Human Brain Project

3-6 February 2020 - Athens, Greece

Human Brain Project SUMMIT & OPEN DAY

Co-funded by
the European Union

Human Brain Project

TVB-multiscale:

interfacing TVB with NEST spiking networks for multiscale co-simulation

Dr. Dionysios Perdikis
Brain Simulation Section, Department of Neurology,
Charité—Universitätsmedizin Berlin

CHARITÉ
UNIVERSITÄTSMEDIZIN BERLIN

BRAIN
SIMULATION
SECTION

THEVIRTUALBRAIN.

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.

THEVIRTUALBRAIN.

Brain regions' level

Human Brain Project

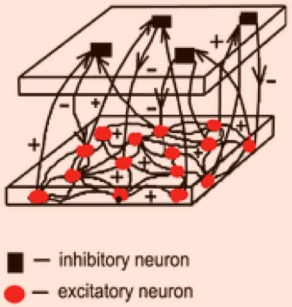
Motivation

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

THEVIRTUALBRAIN.

Brain regions' level

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



Neurons' level

nest ::

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

THEVIRTUALBRAIN.

Brain regions' level

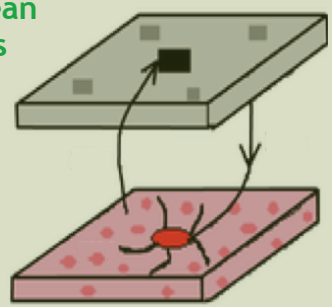
THEVIRTUALBRAIN. nest::

Populations' level

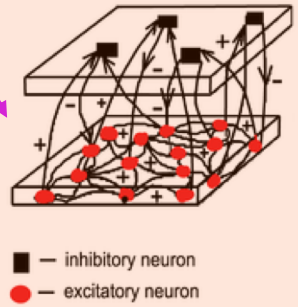
Populations' activity as mean field system state variables

– mean field (X2) of the inhibitory subpopulation

– mean field (X1) of the excitatory subpopulation



Average activity of populations of spiking neurons



Neurons' level

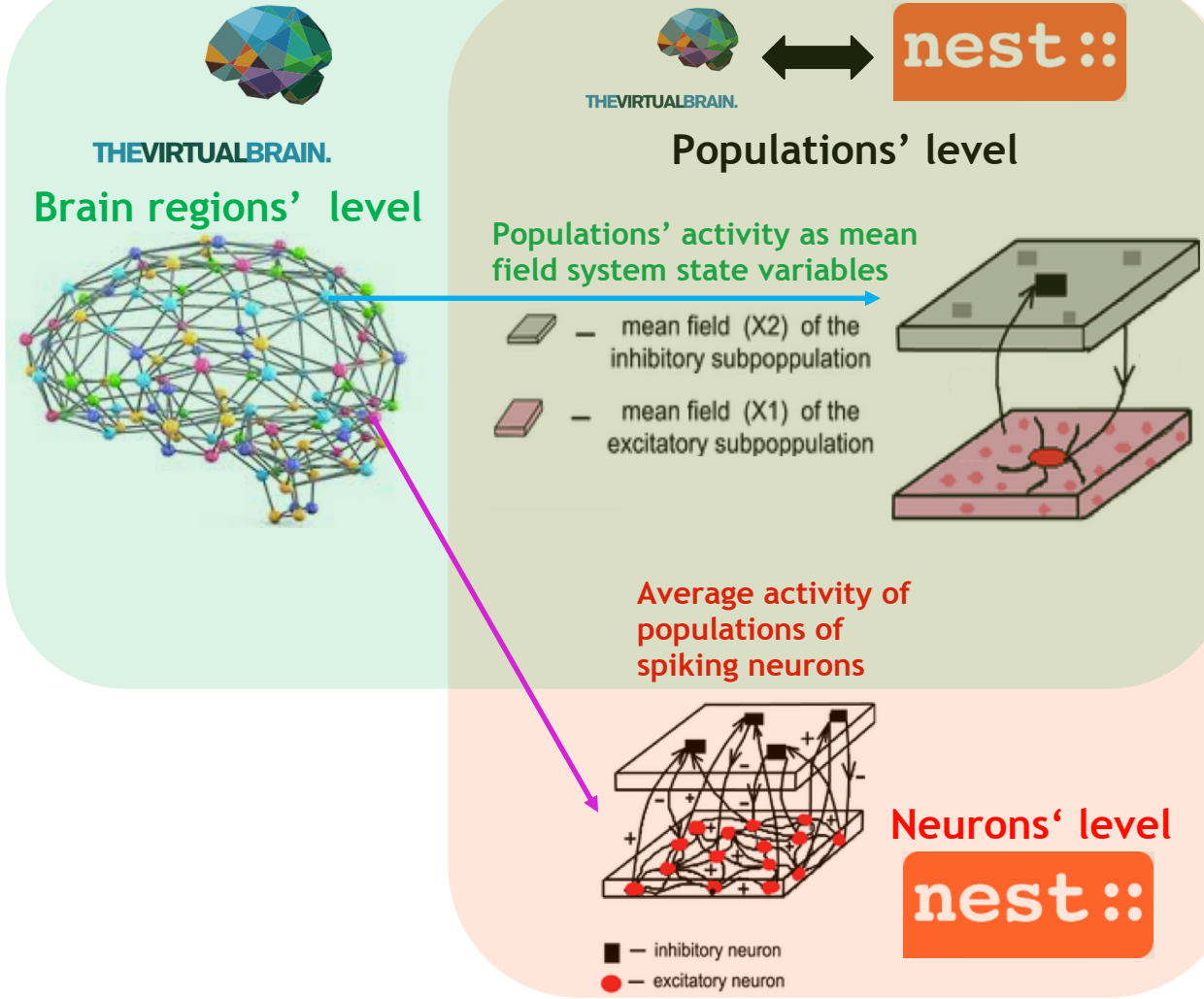
nest::

Human Brain Project

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)



