

# SpiNet Overview

Paulo de Castro Aguiar  
Instituto de Biologia Molecular e Celular  
Universidade do Porto, Portugal

November, 2007

## 1 Introduction

SpiNet is a simulation environment for large networks of spiking neurons with highly heterogeneous synapses. It comprises a tool for producing and editing network models and a simulation engine. The simulation engine is written in C and uses a second order Runge-Kutta method with a linear interpolant to find spike times and recalibrate post-spike potentials. Neurons are modelled, by default, as integrate-and-fire units with dual exponential synaptic conductances, but different models can be introduced. The engine is capable of handling a vast number of properties including dynamical synapses, long-term plasticity, stochastic activity, detailed 3-dimensional architecture, external stimuli, among others. Communication with the engine is made through a simple structured file, loaded at the beginning of the simulation. These network model files containing all network properties are produced using a versatile and user-friendly MATLAB program called NetBuilder. Complex networks can be easily built, or modified, using the graphical user interface of this tool. An OpenGL graphical output is integrated into the simulation environment providing visual information of the model dynamics.

This document serves as a reference manual for both NetBuilder and the simulation engine.

## 2 NetBuilder interface

The analysis of a model starts with a file containing all the information regarding the architecture and the dynamics of the model. SpiNet uses a simple structure to hold all this information and many different tools can be used to write these *.net network model files*, as long as the structure is preserved. However, to facilitate the construction of network models, SpiNet comes already with a tool, NetBuilder, dedicated to this purpose. NetBuilder is a MATLAB program with a graphical user interface capable of producing general network model files. It is extremely easy to use and very versatile, making the process of building or editing large/complex networks very simple. Notice that you do not need to write a single line of code to build or edit network models. It should however be kept in mind that NetBuilder does not take advantage of all SpiNet features. We will come back to this point later.

To call NetBuilder first open MATLAB and change your working directory to `/SpiNet/net-builders`. As an advisable alternative, put SpiNet and all its subdirectories in MATLAB's search path: File → Set Path... → Add Path...). Now, in the command prompt run the command *NetBuilder*. This opens a GUI window where you can select between loading a previously stored model or starting a new network model from scratch. If you choose to start a model from scratch you will have to specify the total number of neuron populations in your model. The load option is quite important as some models can have many parameters specified. Especially in the analysis stage where the contributions of specific parameters to the overall dynamics are being assessed, it is quite important to be able to call NetBuilder and simply edit the required parameter to produce a new network model file to be simulated and analysed.

Once you have entered your choice, the properties button from the Populations frame becomes accessible. All the fields will be filled with previously stored data, case you chose to load a network model file, or will be filled with default data, case you chose to start a new network model from scratch. The population's properties table contains the following fields:

**label** - a designation for the population; the label must have 5 or less characters;

**fdomains** - number of functional domains (compartments) in this population's neurons; all neurons must have at least 1 functional domain, the soma;

**size** - number of neurons in the population;

**V<sub>rest</sub>** - resting potential of the neurons in this population [mV];

**V<sub>thresh</sub>** - threshold potential of the neurons in this population [mV];

**refrac<sub>period</sub>** - absolute refractory period of the neurons in this population [ms];

The fdomains parameter allows you to add dendritic compartments to the soma. Dendritic functional domains serve the purpose of segregating the inputs from different populations and producing local signal integrations. All functional domains are radially attached to the soma

As an example, in a particular model you may have inhibitory interneurons inputs targeted at the soma (fdomain 0), excitatory interneurons inputs targeted at basal dendrites (fdomain 1) and external excitatory inputs at apical dendrites (fdomain 2). In this case, these neural population would be modelled with fdomains = 3. Neurons modelled with one single fdomain (value "1" in fdomains entry) correspond to the conventional integrate-and-fire neurons.

The absolute refractory period sets the time window after a spike during which the neuron discards all incoming synaptic signals. The membrane voltage value at the end of the refractory is, by default, V<sub>rest</sub>. However, this value can be different if the user defines a spike wave to the played-back every time a neuron fires (see section 3).

After accepting all population properties entries by pressing "OK", all other network model property frames will become accessible. We now discuss one by one in detail.

