

TVB-multiscale: interfacing TVB with NEST spiking networks for multiscale co-simulation

Dionysios Perdikis, L. Domide, J. Mersmann, M. Schirner, P. Ritter
Brain Simulation Section, Department of Neurology,
Charité—Universitätsmedizin Berlin



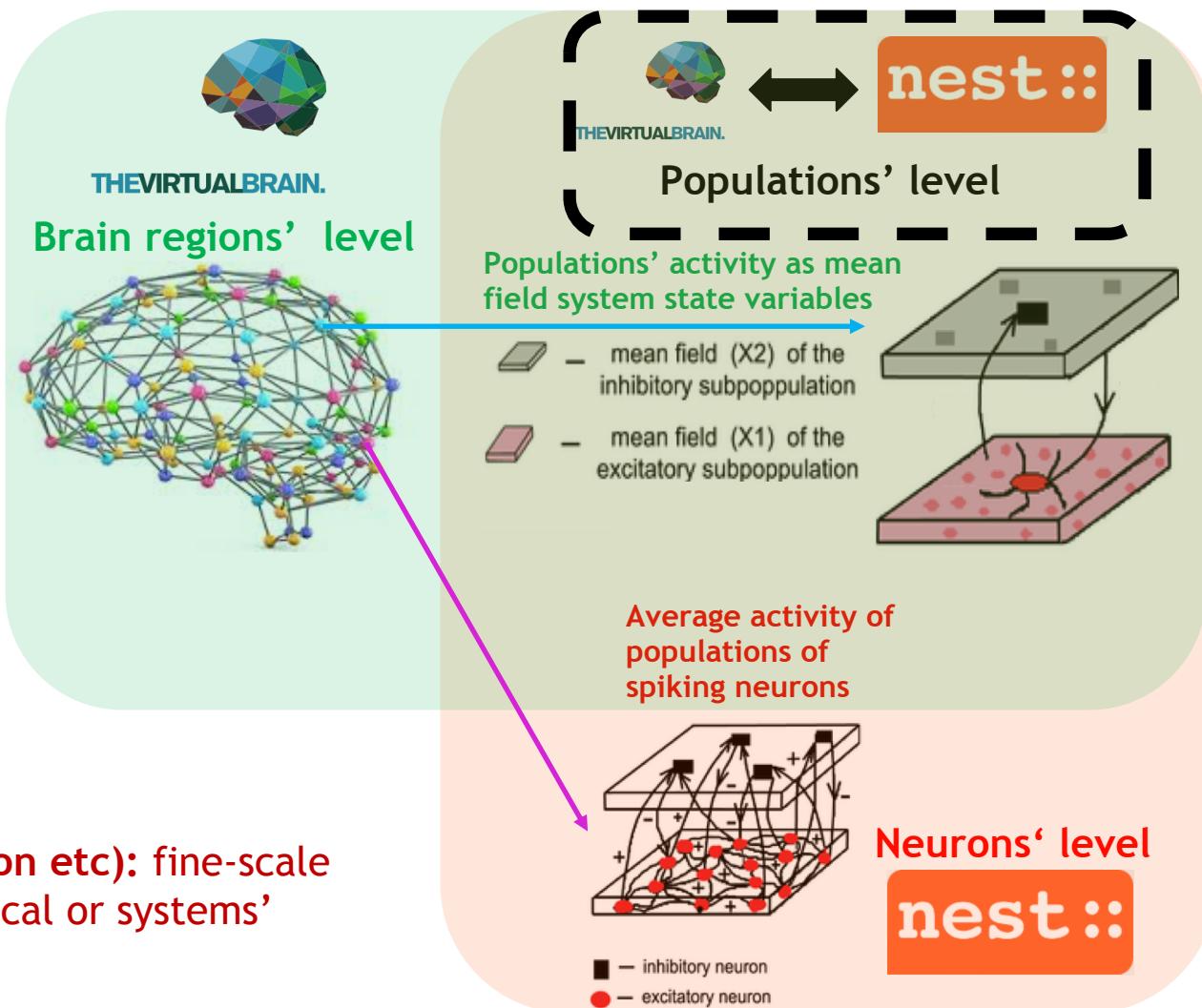
Human Brain Project

Motivation

TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.