Human Brain Project

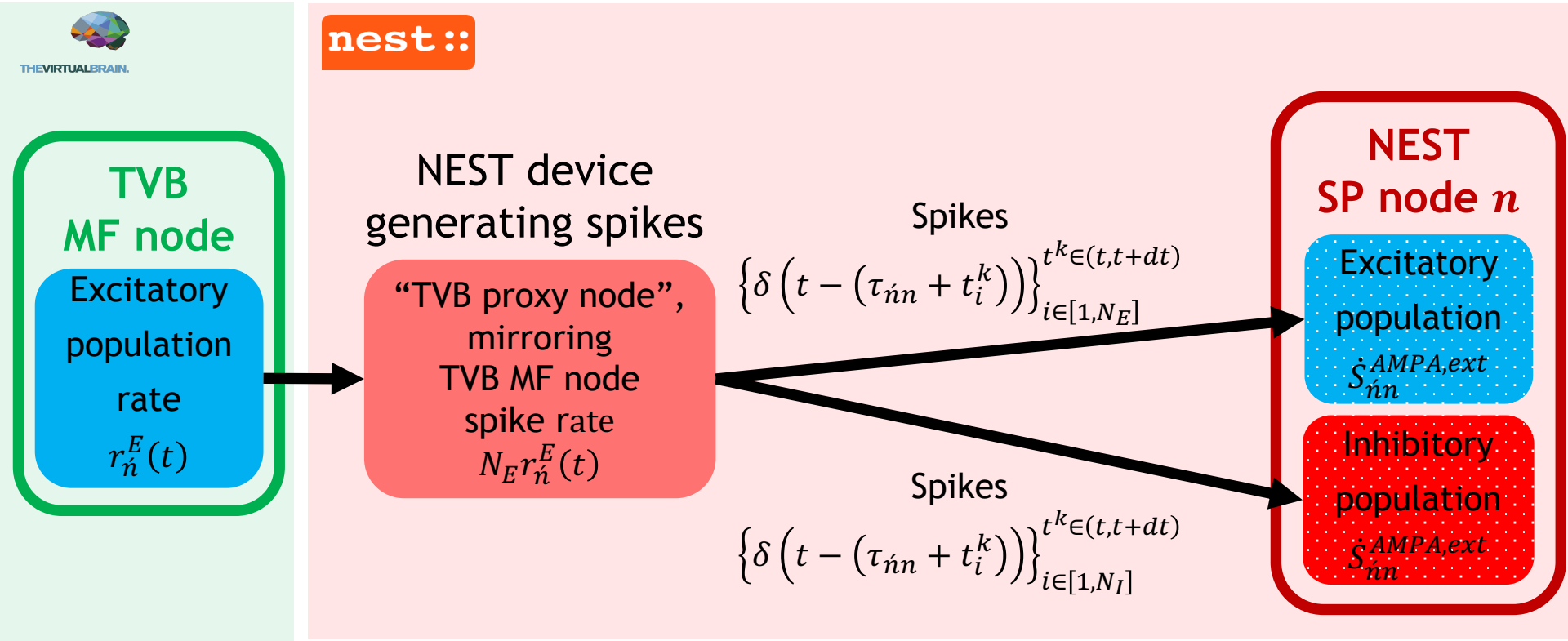
nest::

TVB-multiscale on HBP collaboratory

- ***TVB-multiscale:*** extends TVB to perform multiscale co-simulation.
- ***TVB-multiscale/tvb_nest:*** Python interface of TVB with (Py)NEST, for co-simulation of TVB and NEST models.
- **Use case in Human Brain Project Collaboratory:**
  - 1 excitatory (E) and 1 inhibitory (I) population per region node
  - Coarse-scale MF nodes in TVB with a reduced Wong-Wang model
  - 2 fine-scale spiking networks SP nodes (e.g., Left and Right Parahippocampal cortices), with a Wong-Wang like spiking neuron model programmed in NEST

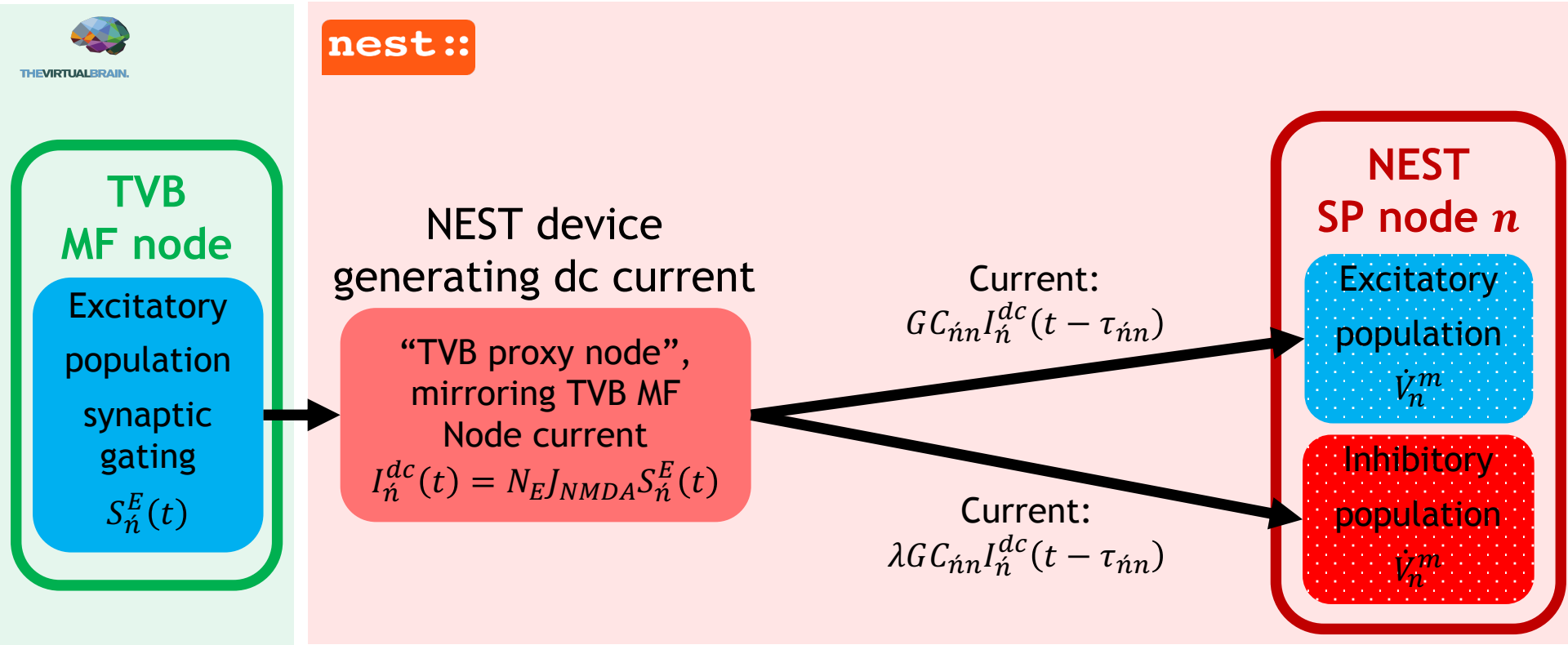
Use case

TVB to NEST coupling via TVB "proxy" nodes: spike rate



Use case

TVB to NEST coupling via TVB "proxy" nodes: current



Use case

TVB to NEST coupling via direct parameter update

THEVIRTUALBRAIN.

