FNS user manual

Website: www.fnsneuralsimulator.org
GitHub: https://github.com/fnsneuralsimulator
Reference paper: “FNS: an event-driven spiking neural network framework for efficient simulations of large-scale brain models” (2018, under review); arXiv link

1. Software installation

1. Install *Java (SE) JDK* from the JAVA official page (add the bin directory of JDK to the PATH environment variable, and check JAVA_HOME environment variable points to the *jdk* folder instead of the *jre* one).
2. Download and install MAVEN from the MAVEN official page (add also the bin directory of MAVEN to the PATH environment variable).
   For Linux users Maven can be also downloaded from the official distribution repository.
3. Download the FNS package from the FNS GitHub repository and unzip it.
4. In *powershell*, go to the FNS root and re-compile by running `.\compile.bat` (it is recommended to keep your computer connected to the internet).[1]
   Linux users can re-compile by running `./compile` .

2. Simulation process

1. A new simulation can be launched by typing the so-called *experiment command*, which basic form is characterized by five *fields*:
   ```
   .\start.bat
   .\configuration_folders\simulation_name\configuration_file.xml
   .\configuration_folders\simulation_name\interconnectivity_set
   simulation_name
   mask
   ```
   written in the same line and separated by a space. Then press *enter* to start the simulation.
   An in-depth description of the *experiment command* is given in Sect.3.
   Linux users can launch the simulation by typing:
   ```
   ./start.bat
   configuration_folders/simulation_name/configuration_file.xml
   configuration_folders/simulation_name/interconnectivity_set
   simulation_name
   mask
   ```

2. During the simulation, FNS displays 3 sets of simulation data:
   Initialization. On the basis of *configuration file* and *interconnectivity set*, *nodes* (regions) and *edges* (fibre tracts) of the network are created, and the *opaque period* value is computed. Inner states of all neurons are initialized to random and subthreshold values (uniformly distributed between 0 and 1).

[1] This step is also necessary after the upgrading to a new version of FNS.

Execution. The program proceeds with the simulation of consecutive temporal slices, showing information on the current simulated cycle:
- cycle number;
- upper bound of the simulation interval (in terms of simulated time).

Simulation stats. At the end of the overall simulation the following simulation data are shown to the user:

- overall duration of the simulation;
- time employed by the single *initialization* procedures:
  - reading of the *interconnectivity_set*
  - reading of the *configuration_file*
  - synthesis of nodes
  - generation of inter-node connections
  - time employed by the *initialization* phase
- minimum tract length among all the inter-node connections (intra-node connections are instantaneous);
- duration of the cycle time (in FNS v.1.0.1 it corresponds to the BOP);
- total number of inter node connections;
- number of active and passive neurons at the end of the simulation;
- number of *passive-to-active* and *active-to-passive* transitions happened during the simulation;
- min and max *Ne_en_ratio*: minimum and maximum number of *Ne_en_ratio* (see par. 3.3) among all the the edges of the network;
- number of missed fires, i.e., fires that have been discarded due to an unsuitable sizing of the BOP. Note that, in order to not have missed fires, a slightly smaller value than the *BOP* is set by default as *cycle time*;
- curve goodness, referred to the gamma distribution generated for the *intra-node connection lengths*. A curve is considered *good* if no negative values are generated. In case of generation of negative values, the distribution *set* is recalculated for 10 times, searching for a set of only positive values, notifying the output message “good curve”. If a complete set of positive values is not achieved, the execution is terminated, and the output message"curve rejected" is notified.

Note that, differently to what happens for the (gamma) *intra-node length distributions*, the (Gaussian) *intra-node weight distributions* do not follow the same synthesis procedure: possible negative values are directly rectified by changing their sign.

3. At the end of the simulation FNS displays the *rasterplot* of the *firing activity*. The user can click to the fire events to obtain information on spike time and neuron order. Note that group external inputs are depicted above the last neuron of the group.

4. FNS saves information about the activity of the network in the *experiments/simulation_name*. Network activity in the simulation interval is reported in two .CSV files, for spiking and burning activity, respectively:
   - mask_*code*_**burning.csv** : Here we collect burning events in which neurons pertaining to selected *node of interests* (NOIs) participate as receiver[2].
   - mask_*code*_**firing.csv**: Here we collect firing events in which neurons pertaining to selected *node of interests* (NOIs) participate as transmitter.