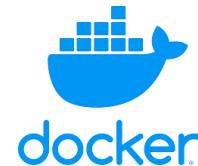


Spiking simulators (NEST, Neuron etc): fine-scale simulation for investigation of local or systems' neural mechanisms



Software structure

- ❖ ***TVB-multiscale/tvb_multiscale***: extends TVB to perform multiscale co-simulation.
 - spiking_models: region_node, devices, network, and their builders
 - interfaces: tvb_to_spikeNet, spikeNet_to_tvb, and their builders
 - tvb/simulator_builder
 - plot, io (read/write to/from .h5 files),
- ❖ ***TVB-multiscale/tvb_nest***: Python interface of TVB with (Py)NEST, for co-simulation of TVB and NEST models.
 - nest_models: region_node, devices, network, their builders, builders of specific model
 - interfaces: tvb_to_nest, nest_to_tvb, their builders, builders of specific models
- ❖ ***TVB-multiscale/tvb_elephant***:
 - spike_stimulus_builder
- ❖ docker, examples, tests, docs, tvb_utils (utility scripts)





Default workflow

1. Setup configuration
(min_delay, integration time resolution etc)
2. Build TVB model and simulator
(connectivity, model, coupling, integrator, monitors etc)
3. Build Spiking Network model
(neural populations, stimulation and measuring devices)
4. Build interfaces
5. (Co-)Simulate
6. Gather results, analyze, plot, write to .h5 files



EBRAINS collaboratory



• Co-Simulation The Virtual Brain Multiscale

Co-Simulation The Virtual Brain Multiscale

Last modified by Lia Domide on 2020/06/26 22:21

jupyter documentation_example Last Checkpoint: 6 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

Logout Control Panel
Not Trusted Python 3 O



TVB-NEST: Bridging multiscale activity by co-simulation

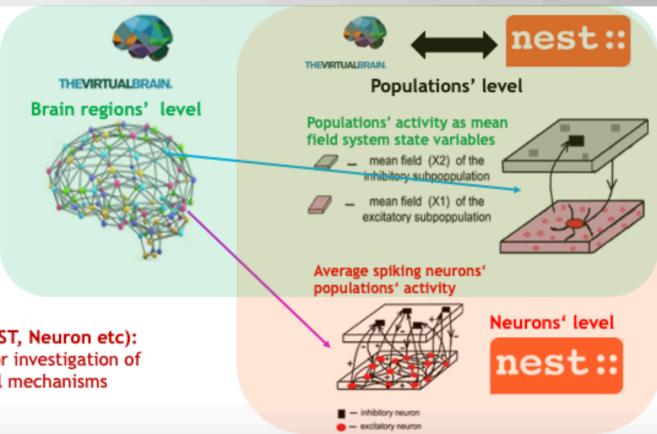
Step-by-step learn how to perform a co-simulation embedding spiking neural networks into large-scale brain networks using TVB.

```
In [1]: from IPython.core.display import Image, display
display(Image(filename='./ConceptGraph1.png', width=1000, unconfined=False))
```

Motivation

Interfacing at populations' level for multiscale co-simulation

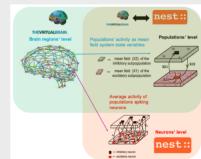
TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.



Spiking simulators (NEST, Neuron etc): fine-scale simulation for investigation of local or systems' neural mechanisms

<https://wiki.ebrains.eu/bin/view/Collabs/the-virtual-brain-multiscale/>

TVB Co-Simulation



Multiscale: TVB - NEST

Authors: D. Perdikis, L. Domide, J. Mersmann, M. Schirner, P. Ritter

For more details on TVB itself, check this: <https://wiki.ebrains.eu/bin/view/Collabs/the-virtual-brain/>

Use our Jupyter Hub setup online

https://tvb-nest.apps.hbp.eu/user/dionperd/notebooks/TVB-NEST-Examples/documentation_example.ipynb

Contents

- TVB Co-Simulation
 - Use our Jupyter Hub setup online
 - Running TVB-NEST in ...