SP node n in TVB

Excitatory population

large-scale delayed coupling

$$S_n^{coupl}(t) = G \sum_{\acute{n}} C_{\acute{n}n} S_{\acute{n}}^{E}(t - \tau_{\acute{n}n})$$

nest::

NEST

external current parameter

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

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

NEST

SP node n

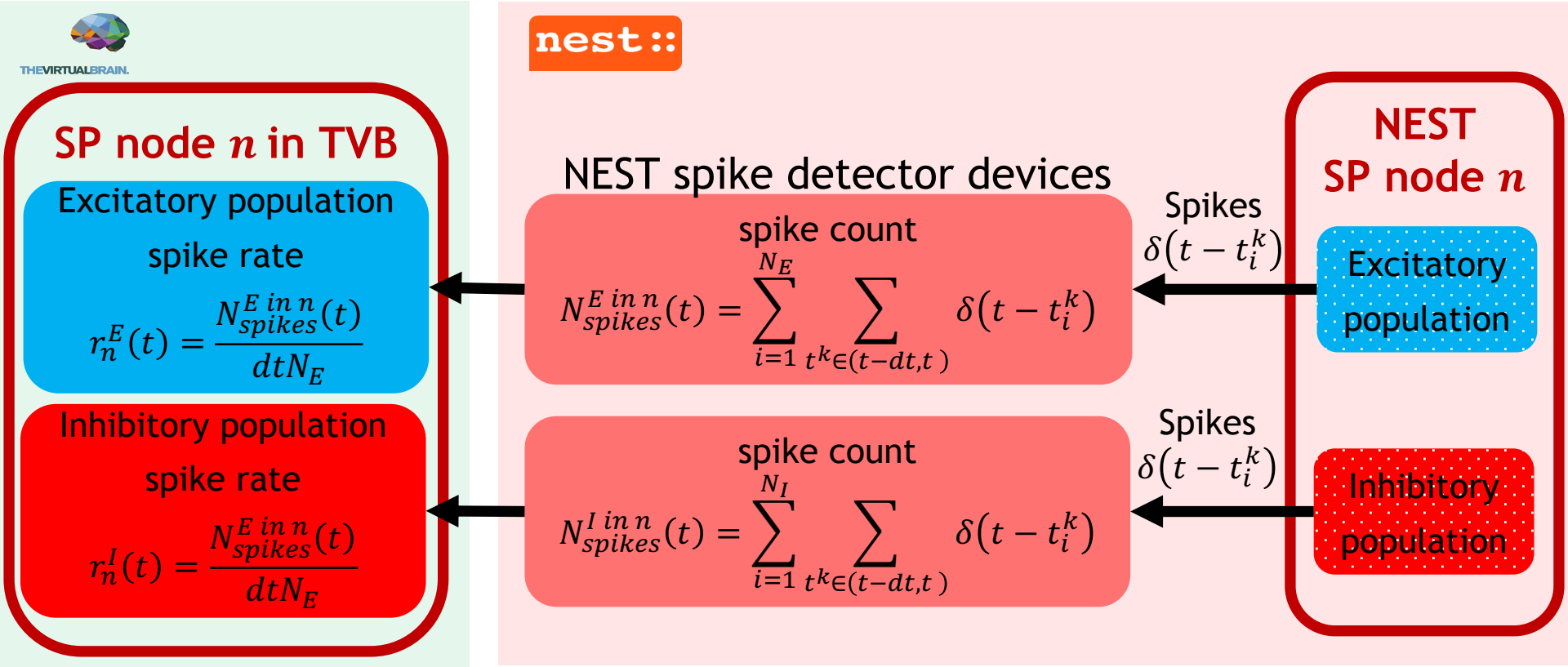
Excitatory population $\dot{V}_n^m$

Inhibitory population $\dot{V}_n^m$

Human Brain Project

Use case

NEST to TVB update



HBP collaboratory

Human Brain Project

nest::

COLLABORATORY HOME COLLABS PLATFORMS HELP FEEDBACK FORUM

TVB NEST Public Member

Navigation ADD

- Overview TVB-NEST
- Team
- Storage
- Settings
- Test TVB-NEST installation
- Run a custom cosimulation
- launch_example
- Run a notebook from storage

Workspace — Run a notebook from storage

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

Markdown

TVB-NEST co-simulation

We demonstrate a multiscale simulation of a reduced Wong-Wang model [1] using TVB [2] for brain region modelled at co and NEST [3] for regions modelled as networks of spiking neural populations.

In [1]:

```
# For interactive plotting:
%matplotlib notebook
! pip uninstall -y tvb
! pip install --upgrade tvb-libra
import time
import numpy as np
from collections import OrderedDi
from tvb.basic.profile import Tvb
TvbProfile.set_profile(TvbProfile
```

```
WARNING: Skipping tvb as it is no
Requirement already up-to-date: t
Requirement already up-to-date: h
Requirement already satisfied, sk
(0.19.1)
Requirement already satisfied, sk
(2.6.9)
Requirement already satisfied, sk
(0.30.0)
Requirement already satisfied, sk
ry) (2.0.2)
Requirement already satisfied, sk
(1.11.2)
Requirement already satisfied, sk
y) (2.3)
```

Human Brain Project, 2019

COLLABORATORY HOME COLLABS PLATFORMS HELP FEEDBACK FORUM

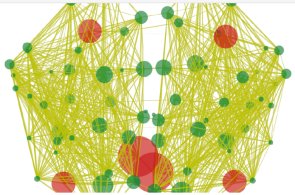
TVB NEST Public Member

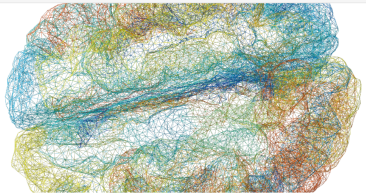
Navigation ADD

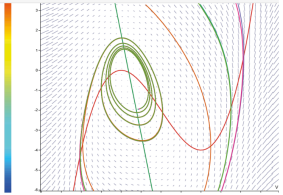
- Overview TVB-NEST
- Team
- Storage
- Settings
- Test TVB-NEST installation
- Run a custom cosimulation
- launch_example
- Run a notebook from storage

Workspace — Overview TVB-NEST

Description Activities Team



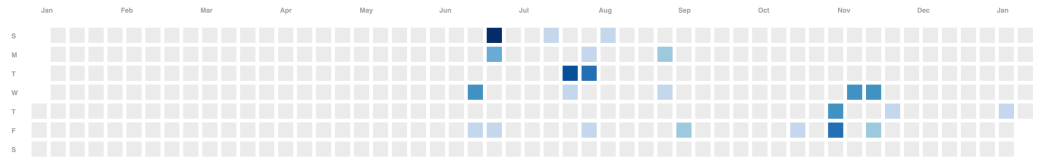




This is the HPC test-bed for this repository:

https://github.com/the-virtual-brain/tvb-multiscale

Recent Activities



7 days ago @paulapopa added new member @bvalean to the Collab TVB NEST

2 months @gabiflorea saved file HBPStorageFile <7be8edd1-690e-47ee-87df-42be9872d1a4> to TVB NEST

Collaboration

Human Brain Project, 2019 Cookie statement Terms of Service

Human Brain Project

THANK YOU!