   For the external inputs, the firing neuron number is not meaningful: considering a node composed of n neurons [*0* to *n-1*], its external inputs will be regarded as pertaining to the same node, but with a *neuron id* starting from *n*.

[2] Note that in the burning csv file not all the burning times are listed in increasing time order, due to the specific BOP-based technique adopted (see the refetrence paper of FNS for details).

3. Create an experiment

Here we describe the five fields the *experiment command* is articulated:

1) **start.bat**. This file is the starter. It must always be present at the beginning of the *experiment command* in such form.

2) **configuration_file.xml**. This file includes all the information the simulator necessitates for the synthesis of the network nodes (Fig.1, in blue), and some global parameters.

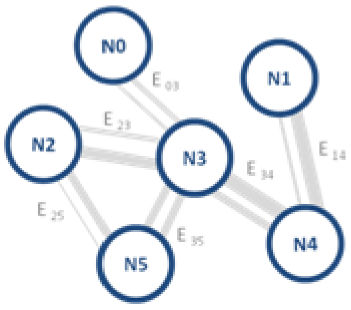

Fig.1: Scheme for the configuration of network nodes (in blue).

Within the file we find the section `<fns_config>` . At this level the user can define the global parameters, i.e., those parameters referred to all network nodes. Such section consists of the following fields:

Field name	**Argument**	**Meaning**
<stop>	Number (decimal)	Duration of the simulation in ms
<glob_rewiring_P>	Number (decimal): [*0,1*]	*small world* rewiring probability
<glob_mu_w>	Number (decimal)	Intra-node weights
<glob_k>	Number (even, integer)	Small World mean degree
<glob_n>	Number (integer)	Number of neurons
<glob_R>	Number (decimal): [*0,1*]	Ratio between excitatory neurons and total number of neurons
<glob_external_inputs_number>	Number (integer)	Number of external inputs

<glob_external_inputs_type>[3]	Number (integer): {*0;1;2*}	Type of external inputs (0=Poisson distribution; 1=constant spike train; 2=noise)
<glob_external_inputs_timestep>	Number (integer)	Firing interval of external inputs (only for Poissonian distributed input) (ms)
<glob_external_inputs_firerate>	Number (decimal): [*0,1*]	Firing probability (only for Poissonian distributed input)
<glob_external_inputs_fireduration>	Number (integer)	External input total duration (ms)
<external_inputs_amplitude>	Number (integer)	External input amplitude
<avg_neuronal_signal_speed>	Number (decimal)	Axonal conduction speed (m/s)
<glob_plasticity>	*true*, *false*	Plasticity on/off

Note that the program rectifies numbers that are not inserted in the required format (removal of fractional digits where ‘integer’ numbers are required, and to the previous even where ‘even integer’ numbers are required).

The `<fns_config>` section also includes two subsections:

- the `<global_neuron_manager>` subsection, where the user can specify the LIFL neuron parameters of the generic network node:

Field name	**Argument**	**Meaning**
<D>	Number (decimal): [*0,1*]	Decay constant (1/ms)
<c>	Number (decimal)	Threshold constant
<t_arp>	Number (decimal)	Absolute refractory period (ms)

- the `<node>` subsection, where the user can re-define parameters for single nodes. Note that each one of such nodes must be redefined individually, and starting from region 0 proceeding consecutively (i.e., it is not possible to define node "*x*" without specify previous *0* to *x-1* nodes). Such section consists of the following fields:

[3] In FNS external inputs start at 0.1 ms

Field name	**Argument**	**Meaning**
<id>	Number (integer)	Node number (starting from *0*)
<rewiring_P>	Number (decimal): [*0,1*]	*small world* rewiring probability
<mu_w>	Number (decimal)	Intra-node weights
<k>	Number (even, integer)	Small World mean degree
<n>	Number (integer)	Number of neurons of each node
<R>	Number (decimal): [*0,1*]	Ratio between excitatory neurons and total number of neurons
<external_inputs_number>	Number (integer)	Number of external inputs
<external_inputs_type>	Number (integer): {*0;1;2*}	Type of external inputs (0=Poisson distribution; 1=constant spike train; 2=noise)
<external_inputs_timestep>	Number (integer)	Firing interval of external inputs (only for Poissonian distributed input) (ms)
<external_inputs_firerate>	Number (decimal): [*0,1*]	Firing probability (only for Poissonian distributed input)
<external_inputs_fireduration>	Number (integer)	External input total duration (ms)
<external_inputs_amplitude>	Number (integer)	External input amplitude
<glob_plasticity>	*true*, *false*	Plasticity on/off

In addition to the abovementioned fields, the user can redefine the neuron parameters of specific nodes through the subsection <node_manager>, to be included within the section <node>.
Note that such nodes are reconfigurable even partially, i.e. only a part of the parameters of the specific node can be specified (FNS will auto-complete with the global values).