TVB-multiscale on EBRAINS collaboratory

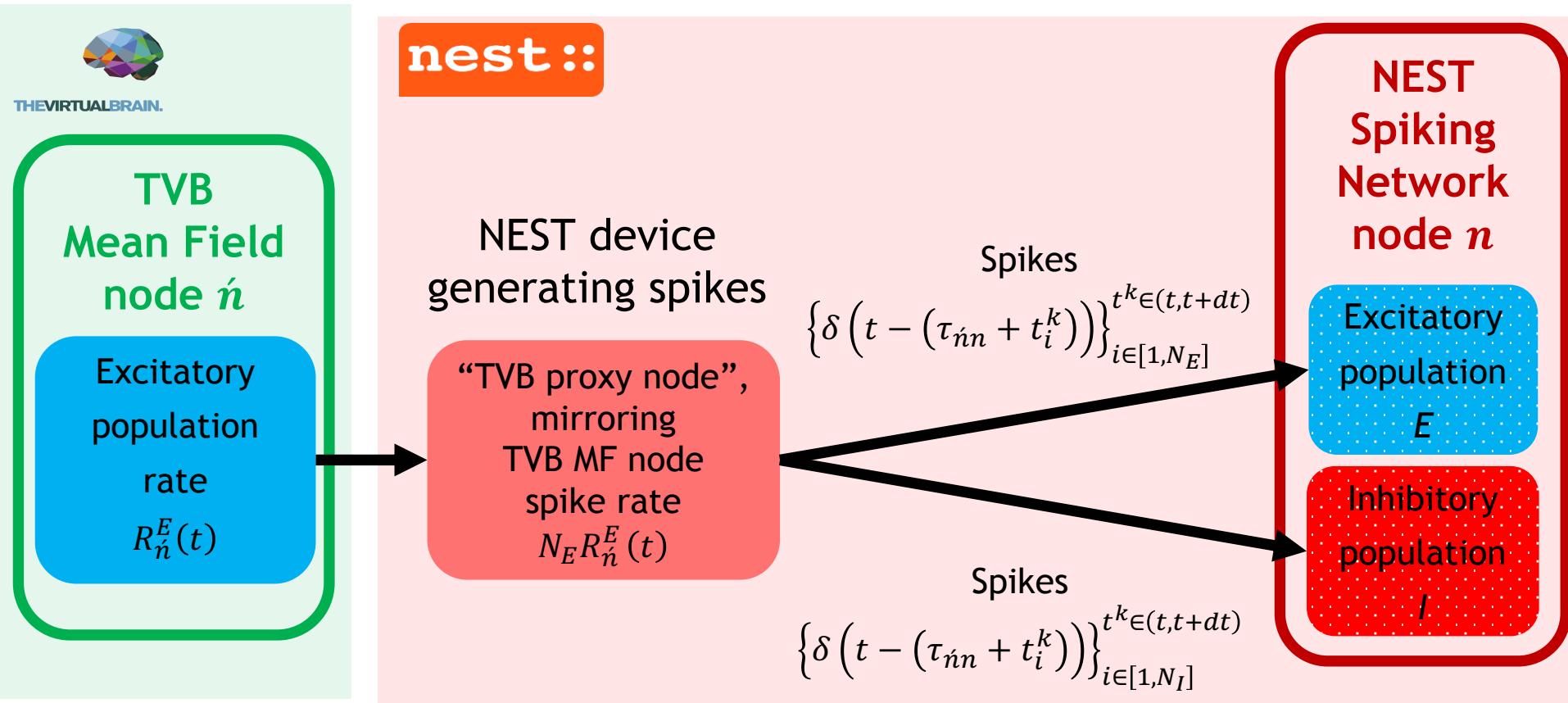
❖ Use case in EBRAINS Collaboratory:

- 1 excitatory (E) and 1 inhibitory (I) population per region node
- Coarse-scale **MF nodes** in TVB with a reduced Wong-Wang model
- fine-scale spiking networks **SP nodes** with a Wong-Wang like spiking neuron model programmed in NEST



Use case

TVB to NEST coupling via TVB “proxy” nodes: spike rate





Use case

TVB to NEST coupling via TVB “proxy” nodes: spike rate

```
# For spike transmission from TVB to NEST devices acting as TVB proxy nodes with TVB delays:

tvb_nest_builder.tvb_to_nest_interfaces = [
    {"model": "inhomogeneous_poisson_generator",
     "params": {"allow_offgrid_times": False},
    # # -----Properties potentially set as function handles with args (nest_node_id=None)-----
     "interface_weights": 1.0 * N_E/10,
    # Applied outside NEST for each interface device
    # -----Properties potentially set as function handles with args (tvb_node_id=None, nest_node_id=None)-
    # To multiply TVB connectivity weight:
     "weights": lambda tvb_node_id, nest_node_id:
                 random_normal_tvb_weight(tvb_node_id, nest_node_id,
                                           tvb_nest_builder.tvb_model.G[0]*
                                           tvb_nest_builder.tvb_weights,
                                           sigma=0.1),
    #
     # To add to TVB connectivity delay:
     "delays": lambda tvb_node_id, nest_node_id:
                 tvb_delay(tvb_node_id, nest_node_id, tvb_nest_builder.tvb_delays),
    "receptor_types": lambda tvb_node_id, nest_node_id:
                      receptor_by_source_region(tvb_node_id, nest_node_id, start=1),
    #
    #           TVB sv -> NEST population
    "connections": {"R_e": ["E"]},
    "source_nodes": None, "target_nodes": None}] # None means all here
```



Use case

TVB to NEST coupling via TVB “proxy” nodes: current



**TVB
MF node**

Excitatory population synaptic gating
 $S_n^E(t)$

nest ::

NEST device generating dc current

“TVB proxy node”, mirroring TVB MF Node current

$$I_n^{dc}(t) = N_E J_{NMDA} S_n^E(t)$$

Current:
 $GC_{nn} I_n^{dc}(t - \tau_{nn})$

Current:
 $\lambda GC_{nn} I_n^{dc}(t - \tau_{nn})$

**NEST
SP node n**

Excitatory population
 V_n^m

Inhibitory population
 \dot{V}_n^m



Use case

TVB to NEST coupling via TVB “proxy” nodes: current

```
# For injecting current to NEST neurons via dc generators acting as TVB proxy nodes with TVB delays:
tvb_nest_builder.tvb_to_nest_interfaces = [
    {"model": "dc_generator", "params": {}},
# -----Properties potentially set as function handles with args (nest_node_id=None)-----
#   Applied outside NEST for each interface device
    "interface_weights": 1.0 * N_E,
# -----Properties potentially set as function handles with args (tvb_node_id=None, nest_node_id=None)-
#   To multiply TVB connectivity weight:
    "weights": lambda tvb_node_id, nest_node_id:
        random_normal_tvb_weight(tvb_node_id, nest_node_id,
                                  simulator.model.G[0]*
                                  tvb_nest_builder.tvb_weights,
                                  sigma=sigma),
#
#   To add to TVB connectivity delay:
    "delays": lambda tvb_node_id, nest_node_id:
        tvb_delay(tvb_node_id, nest_node_id, tvb_nest_builder.tvb_delays),
#
# -----TVB sv -> NEST population
    "connections": {"S_e": ["E"]},
    "source_nodes": None, "target_nodes": None} ] # None means all here
```



Use case

TVB to NEST coupling via direct parameter update



SP node n in TVB

Excitatory population
large-scale delayed coupling

$$S_n^{coup}(t) = G \sum_{\dot{n}} C_{nn} S_{\dot{n}}^E(t - \tau_{nn})$$

nest::