www.humanbrainproject.eu

@humanbrainproj

@humanbrainproj

HumanBrainProject

Co-funded by
the European Union

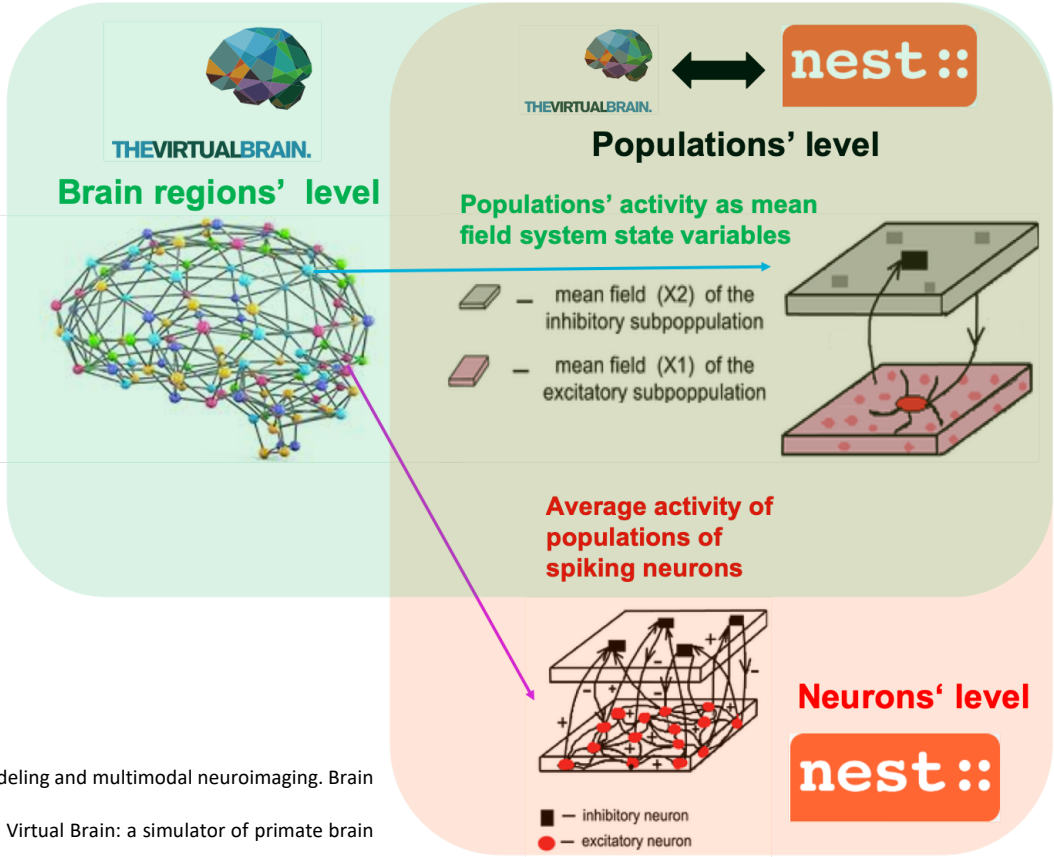
Motivation

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

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)

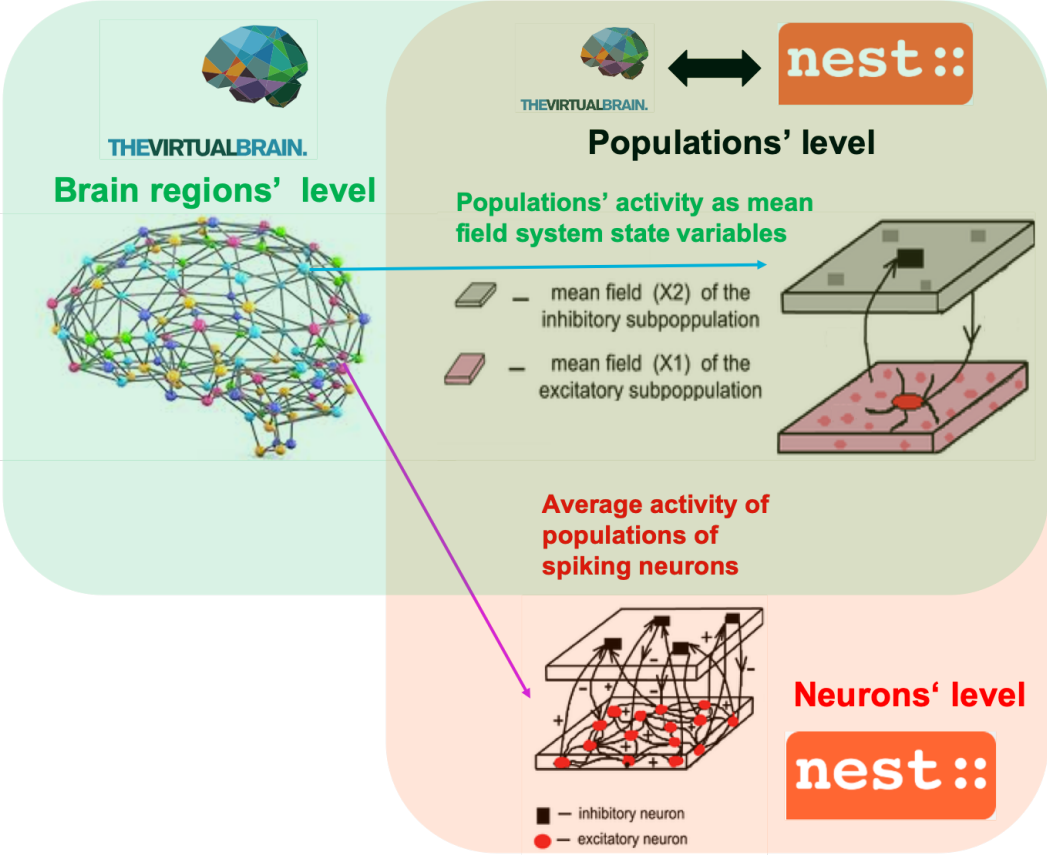


Ritter P, Schirner M, McIntosh AR, Jirsa VK. 2013. The Virtual Brain integrates computational modeling and multimodal neuroimaging. Brain Connectivity 3:121–145.

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.

Interface at the population level

- Populations' activity
  - mean field state variables
  - averaging of populations' spiking neurons
- Location at brain regions' nodes of the large-scale network:
  - Mean-field node (MF)
  - Spiking network node (SP)
- Parameters of the
  - mean field model
  - fine-scale spiking neurons' networks



TVB-multiscale on TVB github, Docker and HBP collaboratory

- *TVB-multiscale:* extends TVB to perform multiscale simulations.
- *TVB-multiscale/tvb_nest:* Python interface of TVB with (Py)NEST (a state-of-the-art platform for high-performance simulations of networks of spiking neurons), which performs co-simulation of TVB and NEST models.
- Proof of concept in Human Brain Project Collaboratory:
  - 1 excitatory (E) and 1 inhibitory (I) population per region node
  - Coarse-scale MF nodes modelled in TVB with a reduced Wong-Wang model
  - 2 fine-scale spiking networks SP nodes (e.g., Left and Right Parahippocampal cortices), with a Wong-Wang like spiking neuron model programmed in NEST