## 2.1 Connectivity

Connectivity information is entered by pressing the *conn* button. This gives access to a table where each entry represents a specific fibres pathway connecting a particular pre-synaptic population (line) to a particular post-synaptic population (column). Data can be entered in this table under two different connectivity assumptions: divergent connectivity or convergent connectivity. With divergent connectivity each entry defines the number of post-synaptic neurons receiving inputs from a single pre-synaptic neuron. On the other hand, with convergent connectivity each entry represents the number of pre-synaptic neurons providing inputs to each post-synaptic neuron. The decision on how the connectivity table is to be interpreted by NetBuilder is reserved to the moment the network model file is written: at this last stage, NetBuilder asks the user on how to use the table. Notice however that, while NetBuilder does not need to know in advance the nature of the table, you do. Parameters such as synaptic peak conductances (efficacy) have a completely different impact depending on which connectivity type is to be used; so, they have to be defined with a particular connectivity type already in mind. As a final comment to the connectivity table, notice that some constraints must be met by the data inserted. Notably, in a convergent (divergent) connectivity scenario, each entry must be smaller than the pre-synaptic (post-synaptic) population size as the default sources selection mechanism is based on a random sample **without** repetitions.

Through the same table structure, the *fdms* button allows you to specify the target functional domains in the post-synaptic populations. The functional domains are identified by number id's.

If you have set for a particular population `fdomains=3` then you have available the `fdomain 0` (soma), `fdomain 1` and `fdomain 2`. The target functional domains are chosen from the post-synaptic population, not from the pre-synaptic population. By default, all target functional domain number ids is 0 (the soma).

**With very few exceptions, NetBuilder input tables follow the same pre-synaptic/post-synaptic population data structure where columns specify the post-synaptic and rows specify the pre-synaptic.** This consistency across parameters serves the purpose of facilitating the process of data insertion.

## 2.2 Transmission Delay

In this frame you can set the transmission delays for all the connection pathways defined previously. The transmission delay includes axon and synaptic delay. It is therefore the total time between threshold crossing in the pre-synaptic neuron and post-synaptic conductance change at the post-synaptic neuron. You can set a *mean* and standard deviation, *stdev*, assuming a normal distribution. All synapses in this pathway will draw their parameters from this distribution. For identical transmission delays among all synapses in a pathway simply set the standard deviation to “0”.

## 2.3 Geometry

This frame allows you to set a simple 3D organization for each population. All neurons in a population can be placed in a box array with *size\_X*, *size\_Y* and *size\_Z* neurons in each orthogonal direction. Please note that the product *size\_X**size\_Y**size\_Z* must be equal to your total population size. The box array is filled with neurons starting in position (*X0*, *Y0*, *Z0*) and moving toward the axis positive direction. The distance between neurons in every direction follows a normal distribution with parameters *d\_mean* and *d\_std* space units. The space units are relative. Take however into account, for visualization purposes, that the diameter of the soma’s representation is set by default to 1 unit.

## 2.4 Functional Domains

This frame lists all the defined populations in your model. By clicking on a population you can set the parameters for all the functional domains present on each neuron belonging to this population. The available fields are: *label*, *Ra*, *Rm* and *tau\_m*. These correspond, respectively, to the label (again, 5 or less characters), axial resistance [Mohm], membrane resistance [Mohm] and (passive) membrane time constant [ms] of the functional domain compartment. By default, you only have 1 functional domain, the soma, on each defined population. **For the soma (fdm=0) always leave *Ra*=0.** The parameters for any additional functional domains are also set here. The parameter *Ra* for compartments other than the soma must be, at least, 1 Mohm. Please go through all populations to set appropriate values for all the entries. Do not use functional domains unless you are sure that your model needs them and you are confident that you can appropriately set their parameters and interpret the simulation results.

## 2.5 Syn. Conductances

The synaptic peak conductances [nS] for all defined pathways are set in this frame. Recall that SpiNet uses by default a dual-exponential curve to represent post-synaptic conductance profiles. The peak of this curve is the synaptic peak conductance. Again you have the option to define a standard deviation which will make the synaptic efficacies heterogeneous.

## 2.6 2Exp Synaptic Response

This frame can be used to set the rise time constant [ms] and decay time constant[ms] of the dual-exponential conductance profiles for each synaptic pathway.

## 2.7 Dynamical Synapses (STP)

Synapses can exhibit short-term plasticity (STP) through a combination of facilitation and depression dynamics. SpiNet uses [?] (equations 2 and 3) to model the short-term dynamics. Three parameters are available: recovery time constant [ms], facilitation time constant [ms] and release increment [nS]. The short-term dynamics modulate the synaptic peak conductance through a multiplication factor between 0 and 1. A high variability in short-term dynamics within the same pathway is often used in models of temporal to spatial decoding. SpiNet recognizes as dynamical synapses all pathways which have the release increment parameter greater than 0. To activate STP in SpiNet you need to use the option -y in the command line. The existence of this option allows you to run the same model with and without short-term dynamics.

