My Project

Generated by Doxygen 1.8.9.1

Sat Sep 17 2016 12:56:12

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Network Class Reference

`#include <network.hpp>`

Public Member Functions

- **Network** (double dt)
- void add_neuron (double refractory_duration, double noise)
- void connect_neurons (unsigned int pre_syn_nb, unsigned int post_syn_nb, double weight, bool type, double ww, double gmax, double delay_duration, int type_of_learning)
- void step (vector< double > &frequency_first_firing, unsigned int nb_purkinje, unsigned int nb_nuclear, unsigned int nb_climbing)
- void resetI ()
- vector< unsigned int > create_pop (unsigned int nb_neurons, double ref, double noise)
- void connect_pop (const vector< unsigned int > &V_pre_syn, const vector< unsigned int > &V_post_syn, double weight, bool type, double ww, double gmax, double delay_duration, unsigned int type_of_learning, unsigned int nb_pre_max_for_one_post)
- void hebbian_learning (vector< unsigned int > a, vector< unsigned int > b, int type_of_learning)
- double get_t ()
- Neuron get_Neuron (unsigned int nb)
- void set_I_neuron (unsigned int neuron_nb, double i)
- void set_ff_neuron (unsigned int neuron_nb, bool state)
- vector< double > get_time_L ()
- vector< double > get_recVM_neuron (unsigned int neuron_nb)

Protected Attributes

- double dt
- double t
- vector< double > time_L
- vector< Neuron > neuron_L
- vector< Synapse > synapse_L
- vector< double > **first_firing_L**
- map< unsigned int, vector< unsigned int > > connection_pre
- map< unsigned int, vector< unsigned int > > connection_post

2.1.1 Detailed Description

Network accounts for the neural network

2.1.2 Member Function Documentation

2.1.2.1 void Network::add_neuron (double *refractory_duration*, double *noise*)

add a Neuron to the Network

2.1.2.2 void Network::connect_neurons (unsigned int *pre_syn_nb*, unsigned int *post_syn_nb*, double *weight*, bool *type*, double *ww*, double *gmax*, double *delay_duration*, int *type_of_learning*)

create a Synapse between two Neuron

2.1.2.3 void Network::connect_pop (const vector< unsigned int > & *V_pre_syn*, const vector< unsigned int > & *V_post_syn*, double *weight*, bool *type*, double *ww*, double *gmax*, double *delay_duration*, unsigned int *type_of_learning*, unsigned int *nb_pre_max_for_one_post*)

connect two populations of Neuron

2.1.2.4 vector< unsigned int > Network::create_pop (unsigned int *nb_neurons*, double *ref*, double *noise*)

create a population of Neurons

2.1.2.5 Neuron Network::get_Neuron (unsigned int *nb*) [inline]

Get the neuron number nb

2.1.2.6 vector< double > Network::get_recVM_neuron (unsigned int *neuron_nb*)

Get the voltage values vector

2.1.2.7 double Network::get_t ( ) [inline]

Get the current time

2.1.2.8 vector< double > Network::get_time_L ( )

Get the vector of time

2.1.2.9 void Network::hebbian_learning (vector< unsigned int > *a*, vector< unsigned int > *b*, int *type_of_learning*)

create a site of plasticity between two population of neurons

2.1.2.10 void Network::resetI ( )

reset the current value of every neurons to 0

2.1.2.11 void Network::set_ff_neuron (unsigned int *neuron_nb,* bool *state*)

Set the first firing to state of neuron #nb

2.1.2.12 void Network::set_I_neuron (unsigned int *neuron_nb,* double *i*)

Set the current value to i of neuron #nb

2.1.2.13 void Network::step (vector< double > & *frequency_first_firing,* unsigned int *nb_purkinje,* unsigned int *nb_nuclear,* unsigned int *nb_climbing*)

update the network ; that is, update all the Synapse and Neuron

2.1.3 Member Data Documentation

2.1.3.1 map<unsigned int, vector< unsigned int > > Network::connection_post [protected]

contain post synaptic neuron IDs and associates their synapses IDs

2.1.3.2 map<unsigned int, vector< unsigned int > > Network::connection_pre [protected]

contain pre synaptic neuron IDs and associates their synapses IDs

2.1.3.3 double Network::dt [protected]

time step (s)

2.1.3.4 vector<Neuron> Network::neuron_L [protected]

neurons list of the network

2.1.3.5 vector<Synapse> Network::synapse_L [protected]

synapses list of the network

2.1.3.6 double Network::t [protected]

time t (s)

2.1.3.7 vector<double> Network::time_L [protected]

time list (s)

The documentation for this class was generated from the following files:

- network.hpp
- network.cpp

2.2 Neuron Class Reference

`#include <network.hpp>`

Public Member Functions

- **Neuron** (double ref, double _noise)
- const double f_v ()
- bool step (double dt)
- void set_I (double i)
- double get_I ()
- void set_first_firing (bool state)
- bool get_first_firing ()
- vector< double > get_rec_Vm ()
- unsigned int get_nb ()
- double get_v ()
- double get_refractory_delay ()
- void set_refractory_delay (double rd)
- double get_refractory_duration ()
- double get_V_max ()

Static Public Member Functions

- static unsigned int get_NB_NEURONS ()
- static void set_NB_NEURONS (unsigned int nb)

Protected Attributes

- unsigned int nb
- double R
- double C
- double V_max
- double V_hyp
- double V_rest
- double V_tresh
- double I
- double v
- double noise
- bool first_firing
- double refractory_delay
- double refractory_duration
- vector< double > rec_Vm