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S_{ń}^{E}} = -S_{ń}^{E}(t)/\tau_E + \left(1 - S_{ń}^{E}\right)\gamma r_{ń}^{E}(t)$$

$$\dot{S_{ń}^{I}} = -S_{ń}^{I}(t)/\tau_I + r_{ń}^{I}(t)$$

$$S_{ń}^{E/I} \in [0,1]$$

Rates:

$$r_{ń}^{E/I}(t) = \frac{a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}}{1 - e^{\left(d_{E/I}\left(a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}\right)\right)}}$$

Currents:

$$I_{ń}^{E}(t) = W_E I_0 + w_+ J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + G J_{NMDA} \sum_{ḿ \neq ń} C_{ńḿ}\, S_{ḿ}^{E}\left(t - \tau_{ńḿ}\right) + I_{ext}$$

$$I_{ń}^{I}(t) = W_I I_0 + J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + \lambda G J_{NMDA} \sum_{ḿ} C_{ńḿ}\, S_{ḿ}^{E}(t - \tau_{ńḿ})$$

$$\lambda \in [0,1]$$

Wong-Wang like "integrate-and-fire" spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t) - I_{ni}^{g_{AMPA,ext}}(t) - \lambda G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

$$= -g_m(V_{ni}^{m}(t) - V_L) - g_{AMPA}(V_{ni}^{m}(t) - V_E) w_E S_{ni}^{AMPA}(t) - \frac{g_{NMDA}(V_{ni}^{m}(t) - V_E)}{1 + \lambda_{NMDA} e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}(V_{ni}^{m}(t) - V_I) w_I S_{ni}^{GABA}(t) - g_{AMPA,ext}(V_{ni}^{m}(t) - V_E)\lambda G \sum_{ń} C_{ńn} S_{ńn,i}^{AMPA,ext}(t) - G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise} + \sum_{j=1}^{N_E} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ńn,i}^{AMPA,ext} = -S_{ńn,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_{k} \delta\left(t - \tau_{ńn} - t_j^k\right)\right]_{j \in ń}$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S}_{ń}^{E} = -S_{ń}^{E}(t)/\tau_E + \left(1 - S_{ń}^{E}\right)\gamma r_{ń}^{E}(t)$$

$$\dot{S}_{ń}^{I} = -S_{ń}^{I}(t)/\tau_I + r_{ń}^{I}(t)$$

$$S_{ń}^{E/I} \in [0,1]$$

Currents:

$$I_{ń}^{E}(t) = W_E I_0 + w_+ J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + G J_{NMDA} \sum_{ḿ \neq ń} C_{ńḿ}\, S_{ḿ}^{E}\left(t - \tau_{ńḿ}\right) + I_{ext}$$

$$I_{ń}^{I}(t) = W_I I_0 + J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + \lambda G J_{NMDA} \sum_{ḿ} C_{ńḿ}\, S_{ḿ}^{E}(t - \tau_{ńḿ})$$

$\lambda \in [0,1]$

Wong-Wang-like “integrate-and-fire” spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t)$$

$$I_{ni}^{g_{AMPA}}(t) = g_{AMPA,ext}(t) - \lambda G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

$$I_{ni}^{L}(t) = g_m(V_{ni}^{m}(t) - V_L) - g_{AMPA}(V_{ni}^{m}(t) - V_E) w_E S_{ni}^{AMPA}(t)$$

$$I_{ni}^{g_{NMDA}}(t) = \frac{g_{NMDA}(V_{ni}^{m}(t) - V_E)}{1 + \lambda_{NMDA} e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}(V_{ni}^{m}(t) - V_I) w_I S_{ni}^{GABA}(t)$$

$$I_{ni}^{g_{AMPA,ext}}(t) = g_{AMPA,ext}(V_{ni}^{m}(t) - V_E) \lambda G \sum_{ń} C_{ńn} S_{ńn,i}^{AMPA,ext}(t)$$

$$I_{ni}^{ext}(t) = \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_k \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise} + \sum_{j=1}^{N_E} \sum_k \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ńn,i}^{AMPA,ext} = -S_{ńn,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_k \delta\left(t - \tau_{ńn} - t_j^k\right)\right]_{j \in ń}$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S}_{ń}^{E} = -S_{ń}^{E}(t)/\tau_E + (1 - S_{ń}^{E})\gamma r_{ń}^{E}(t)$$

$$\dot{S}_{ń}^{I} = -S_{ń}^{I}(t)/\tau_I + r_{ń}^{I}(t)$$

$$S_{ń}^{E/I} \in [0,1]$$

Rates:

$$r_{ń}^{E/I}(t) = \frac{a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}}{1 - e^{\left(d_{E/I}\left(a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}\right)\right)}}$$

Currents:

$$I_{n}^{E}(t) = W_E I_0 + w_+ J_{NMDA} S_{n}^{E}(t) - J_{ń} S_{ń}^{I}(t) + G J_{NMDA} \sum_{ń \neq ń} C_{ńń} S_{ń}^{E}(t - \tau_{ńń}) + I_{ext}$$

$$I_{ń}^{I}(t) = W_I I_0 + J_{NMDA} S_{ń}^{E}(t) - J_{ń} S_{ń}^{I}(t) + \lambda G J_{NMDA} \sum_{ń} C_{ńń} S_{ń}^{E}(t - \tau_{ńń})$$

$$\lambda \in [0,1]$$

Wong-Wang like "integrate-and-fire" spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t) - I_{ni}^{g_{AMPA,ext}}(t) - \lambda G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

$$= -g_m(V_{ni}^{m}(t) - V_L) - g_{AMPA}(V_{ni}^{m}(t) - V_E) w_E S_{ni}^{AMPA}(t) - \frac{g_{NMDA}(V_{ni}^{m}(t) - V_E)}{1 + \lambda e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}(V_{ni}^{m}(t) - V_I) w_I S_{ni}^{GABA}(t) - g_{AMPA,ext}(V_{ni}^{m}(t) - V_E) \lambda G \sum_{ń} C_{ńn} S_{ńn,i}^{AMPA,ext}(t) - G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_{k} \delta(t - t_j^k)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise} + \sum_{j=1}^{N_E} \sum_{k} \delta(t - t_j^k)$$

$$\dot{S}_{ńn,i}^{AMPA,ext} = -S_{ńn,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_{k} \delta(t - \tau_{ńn} - t_j^k)\right]_{j \in ń}$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S}_{\acute{n}}^{E} = -S_{\acute{n}}^{E}(t)/\tau_E + (1 - S_{\acute{n}}^{E})\gamma r_{\acute{n}}^{E}(t)$$