NEST
external current parameter

$$I_n^{ext}(t) = N_E J_{NMDA} S_n^{coup}(t)$$

$$I_n^{ext}(t) = N_E J_{NMDA} \lambda S_n^{coup}(t)$$

NEST
SP node n

Excitatory
population
 \dot{V}_n^m

Inhibitory
population
 \dot{V}_n^m



Use case

TVB to NEST coupling via direct parameter update

nest::

```
# For directly setting an external current parameter in NEST neurons instantaneously:  
tvb_nest_builder.tvb_to_nest_interfaces = [  
    {"model": "current", "parameter": "I_e",  
     # -----Properties potentially set as function handles with args (nest_node_id=None)-  
     "interface_weights": 1.0 * N_E,  
    # -----  
    #           TVB sv -> NEST population  
    "connections": {"S_e": ["E"]},  
    "nodes": None} ] # None means all here
```

$$I_n^{ext}(t) = N_E J_{NMDA} \lambda S_n^{coupl}(t)$$

HIBDITORY
population
 V_n^m



Human Brain Project

NEST to TVB update



THE VIRTUAL BRAIN

TVB Mean Field node n

Excitatory population
spike rate

$$Rin_n^E(t) = \frac{N_{\text{spikes}}^{\text{in } n}(t)}{dt N_E}$$

Inhibitory population
spike rate

$$Rin_n^I(t) = \frac{N_{\text{spikes}}^{\text{in } n}(t)}{dt N_I}$$

nest::

NEST spike detector devices

spike count

$$N_{\text{spikes}}^{\text{in } n}(t) = \sum_{i=1}^{N_E} \sum_{t^k \in (t-dt, t)} \delta(t - t_i^k)$$

Spikes
 $\delta(t - t_i^k)$

spike count

$$N_{\text{spikes}}^{\text{in } n}(t) = \sum_{i=1}^{N_I} \sum_{t^k \in (t-dt, t)} \delta(t - t_i^k)$$

Spikes
 $\delta(t - t_i^k)$

NEST Spiking Network node n

Excitatory
Population
 E

Inhibitory
Population
 I



Use case

nest ::

NEST

```
# NEST -> TVB:  
# Use S_e and S_i instead of r_e and r_i  
# for transmitting to the TVB state variables directly  
connections = OrderedDict()  
#           TVB <- NEST  
connections["R_e"] = ["E"]  
connections["R_i"] = ["I"]  
tvb_nest_builder.nest_to_tvb_interfaces = [  
    {"model": "spike_detector", "params": {}},  
# -----Properties potentially set as function handles with args (nest_node_id=None)---  
    "weights": 1.0, "delays": 0.0,  
# -----  
    "connections": connections, "nodes": None}] # None means all here
```

$$Rin_n^I(t) = \frac{N_{\text{spikes}}^{I \text{ in } n}(t)}{dtN_I}$$

$$N_{\text{spikes}}^{I \text{ in } n}(t) = \sum_{i=1}^{N_I} \sum_{t^k \in (t-dt, t)} \delta(t - t_i^k)$$

Population
I



Use case

NEST network builder: define neural populations

```
# Populations' configurations
# When any of the properties model, params and scale below depends on regions,
# set a handle to a function with
# arguments (region_index=None) returning the corresponding property
nest_model_builder.default_population["model"] = "aeif_cond_beta_multisynapse"
nest_model_builder.populations = [
    {"label": "E", "model": nest_model_builder.default_population["model"],
     "nodes": None, # None means "all"
     "params": nest_model_builder.params_ex,
     "scale": exc_pop_scale},
    {"label": "I", "model": nest_model_builder.default_population["model"],
     "nodes": None, # None means "all"
     "params": nest_model_builder.params_in,
     "scale": 1.0}
]
```