## 2.8 Long-Term Plasticity (LTP)

SpiNet uses a flexible spike timing dependent plasticity (STDP) paradigm to define the long-term plasticity changes. The STDP time is defined as the difference between the post-synaptic spike time and the post-synaptic response initiation time (not the pre-synaptic spike time). The following parameters set the characteristics of the learning window: *STDP t.0* sets the inversion time (the STDP time separating depression from potentiation); potentiation *delta* and *tau* set, respectively, the magnitude of the synaptic change and the length of the time window; depression *delta* and *tau* parameters are equivalent to their potentiation counterparts. Through a flag in SpiNet's command line, the user can choose between rectangular windows or truncated exponentially decaying windows (with length and time constants both set by the respective parameter *tau*). The ability to independently set all the parameters allow the creation of learning windows with distinct properties. For example, an associative learning window with no depression and length of 20 ms can be defined by setting *STDP t.0*=-10, potentiation *tau*=20 and depression *tau*=0. Long-term plasticity dynamics can be applied to either excitatory (exc) or inhibitory synapses (inh). Please note that the *tau* parameters must always be positive, potentiation *delta* must be positive for excitatory synapses (or negative for inhibitory synapses) and depression *delta* must be negative for excitatory synapses (or positive for inhibitory synapses). Synapses peak conductances are bound between 0 and *Smax* (which is positive for excitatory synapses and negative for inhibitory synapses). The depression limit (0) is a hard-bound while the potentiation limit (*Smax*) is a soft-bound ([?]). Please refer to SpiNet original paper for more details on the long-term plasticity dynamics.

In addition to the long-term depression dynamics, SpiNet also has available a homosynaptic depression factor. If set, this parameter is used to update the synaptic peak conductance each time no post-synaptic spike can be associated with the synaptic release. Synapses which systematically fail in firing the neuron are therefore substantially depressed. The homosynaptic depression factor is multiplication factor in the range [0, 1]. For example, if frustration=0.98, a 2.0% peak conductance reduction is produced each time the synaptic release is dissociated from any post-synaptic response. Homosynaptic depression only occurs when neither LTP nor LTD takes place.

## 2.9 Saving Model Data

Once you are happy with all your parameters you can save your data. NetBuilder allows you to store the network model data in two types of files: .mat are MATLAB files which store all

NetBuilder inserted data and .net files which convert all data into a network file which can be loaded and simulated in SpiNet simulation engine. The .mat files allow you to rapidly load and edit previous NetBuilder sessions. SpiNet simulation engine only receives .net files.

### 3 Simulation Engine

A simulation experiment is initiated by calling SpiNet with a set of parameters and options. The most relevant parameter is the network model file where all the details about the model are defined. Calling SpiNet without any parameters lists all the options and parameters and produces a self-explanatory template file for input stimuli data (*stimuli\_template.stim*). The available options and parameters are:

**-t, -total-time=TIME**

Sets the total simulation time in ms (default = 1 time step).

**-d, -dt=STEP**

Sets the simulation time step in ms (default = 0.1 ms).

**-o, -output-netfile=FILE**

At the end of the simulation an output network file is created. This is important if you want to save a network model that has been subject to changes during the simulation (e.g. plasticity). The output network model file contains a complete image of the network state, including all unreached spikes, ongoing post-synaptic potentials and plasticity changes.

**-s, -stimfile=FILE**

Calls a data file with information regarding input stimuli. Many types of stimuli are available and are discussed in detail in section 3.2. When you call *SpiNet* without parameters an input stimuli template file is created in your working directory. This file contains a commented part describing the structure of the file and the types of stimuli available.

**-v, -textinfo**

This option signals the simulation engine to provide some text information regarding the simulation. Information such as time, stimuli activation and long-term plasticity changes are displayed.

**-V, -store-states=PERIOD**

Stores neuron's voltage states in 'out\_states.dat' with PERIOD in ms. This file can then be opened in MATLAB or Octave to display every neuron's voltage profile during the simulation.

**-a, -store-activity-levels**

Stores the activity levels of all neuronal population in 'activity\_levels.dat', at each time step. The activity level definition used here is the fraction of neurons in the populations that fired in the last 1ms.