Currents:

$$I_{\acute{n}}^{E}(t) = W_E I_0 + w_+ J_{NMDA}\, S_{\acute{n}}^{E}(t) - J_{\acute{n}}\, S_{\acute{n}}^{I}(t) + G J_{NMDA} \sum_{\acute{m} \neq \acute{n}} C_{\acute{n}\acute{m}}\, S_{\acute{m}}^{E}\left(t - \tau_{\acute{n}\acute{m}}\right) + I_{ext}$$

$$I_{\acute{n}}^{I}(t) = W_I I_0 + J_{NMDA}\, S_{\acute{n}}^{E}(t) - J_{\acute{n}}\, S_{\acute{n}}^{I}(t) + \lambda G J_{NMDA} \sum_{\acute{m}} C_{\acute{n}\acute{m}}\, S_{\acute{m}}^{E}\left(t - \tau_{\acute{n}\acute{m}}\right)$$

$$\lambda \in [0,1]$$

Rates:

$$r_{\acute{n}}^{E/I}(t) = \frac{a_{E/I}\left(I_{\acute{n}}^{E/I}(t) - b_{E/I}\right)}{1 - e^{-d_{E/I}\left(I_{\acute{n}}^{E/I}(t) - b_{E/I}\right)}}$$

Synaptic gating variables:

$$\dot{S}_{\acute{n}}^{E/I} = -S_{\acute{n}}^{E/I}(t)/\tau_{E/I} + \gamma r_{\acute{n}}^{E/I}(t) \qquad S_{\acute{n}}^{E/I} \in [0,1]$$

Wong-Wang like "integrate-and-fire" spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t)$$

$$\sum C_{\acute{n}n} I_{\acute{n}i}^{dc}(t - \tau_{\acute{n}n}) + I_{ni}^{ext}(t)$$

$$_{MPA}(V_{ni}^{m}(t) - V_E) w_E S_{ni}^{AMPA}(t)$$

$$_{i}^{MDA}(t) - g_{GABA}(V_{ni}^{m}(t) - V_I) w_I S_{ni}^{GABA}(t)$$

$$)\lambda G \sum_{\acute{n}} C_{\acute{n}n} S_{\acute{n}n,i}^{AMPA,ext}(t)$$

$$+ I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_{k} \delta\left(t - t_j^k\right)$$

$$_{y} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$+ \sum_{j=1}^{N_E} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise}$$

$$_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_{k} \delta\left(t - \tau_{\acute{n}n} - t_j^k\right)\right]_{j \in \acute{n}}$$

$$\dot{S}_{\acute{n}n,i}^{AMPA,ext} = -S_{\acute{n}n,i}^{AMPA,ext}(t)/\tau_{AMPA,decay} + \sum_{k} \delta\left(t - t_{\acute{n}i}^{k}\right)$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced M

Synaptic

$S_{ń}^{E} = -$

$S_{ń}^{I} = -$

$S_{ń}^{E/I} \in [$

Rates:

$r_{ń}^{E/I}(t)$

Currents

$I_{ń}^{E}(t) =$

$I_{ń}^{I}(t) =$

$\lambda \in [0,1]$

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t)$$

$$-I_{ni}^{g_{AMPA,ext}}(t) - \lambda G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

$$= -g_m (V_{ni}^{m}(t) - V_L) - g_{AMPA}(V_{ni}^{m}(t) - V_E) w_E S_{ni}^{AMPA}(t)$$

$$- \frac{g_{NMDA}(V_{ni}^{m}(t) - V_E)}{1 + \lambda_{NMDA} e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}(V_{ni}^{m}(t) - V_I) w_I S_{ni}^{GABA}(t)$$

$$- g_{AMPA,ext}(V_{ni}^{m}(t) - V_E) \lambda G \sum_{ń} C_{ńn} S_{ńn,i}^{AMPA,ext}(t)$$

$$- G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_k \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise} + \sum_{j=1}^{N_E} \sum_k \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ńn,i}^{AMPA,ext} = -S_{ńn,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_k \delta\left(t - \tau_{ńn} - t_j^k\right)\right]_{j \in ń}$$

Rates:

$r_ń^{E/I}(t)$

Currents:

$I_ń^E(t) = V$

$+$

$I_ń^I(t) = V$

$+\lambda$

$\lambda \in [0,1]$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S}_{ń}^{E} = -S_{ń}^{E}(t)/\tau_E + \left(1 - S_{ń}^{E}\right)\gamma r_{ń}^{E}(t)$$

$$\dot{S}_{ń}^{I} = -S_{ń}^{I}(t)/\tau_I + r_{ń}^{I}(t)$$

$$S_{ń}^{E/I} \in [0,1]$$

Rates:

$$r_{ń}^{E/I}(t) = \frac{a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}}{1 - e^{\left(d_{E/I}\left(a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}\right)\right)}}$$

Currents:

$$I_{ń}^{E}(t) = W_E I_0 + w_+ J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + G J_{NMDA} \sum_{ḿ \neq ń} C_{ńḿ}\, S_{ḿ}^{E}\left(t - \tau_{ńḿ}\right) + I_{ext}$$

$$I_{ń}^{I}(t) = W_I I_0 + J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + \lambda G J_{NMDA} \sum_{ḿ} C_{ńḿ}\, S_{ḿ}^{E}(t - \tau_{ńḿ})$$

$$\lambda \in [0,1]$$

Wong-Wang like "integrate-and-fire" spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t) - I_{ni}^{g_{AMPA,ext}}(t) - \lambda G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

$$= -g_m\left(V_{ni}^{m}(t) - V_L\right) - g_{AMPA}\left(V_{ni}^{m}(t) - V_E\right) w_E S_{ni}^{AMPA}(t) - \frac{g_{NMDA}\left(V_{ni}^{m}(t) - V_E\right)}{1 + \lambda_{NMDA} e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}\left(V_{ni}^{m}(t) - V_I\right) w_I S_{ni}^{GABA}(t) - g_{AMPA,ext}\left(V_{ni}^{m}(t) - V_E\right) \lambda G \sum_{ń} C_{ńn} S_{ńn,i}^{AMPA,ext}(t) - G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise} + \sum_{j=1}^{N_E} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ńn,i}^{AMPA,ext} = -S_{ńn,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_{k} \delta\left(t - \tau_{ńn} - t_j^k\right)\right]_{j \in ń}$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S}_{ń}^{E} = -S_{ń}^{E}(t)/\tau_E + \left(1 - S_{ń}^{E}\right)\gamma r_{ń}^{E}(t)$$

$$\dot{S}_{ń}^{I} = -S_{ń}^{I}(t)/\tau_I + r_{ń}^{I}(t)$$

$$S_{ń}^{E/I} \in [0,1]$$

Rates:

$$r_{ń}^{E/I}(t) = \frac{a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}}{1 - e^{\left(d_{E/I}\left(a_{E/I} I_{ń}^{E/I}(t) - b_{E/I}\right)\right)}}$$