Use case

NEST network builder: define connections within regions

```
# Within region-node connections
# When any of the properties model, conn_spec, weight, delay, receptor_type below
# set a handle to a function with
# arguments (region_index=None) returning the corresponding property
nest_model_builder.populations_connections = [
    #           ->
    {"source": "E", "target": "E",  # E -> E This is a self-connection for population "E"
     "model": nest_model_builder.default_populations_connection["model"],
     "conn_spec": nest_model_builder.default_populations_connection["conn_spec"],
     "weight": nest_model_builder.tvb_model.w_p[0],
     "delay": nest_model_builder.default_populations_connection["delay"],
     "receptor_type": 1, "nodes": None},  # None means "all"
```



Use case

NEST network builder: define connections among regions

```
nest_model_builder.nodes_connections = [
    #             ->
    {"source": "E", "target": ["E"],
     "model": nest_model_builder.default_nodes_connection["model"],
     "conn_spec": nest_model_builder.default_nodes_connection["conn_spec"],
     # weight scaling the TVB connectivity weight
     "weight": lambda source_node, target_node:
                scale_tvb_weight(source_node, target_node,
                                  nest_model_builder.tvb_weights, scale=nest_model_builder.tvb_model.G[0]),
     # additional delay to the one of TVB connectivity
     "delay": lambda source_node, target_node:
                tvb_delay(source_node, target_node, nest_model_builder.tvb_delays),
     # Each region emits spikes in its own port:
     "receptor_type": lambda source_region_index, target_region_index=None:
                        receptor_by_source_region(source_region_index, target_region_index, start=3),
     "source_nodes": None, "target_nodes": None} # None means "all"
]
```



Use case

NEST network builder: stimulus input devices

```
# Create a spike stimulus input device
nest_model_builder.input_devices = [
    {"model": "poisson_generator",
     "params": {"rate": 2400.0, "origin": 0.0, "start": 0.1},
     "connections": {"Stimulus": ["E", "I"]},
     "nodes": None,           # None means apply to all
     "weights": w_E,
     "delays": {"distribution": "uniform",
                "low": nest_model_builder.tvb_dt,
                "high": 2*nest_model_builder.tvb_dt},
     "receptor_type": lambda target_node: int(target_node + 1)}
] #
```



Use case

NEST network builder: output measuring devices

```
# Creating devices to be able to observe NEST activity:  
# Labels have to be different  
nest_model_builder.output_devices = []  
connections = OrderedDict({})  
#           label <- target population  
connections["E"] = "E"  
connections["I"] = "I"  
nest_model_builder.output_devices.append(  
    {"model": "spike_detector", "params": {},  
     "connections": connections, "nodes": None}) # None means "all"  
  
connections = OrderedDict({})  
#           label      <- target population  
connections["Excitatory"] = "E"  
connections["Inhibitory"] = "I"  
params = dict(nest_model_builder.config.NEST_OUTPUT_DEVICES_PARAMS_DEF["multimeter"])  
params["interval"] = nest_model_builder.nest_instance.GetKernelStatus("resolution") #  
params['record_from'] = ["V_m"]  
nest_model_builder.output_devices.append(  
    {"model": "multimeter", "params": params,  
     "connections": connections, "nodes": None}) # None means "all"
```



Modifications to the TVB simulator

- TVB-SpikeNet interface properties and configurations
- Dynamical non-state variables (e.g. $R_n^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- Model specific modifications (e.g. $Rin_n^{E/I}(t)$) variables to smooth instantaneous spike rate coming from NEST via linear integration acting as a low pass filter)
- Integration loop
 1. TVB to SpikeNet coupling
 2. TVB integration
 3. SpikeNet integration
 3. (Optional) SpikeNet to TVB update
 4. Compute TVB large-scale coupling and update stimulus, if any
 5. Update non-state dynamical variables (if any)
 6. Update TVB history buffer
 7. Generate TVB monitor outputs