3) **interconnectivity_set**. It consists in a folder containing the following files, in order to describe the edge structural data (Fig.2):

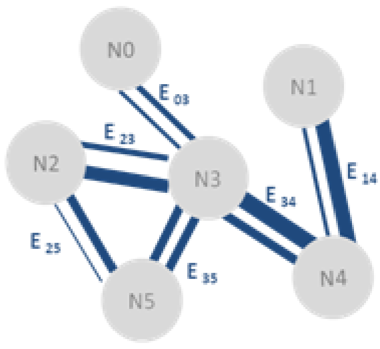

Fig.2: Scheme for the configuration of network edges (in blue). For simplicity in the figure the term E $_{ab}$ indicates both the edge from *a* to *b* and that from *b* to *a*.

File name	**Content**
Ne_en_ratio.txt	Connections to be created from excitatory neurons of node *a* to excitatory neurons of node *b*, divided by the number of neurons of node *a*[4]
mu_omega.txt	Mean of the inter-node synaptic weights
sigma_omega.txt	Standard deviation of the inter-node synaptic weights
mu_lambda.txt	Mean parameter of the inter-node lengths.
alpha_lambda.txt	Shape parameter of the inter-node lengths

All the files consist in adjacency matrices, which values specify parameters of the edge directed from raw index to column index.
Note that the files *sigma_omega*.txt and *alpha_lambda*.txt are optional; if not present, FNS will consider homogeneous inter-node lengths and weights, with the values present in *mu_omega*.txt and *mu_lambda*.txt, respectively.
FNS supports asymmetrical matrices, in order to be able to model different edge values for the two directions. Values pertaining to the main diagonals of all the matrices are not taken in account.
Note that this version of the software does not support instantaneous (zero-length) inter-node connections.

4) **simulation_name**. Such field allows the user to specify the name of the simulation. Simulation data will be placed in the *experiments* folder and characterized by such simulation name.
Two .CSV files will be created, containing information about the activity from and to the selected NOIs, as described in par.2.

[4] Edge cardinality is expressed as *Ne/n$_e$* ratio in order to simplify the network scalability in a parametric simulation approach. For example: a value of 0.8 applied to an edge which transmitter node has 100 excitatory neurons will result in 80 (excitatory) connections directed to the receiver node.

5) **mask**. Such field allows the user to specify nodes of interest (NOIs) which data will be reported in the CSV files (see Par.2). Such list of nodes is coded exploiting a binary mechanism. For example, in order to obtain the mask for the nodes 0,2,3:
2^**0**+2^**2+** 2^**3** => *mask = 13*.
Alternatively, if we are interested to the events pertaining to all nodes, the mask to be considered is **2^N (**where N is the total number of nodes**)**

The mask code can result in a very large number. In such cases we recommend to use the Matlab function *vpa()* to compute the mask.

Note: This version of FNS is based on the basic LIFL configuration. The user can refer to the reference paper for the behaviour of such neuron model.

4. Simulation examples

Three configuration presets are present in the "FNS-simulation_examples" folder:

- *resonance pair*, please refer to:
    - Maslennikov and Nekorkin, 2014. *Modular networks with delayed coupling: Synchronization and frequency control.* Phys. Rev. E 90.
    - Gollo, L. et al., 2014. *Mechanisms of zero-lag synchronization in cortical motifs*. PLOS computational biology 10 (4), 1–17)
- *dynamical relaying*, please refer to
    - Vicente et al., 2008. *Dynamical relaying can yield zero time lag neuronal synchony despite long conduction delays*. Proceedings of the National Academy of Sciences USA 105 (44)
- connectome 14 (representing 14 nodes of the default mode network), please refer to
    - Susi G, Garces P, Cristini A, Paracone E, Salerno M, Maestu F, Pereda E. *FNS: an event-driven spiking neural network framework for efficient simulations of large-scale brain models*. PDF available at arXiv [1801.00864]