Currents:

$$I_{ń}^{E}(t) = W_E I_0 + w_+ J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + G J_{NMDA} \sum_{ḿ \neq ń} C_{ńḿ}\, S_{ḿ}^{E}\left(t - \tau_{ńḿ}\right) + I_{ext}$$

$$I_{ń}^{I}(t) = W_I I_0 + J_{NMDA}\, S_{ń}^{E}(t) - J_{ń}\, S_{ń}^{I}(t) + \lambda G J_{NMDA} \sum_{ḿ} C_{ńḿ}\, S_{ḿ}^{E}(t - \tau_{ńḿ})$$

$$\lambda \in [0,1]$$

Wong-Wang like "integrate-and-fire" spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t)$$

$$-I_{ni}^{g_{AMPA,ext}}(t) - \lambda G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

$$= -g_m(V_{ni}^{m}(t) - V_L) - g_{AMPA}(V_{ni}^{m}(t) - V_E) w_E S_{ni}^{AMPA}(t)$$

$$- \frac{g_{NMDA}(V_{ni}^{m}(t) - V_E)}{1 + \lambda_{NMDA} e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}(V_{ni}^{m}(t) - V_I) w_I S_{ni}^{GABA}(t)$$

$$- g_{AMPA,ext}(V_{ni}^{m}(t) - V_E) \lambda G \sum_{ń} C_{ńn} S_{ńn,i}^{AMPA,ext}(t)$$

$$- G \sum_{ń} C_{ńn} I_{ńi}^{dc}(t - \tau_{ńn}) + I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{I_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA\,rise} + \sum_{j=1}^{N_E} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ńn,i}^{AMPA,ext} = -S_{ńn,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_{k} \delta\left(t - \tau_{ńn} - t_j^k\right)\right]_{j \in ń}$$

Human Brain Project

Demonstration on Wong-Wang model

Reduced Mean Field model

Synaptic gating state variables:

$$\dot{S}_{\acute{n}}^{E} = -S_{\acute{n}}^{E}(t)/\tau_E + \left(1 - S_{\acute{n}}^{E}\right)\gamma r_{\acute{n}}^{E}(t)$$

$$\dot{S}_{\acute{n}}^{I} = -S_{\acute{n}}^{I}(t)/\tau_I + r_{\acute{n}}^{I}(t)$$

$$S_{\acute{n}}^{E/I} \in [0,1]$$

Rates:

$$r_{\acute{n}}^{E/I}(t) = \frac{a_{E/I} I_{\acute{n}}^{E/I}(t) - b_{E/I}}{1 - e^{\left(d_{E/I}\left(a_{E/I} I_{\acute{n}}^{E/I}(t) - b_{E/I}\right)\right)}}$$

Currents:

$$I_{\acute{n}}^{E}(t) = W_E I_0 + w_+ J_{NMDA}\, S_{\acute{n}}^{E}(t) - J_{\acute{n}}\, S_{\acute{n}}^{I}(t) + G J_{NMDA} \sum_{\acute{m} \neq \acute{n}} C_{\acute{n}\acute{m}}\, S_{\acute{m}}^{E}\left(t - \tau_{\acute{n}\acute{m}}\right) + I_{ext}$$

$$I_{\acute{n}}^{I}(t) = W_I I_0 + J_{NMDA}\, S_{\acute{n}}^{E}(t) - J_{\acute{n}}\, S_{\acute{n}}^{I}(t) + \lambda G J_{NMDA} \sum_{\acute{m}} C_{\acute{n}\acute{m}}\, S_{\acute{m}}^{E}\left(t - \tau_{\acute{n}\acute{m}}\right)$$

$$\lambda \in [0,1]$$

Wong-Wang like "integrate-and-fire" spiking neuron model

Membrane potential:

$$C_m \dot{V}_{ni}^{m} = -I_{ni}^{L}(t) - I_{ni}^{g_{AMPA}}(t) - I_{ni}^{g_{NMDA}}(t) - I_{ni}^{g_{GABA}}(t) - I_{ni}^{g_{AMPA,ext}}(t) - \lambda G \sum_{\acute{n}} C_{\acute{n}n} I_{\acute{n}i}^{dc}(t - \tau_{\acute{n}n}) + I_{ni}^{ext}(t)$$

$$= -g_m\left(V_{ni}^{m}(t) - V_L\right) - g_{AMPA}\left(V_{ni}^{m}(t) - V_E\right) w_E S_{ni}^{AMPA}(t) - \frac{g_{NMDA}\left(V_{ni}^{m}(t) - V_E\right)}{1 + \lambda_{NMDA} e^{-\beta V_{ni}^{m}(t)}} w_E S_{ni}^{NMDA}(t) - g_{GABA}\left(V_{ni}^{m}(t) - V_I\right) w_I S_{ni}^{GABA}(t) - g_{AMPA,ext}\left(V_{ni}^{m}(t) - V_E\right) \lambda G \sum_{\acute{n}} C_{\acute{n}n} S_{\acute{n}n,i}^{AMPA,ext}(t) - G \sum_{\acute{n}} C_{nn} I_{\acute{n}i}^{dc}(t - \tau_{\acute{n}n}) + I_{ni}^{ext}(t)$$

Synaptic gating variables:

$$\dot{S}_{ni}^{AMPA/GABA} = -S_{ni}^{AMPA/GABA}(t)/\tau_{AMPA/GABA,decay} + \sum_{j=1}^{N_{E/I}} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{ni}^{NMDA} = -S_{ni}^{NMDA}(t)/\tau_{NMDA,decay} + \alpha X_{ni}^{NMDA}(t)\left(1 - \frac{1}{N_E - 1} S_{ni}^{NMDA}(t)\right)$$

$$\dot{X}_{ni}^{NMDA} = -X_{ni}^{NMDA}(t)/\tau_{NMDA,rise} + \sum_{j=1}^{N_E} \sum_{k} \delta\left(t - t_j^k\right)$$

$$\dot{S}_{\acute{n}n,i}^{AMPA,ext} = -S_{\acute{n}n,i}^{AMPA,ext}(t)/\tau_{NMDA,decay} + \left[\sum_{j=1}^{N_E} \sum_{k} \delta\left(t - \tau_{\acute{n}n} - t_j^k\right)\right]_{j \in \acute{n}}$$

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\dot{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeNet coupling
  2. TVB integration

     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
  6. Update TVB history buffer
  7. Generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # Integrate TVB to get the new TVB state
    state = self.integrator.scheme(state, self.model.dfun, node_coupling, local_coupling, stimulus)
    if numpy.any(numpy.isnan(state)) or numpy.any(numpy.isinf(state)):
        raise ValueError("NaN or Inf values detected in simulator state!:\n%s" % str(state))
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    if updateTVBstateFromSpikeNet:
        # SpikeNet state -> TVB state
        state = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_state(state)
        self.bound_and_clamp(state)
    # Prepare coupling and stimulus for next time step
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Update any non-state variables and apply any boundaries again to the new state:
    self.update_state(state, node_coupling, local_coupling)
    # Now direct the new state to history buffer and monitors
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\dot{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeN
  2. TVB integration