References

1. Ritter P, Schirner M, McIntosh AR, Jirsa VK. 2013. The Virtual Brain integrates computational modeling and multimodal neuroimaging. *Brain Connectivity* 3:121–145.
2. Sanz Leon P, Knock SA, Woodman MM, Domide L, Mersmann J, McIntosh AR, Jirsa V. 2013. The Virtual Brain: a simulator of primate brain network dynamics. *Frontiers in Neuroinformatics* 7:10.
3. Jordan, Jakob, Mørk, Håkon, Vennemo, Stine Brekke, Terhorst, Dennis, Peyser, Alexander, Ippen, Tammo, ... Plessner, Hans Ekkehard. (2019, June 27). NEST 2.18.0 (Version 2.18.0). Zenodo.
4. Deco G, Ponce-Alvarez A, Mantini D, Romani GL, Hagmann P, Corbetta M. 2013. Resting-State Functional Connectivity Emerges from Structurally and Dynamically Shaped Slow Linear Fluctuations. *The Journal of Neuroscience* 33(27): 11239 –11252.
5. Perdikis D, Schirner M, Diaz-Cortes, M-A, Domide L, Mersmann J, Ritter P (*in prep*).



Human Brain Project

THANK YOU!



www.humanbrainproject.eu



@humanbrainproj



@humanbrainproj



HumanBrainProject

Co-funded by
the European Union





Human Brain Project

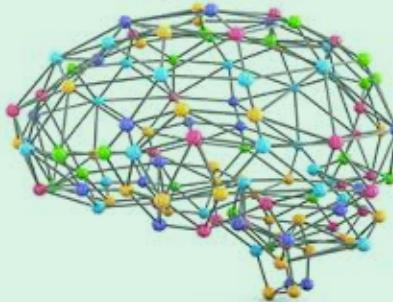
Motivation

TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.



THE VIRTUAL BRAIN.

Brain regions' level



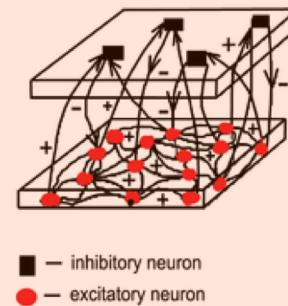
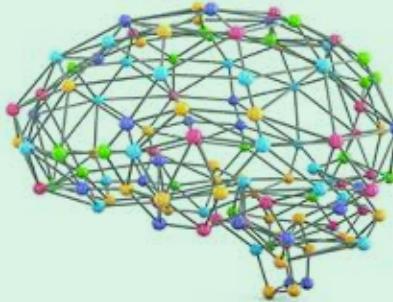


Motivation

TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.



THE VIRTUAL BRAIN.
Brain regions' level



Neurons' level
nest::



Motivation

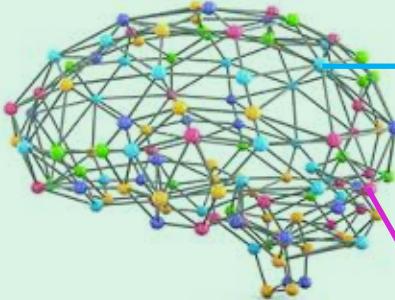
TVB-multiscale interfaces for multiscale co-simulation:

- TVB → Spiking simulators:
Generate biologically realistic spatio-temporal context and large-scale connectivity for local neural networks
- TVB ↔ Spiking simulators:
Confirm mean field approximations on large-scale brain statistics (e.g., functional connectivity)



THEVIRTUALBRAIN.

Brain regions' level



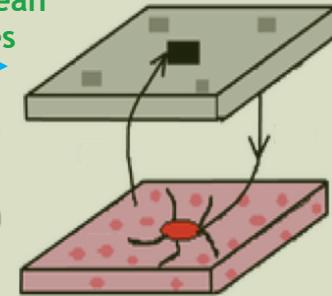
THEVIRTUALBRAIN.

nest::

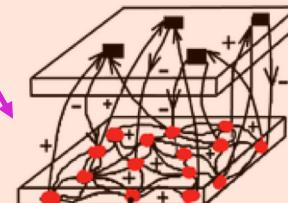
Populations' level

Populations' activity as mean field system state variables

- mean field (X_2) of the inhibitory subpopulation
- mean field (X_1) of the excitatory subpopulation



Average activity of populations of spiking neurons



- — inhibitory neuron
- — excitatory neuron

Neurons' level

nest::



Benchmarking