**-A, -store-spike-times**

Stores all neuron's spike times in 'out\_spike\_times.dat'.

**-g, -graphinfo**

If you have OpenGL in your system, this option opens a graphical window where your network model architecture and dynamics are represented. Depending on your graphics card and on the visualization mode selected, this option may considerably slow down simulation. The graphical window is described in detail in section 3.3.

**-c, -draw-sample-connections**

With the graphical output selected, this option draws sample connections between the defined neuronal population.

**-C, -draw-all-connections**

Same as above but now it draws all network connections.

**-k, -draw-spikes**

With the graphical output selected, this option draws all spikes.

**-u, -myfunctions**

Although the data input/output is versatile, we believe some users may want to extend some of the features of the simulation environment. This option enables the `Manipulate_Data` and `Output_Data` functions which serve as a gateway to the simulation engine. These functions are defined in `myfunctions.c`. This file also contains explanatory information regarding these functions. The use of the option is not advisable unless you really know what you are doing...

**-b, -debug-netfile**

In case you want to use other ways to produce network model files other than NetBuilder, there is a chance of you producing files with inconsistencies. Although the network file structure is simple and straightforward, errors will be hard to detect in models with thousands of neurons. This option activates a debug tool which simply writes all input network data as it was been interpreted by the simulation engine into a new file, 'debug.net', and up to the point where the error occurs. The 'debug.net' file is itself a network file and by analysing its end it is possible to detect the error source.

**-p, -enable-LTPlasticity**

This option enables long-term synaptic plasticity. By default all long-term plasticity dynamics defined in the network model file are disabled. This switch is important for memory models where storage and recall stages are to be analysed using the same network model. Please refer to SpiNet paper for detail about the long-term plasticity dynamics.

**-y, -enable-STPlasticity**

This option enables short-term synaptic plasticity. By default all short-term plasticity dynamics defined in the network model file are disabled. This switch is important for memory models where storage and recall stages are to be analysed using the same network model. Please refer to SpiNet paper for detail about the long-term plasticity dynamics.

**-f, -enable-rate-estimation**

This option enables spike rate estimation for all neurons. This information may be important to compare simulation results with experimental data where the neuron output is recorded as firing-rates. The rate estimation uses an causal kernel, with a time window of 100ms and a pool of up to 50 last spike times. This option considerably slows down the simulation.

**-r, -initial-reset**

This option perform a pseudo-reset to the network model. This 'reset' erases all spikes, post-synaptic signals, short-term memory and puts all membrane voltages at resting potential. This option may be useful if you want to used a stored output network file but with silenced activity.

**-R, -periodic-reset=PERIOD**

Same as above but now it allows you to define periodic resets with PERIOD ms. This may be useful in some learning experiments where a sequence of activity patterns are to be stored. The periodic reset assures that the state in each storing phase is identical.

**-S, -input-spontaneous-rate=RATE**

Forces neurons in the first population (called *Pop00* by default) to fire spontaneously at RATE Hz. The interspike intervals follow independent Poisson distributions with parameter 1/RATE. Through this parameter, the first population can be used as a source of stochastic activity to the network.

### 3.1 Action Potential Wave

By default, at run time, SpiNet searches for a file in the working directory called `ap_wave.dat`. This file contains the spike wave profiles which will be played-back every time a neuron fires

and during its absolute refractory period. This file is supposed to be an ascii file containing the action potential wave profiles for each population present in the model. The data must be organized as a table with the first line containing time values (ms) in steps smaller than the simulation time step  $dt$ , and the remaining lines containing voltage values (mV) for each of the model populations. Each population list of voltages must end in the column with the correct absolute refractory period of that population. The table must start with the “START” keyword and finish with the “END” keyword. Below is a template for an *ap\_wave.dat* file for a model with two population with absolute refractory periods of 3.0 and 5.0 ms:

```
START
0.00 0.01 ... 2.99 3.00 3.01 ... 4.99 5.00
-50.0 -49.7 ... -70.1 -70.0
-50.0 -49.7 ... -75.2 -75.0 -74.8 ... -70.1 -70.0
END
```

Such a *ap\_wave.dat* file could be used for simulations with a time step not smaller than 0.01 ms. If SpiNet does not find a *ap\_wave.dat* file in the working directory it produces artificial wave profiles for each population: square wave with 1 ms width at 30 mV, with the base line at the population’s resting potential and lasting for the entire refractory period.