     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
  6. Update TVB history buffer
  7. Generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    if updateTVBstateFromSpikeNet:
        # SpikeNet state -> TVB state
        state = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_state(state)
        self.bound_and_clamp(state)
    # Prepare coupling and stimulus for next time step
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Update any non-state variables and apply any boundaries again to the new state:
    self.update_state(state, node_coupling, local_coupling)
    # Now direct the new state to history buffer and monitors
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\dot{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeNet coupling
  2. TVB integration
     SpikeNet integration
  3. (Optional) SpikeNet to TV
  4. Compute TVB large-scale coupling and update stimulus, if any
  5. Update non-state dynamical variables
  6. Update TVB history buffer
  7. Generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # Integrate TVB to get the new TVB state
    state = self.integrator.scheme(state, self.model.dfun, node_coupling, local_coupling, stimulus)
    if numpy.any(numpy.isnan(state)) or numpy.any(numpy.isinf(state)):
        raise ValueError("NaN or Inf values detected in simulator state!:\n%s" % str(state))
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    self.bound_and_clamp(state)
    # Prepare coupling and stimulus for next time step
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Update any non-state variables and apply any boundaries again to the new state:
    self.update_state(state, node_coupling, local_coupling)
    # Now direct the new state to history buffer and monitors
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\acute{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeNet coupling
  2. TVB integration

     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
  6. Update TVB history buffer
  7. Generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # Integrate TVB to get the new TVB state
    state = self.integrator.scheme(state, self.model.dfun, node_coupling, local_coupling, stimulus)
    if numpy.any(numpy.isnan(state)) or numpy.any(numpy.isinf(state)):
        raise ValueError("NaN or Inf values detected in simulator state!:\n%s" % str(state))
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    if updateTVBstateFromSpikeNet:
        # SpikeNet state -> TVB state
        state = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_state(state)
        self.bound_and_clamp(state)
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Update any non-state variables and apply any boundaries again to the new state:
    self.update_state(state, node_coupling, local_coupling)
    # Now direct the new state to history buffer and monitors
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\dot{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeNet coupling
  2. TVB integration

     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
  6. Update TVB history buffer
  7. Generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # Integrate TVB to get the new TVB state
    state = self.integrator.scheme(state, self.model.dfun, node_coupling, local_coupling, stimulus)
    if numpy.any(numpy.isnan(state)) or numpy.any(numpy.isinf(state)):
        raise ValueError("NaN or Inf values detected in simulator state!:\n%s" % str(state))
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    if updateTVBstateFromSpikeNet:
        # SpikeNet state -> TVB state
        state = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_state(state)
    # Prepare coupling and stimulus for next time step
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Update any non-state variables and apply any boundaries again to the new state:
    self.update_state(state, node_coupling, local_coupling)
    # Now direct the new state to history buffer and monitors
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Human Brain Project

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\acute{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeNet coupling
  2. TVB integration

     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
  6. Update TVB history buffer
  7. Generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # Integrate TVB to get the new TVB state
    state = self.integrator.scheme(state, self.model.dfun, node_coupling, local_coupling, stimulus)
    if numpy.any(numpy.isnan(state)) or numpy.any(numpy.isinf(state)):
        raise ValueError("NaN or Inf values detected in simulator state!:\n%s" % str(state))
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    if updateTVBstateFromSpikeNet:
        # SpikeNet state -> TVB state
        state = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_state(state)
        self.bound_and_clamp(state)
    # Prepare coupling and stimulus for next time step
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Update any non-state variables and apply any boundaries again to the new state:
    self.update_state(state, node_coupling, local_coupling)
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Modifications to TVB simulator

- Dynamical non-state variables (e.g. $r_{\dot{n}}^{E/I}(t)$) that need to be updated in a model specific manner after integration of each time step
- TVB-SpikeNet interface properties
- Integration loop
  1. TVB to SpikeNet coupling
  2. TVB integration

     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
  6. Update TVB history buffer and generate TVB monitor outputs

```
for step in range(self.current_step + 1,  self.current_step + n_steps + 1):
    # TVB state -> SpikeNet (state or parameter)
    self.tvb_spikeNet_interface.tvb_state_to_spikeNet(state, node_coupling+local_coupling, stimulus, self.model)
    # SpikeNet state -> TVB model parameter
    self.model = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_parameter(self.model)
    # Integrate TVB to get the new TVB state
    state = self.integrator.scheme(state, self.model.dfun, node_coupling, local_coupling, stimulus)
    if numpy.any(numpy.isnan(state)) or numpy.any(numpy.isinf(state)):
        raise ValueError("NaN or Inf values detected in simulator state!:\n%s" % str(state))
    # Integrate Spiking Network to get the new Spiking Network state
    self.run_spiking_simulator(self.integrator.dt)
    if updateTVBstateFromSpikeNet:
        # SpikeNet state -> TVB state
        state = self.tvb_spikeNet_interface.spikeNet_state_to_tvb_state(state)
        self.bound_and_clamp(state)
    # Prepare coupling and stimulus for next time step
    node_coupling = self._loop_compute_node_coupling(step)
    self._loop_update_stimulus(step, stimulus)
    # Now direct the new state to history buffer and monitors
    self._loop_update_history(step, n_reg, state)
    output = self._loop_monitor_output(step, state)
    if output is not None:
        yield output
```

Human Brain Project

Future work

- Validation and comparative analysis of the different coupling schemes between the coarse (MF)- and fine- scale (SP) models.
- Development of parameter calibration methods for the coupling between MF and SP models for biologically meaningful dynamics to emerge, starting from a relatively simple network toy-model, up to a full brain model with realistic anatomy. We will target parameters such as spiking population size, local coupling among spiking populations, large-scale coupling scaling. To this end, spiking and mean field regions' nodes output will be compared statistically.
- Optimization of TVB-NEST co-simulation to increase speed as well as parallelization and optimal use of HPC hardware resources.
- Improve usability of TVB-multiscale in CSCS cluster and HBP Collaboratory.
- Interface to other spiking neural network simulators (e.g., Neuron, PyNN).
- Use the toolbox in a real-world scientific problem (e.g., modeling memory consolidation networks).

Human Brain Project

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.

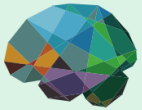
2. Sanz Leon P, Knock SA, Woodman MM, Domide L, Mersmann J, McIntosh AR, Jirsa V. 2013. The Virtual Brain: a simulator of primate brain network dynamics. Frontiers in Neuroinformatics 7:10.

3. Jordan, Jakob, Mørk, Håkon, Vennemo, Stine Brekke, Terhorst, Dennis, Peyser, Alexander, Ippen, Tammo, … Plesser, 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 Pi



THEVIRTUALBRAIN.

Brain regions' level

THEVIRTUALBRAIN.

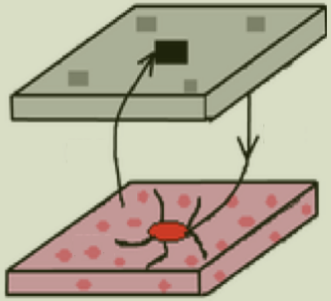
nest ::

Populations' level

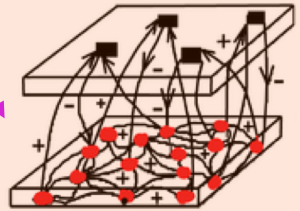
Populations' activity as mean field system state variables

– mean field (X2) of the inhibitory subpoppulation

– mean field (X1) of the excitatory subpoppulation



Average activity of populations of spiking neurons



– inhibitory neuron

– excitatory neuron

Neurons' level

nest ::