - You can stimulate circuit and record its activity with the **devices**

- Use the "**bsb reconfigure**" command if you need to update your stored configuration without compiling your circuit

- Use the "**bsb simulate**" command to run your simulations

- BSB simulations produce **.nio** files in the **Neo** format.

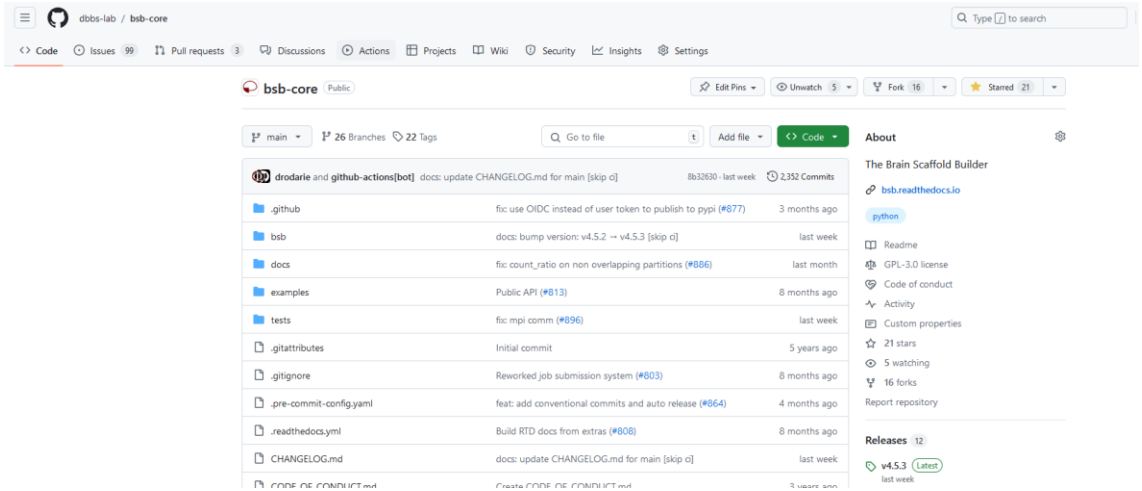
Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Find help on the BSB

- The BSB is a suite of python libraries which code is available on github: https://github.com/dbbs-lab/bsb

- Find its documentation and tutorials online: https://bsb.readthedocs.io/en/latest/

- If you encounter any issue with your code or documentation, please contact us.



Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
EBRAINS Italy

Acknowledgements

D'Angelo LAB:

- Egidio D'Angelo (Director)
- Claudia Gandini Wheeler Kingshoot
- Francesca Prestori
- Lisa Mapelli
- Claudia Casellato
- Fulvia Palesi
- Stefano Masoli
- Pawan Faris
- Danilo Benozzo
- Doris Pischedda
- Simona Tritto
- Dimitri Rodarie
- Alberto Vergani
- Abdul Haleem But
- Martina Rizza
- Dianela Osorio
- MariaLaura De Grazia
- Emiliano Buttarazzi
- Francesco Pio Ercolino

Contacts:

me: filippo.marchetti@unipv.it

Funded by the European Union NextGenerationEU