### 3.2 Stimuli file

Different stimuli can be applied during a stimulation experiment. Stimuli data is stored in ascii .stim files which can be loaded by SpiNet simulation engine with the `-s filename.stim` option. The file template and possible stimuli are show below:

```
# This is a template for an input stimuli file.
# The commands only start after the 'START' keyword.
# Times have to be sorted. The data structure is:
#
# START
# <stim0 start time [ms]><n. electrodes>[<first neuron id>,<last neuron id>]:<command> <argu
# <stim1 start time [ms]><n. electrodes>[<first neuron id>,<last neuron id>]:<command> <argu
# ...
#
# Possible commands, and respective arguments, are:
# fire neurons in range : F
# fire neurons periodically : T <period [ms]> <stim duration [ms]>
# fires a selected number of random neurons : R <sample size [1]>
# fire neurons independently with Poisson ISI : P <mean value [ms]> <stim duration [ms]>
# fire neurons synchronously with Poisson ISI : S <mean value [ms]> <stim duration [ms]>
# inject constant current : I <current [nA]> <stim duration [ms]>
# inject random current (Gaussian) : N <current mean [nA]> <current stdev [nA]> <
# Example:
#
# START
#
# 100.0 1 [100,200]:N 500.0 1.5 0.2
# 150.0 3 [0,100]:F [150,200]:I 100.0 2.0 [300,400]:S 20.0 100.0
# 250.0 2 [1,1]:T 10.0 50.0 [210,210]:P 50.0 200.0
# 300.0 1 [400,500]:R 20
```

START

0 0

### 3.3 Graphical environment

If you set the graphical output flag **-g** in SpiNet command line a graphical window will open with a representation of your model. Left-clicking the mouse in the graphical window allows you to chose from 3 visualization modes:

- schematic: 2D visualization of all units in all populations;
- dot: light 3D visualization with neurons represented as dots;
- detailed: heavy 3D rendered visualization with detailed neuron representations;

In the 3D visualization modes you can navigate in the model architecture using the **cursor** to pan the image and **PageUp/PageDown** to zoom in and out. Holding down the **Ctrl** key while pressing the above keys increases the step of the translations. Rotations can be produced holdind down the left button and moving the **mouse**. You can pause the simulation with the **P** key, move time step by time step pressing **S** and quit pressing **Q** or **Esc**. Some of the command line graphical options are also accessible within the graphical window: **k** switches on and off spikes representations, **c** and **C** controls connections representations. Finally **f** switches on and off frequency estimation. Neurons membrane voltages are color coded: blue for below  $V_{rest}$ , red between  $V_{rest}$  and  $V_{thresh}$  and yellow for membrane voltages above  $V_{thresh}$ ; neurons are white when at rest.

### 3.4 Export Data and Analysis Tools

The results of simulations experiments can be exported using the specific command line options. For each of the options there is a basic MATLAB program to open and visualize the exported data. These MATLAB programs are placed in the *SpiNet/Analysis* directory. You are advised to put *SpiNet/* and all its subdirectories in MATLAB's search path: File → Set Path... → Add Path...).

#### **-V, -store-states=PERIOD**

Stores neuron's voltage states in *out\_states.dat* with PERIOD in ms. Note that this option can produce very large files if your neuron populations are large and/or the simulation time is large. To open and visualize the data change your MATLAB working directory to the directory you have the file *out\_states.dat* and call *SpiNetPlotV.m* to display every neuron's voltage profile during the simulation.

#### **-a, -store-activity-levels**

Stores the activity levels of all neuronal population in *activity\_levels.dat*, at each time step. The activity level definition used here is the fraction of neurons in the populations that fired in the last 1ms. Again, to open and visualize the data change your MATLAB working directory to the directory you have the file *activity\_levels.dat* and call *SpiNetPlotActivity.m* to display every population activity profile during the simulation.

#### **-A, -store-spike-times**

Stores all neuron's spike times in *out\_spike\_times.dat*. Use *SpiNetPlotRaster.m* to display the data.

#### **-o, -output-netfile=FILE**

At the end of the simulation an output network file is created. This is important if you want to save a network model that has been subject to changes during the simulation (e.g. plasticity). The output network model file contains a complete image of the network state, including all

unreached spikes, ongoing post-synaptic potentials and plasticity changes. The output file can be loaded by SpiNet just as any other network model file.