Static Protected Attributes

- static unsigned int NB_NEURONS = 0

2.2.1 Detailed Description

Neuron accounts for a unit of the network

2.2.2 Member Function Documentation

2.2.2.1 const double Neuron::f_v ( )

return value of voltage

2.2.2.2 bool Neuron::get_first_firing ( )

get the value of first firing variable

2.2.2.3 double Neuron::get_I ( )

Get the current value (A) of the neuron

2.2.2.4 unsigned int Neuron::get_nb ( )

Get the number of the neuron

2.2.2.5 unsigned int Neuron::get_NB_NEURONS ( ) [static]

Get the number of neurons created

2.2.2.6 vector< double > Neuron::get_rec_Vm ( )

Get the vector containing voltage values

2.2.2.7 double Neuron::get_refractory_delay ( )

Get the refractory delay of the neuron

2.2.2.8 double Neuron::get_refractory_duration ( )

Get the refractory duration of the neuron

2.2.2.9 double Neuron::get_v ( )

Get the current voltage value (V) of the neuron

2.2.2.10 double Neuron::get_V_max ( )

Get the maximal value of the voltage (V)

2.2.2.11 void Neuron::set_first_firing (bool *state*)

Set the first firing to state

2.2.2.12 void Neuron::set_I (double *i*)

Set the current(A) of the neuron

2.2.2.13 void Neuron::set_NB_NEURONS (unsigned int *nb*) [static]

Set the number of neuron created

2.2.2.14 void Neuron::set_refractory_delay (double *rd*)

Set the refractory delay of the neuron

2.2.2.15 bool Neuron::step (double *dt*)

update membrane potential value of the Neuron

2.2.3 Member Data Documentation

2.2.3.1 double Neuron::C [protected]

cell capacitance (F.m-2)

2.2.3.2 bool Neuron::first_firing [protected]

indicates if the neuron has fired already or not

2.2.3.3 double Neuron::I [protected]

current value (A) at time t

2.2.3.4 unsigned int Neuron::nb [protected]

Neuron ID

2.2.3.5 unsigned int Neuron::NB_NEURONS = 0 [static], [protected]

number of neurons in the network

2.2.3.6 double Neuron::noise [protected]

noise value applied to current I

2.2.3.7 double Neuron::R [protected]

cell resistance (ohm.m2)

2.2.3.8 vector<double> Neuron::rec_Vm [protected]

recordings list of membrane potentials

2.2.3.9 double Neuron::refractory_delay [protected]

refractory delay in seconds (remaining time for a neuron to be ready to fire again after an action potential)

2.2.3.10 double Neuron::refractory_duration [protected]

refractory duration in seconds (fixed duration for a neuron to be ready to fire again after an action potential)

2.2.3.11 double Neuron::v [protected]

membrane potential value (V) at time t

2.2.3.12 double Neuron::V_hyp [protected]

tresholding potential (V)

2.2.3.13 double Neuron::V_max [protected]

resting potential (V)

2.2.3.14 double Neuron::V_rest [protected]

voltage value (V) reached by an action potential (at the peak)

2.2.3.15 double Neuron::V_tresh [protected]

voltage value (V) during hyperpolarization (after action potential)

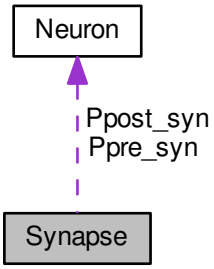
The documentation for this class was generated from the following files:

- network.hpp
- network.cpp

2.3 Synapse Class Reference

`#include <network.hpp>`

Collaboration diagram for Synapse:



Public Member Functions

- **Synapse** (double _weight, Neuron *_pre_syn, Neuron *_post_syn, bool _type, double _ww, double _gMax, double _delay_duration, int type_of_learning)
- void trigger_AP ()
- void activate_synapse ()
- const bool check_triggering ()
- void step (double dt, double t)
- double get_PSP ()
- void update_weight (double t)
- double calculate_delta_w (double delta_timing)
- unsigned int get_nb ()
- void set_I_Ppost_syn (double i)
- double get_I_Ppost_syn ()
- void set_type_of_learning (int type_of_learning)

Static Public Member Functions

- static void set_NB_SYN (unsigned int nb)

Protected Attributes

- unsigned int nb
- int type
- double weight
- bool state
- Neuron * Ppre_syn
- Neuron * Ppost_syn
- double delay
- double delay_duration
- double ww
- double g
- double gMax
- int type_of_learning
- double t_last_AP

Static Protected Attributes

- static unsigned int NB_SYN = 0

2.3.1 Detailed Description

Synapse accounts for the link between two neurons (axon + synapse)

2.3.2 Member Function Documentation

2.3.2.1 void Synapse::activate_synapse ( )

activate a synapse by increasing synaptic conductance

2.3.2.2 double Synapse::calculate_delta_w (double *delta_timing*)

return the weight delta value

2.3.2.3 const bool Synapse::check_triggering ( )

activate a synapse if delay == 0 (that is, if the AP reached the synapse)

2.3.2.4 double Synapse::get_I_Ppost_syn ( )

Get the current value of post synaptic neuron

2.3.2.5 unsigned int Synapse::get_nb ( )

Get the number of the synapse

2.3.2.6 double Synapse::get_PSP ( )

returns value of PSP (post-synaptic potential)

2.3.2.7 void Synapse::set_I_Ppost_syn (double *i*)

Set the current value of post synaptic neuron

2.3.2.8 void Synapse::set_NB_SYN (unsigned int *nb*) [static]

Set the number of synapses created

2.3.2.9 void Synapse::set_type_of_learning (int *type_of_learning*)

Set the type of learning of the synapse

2.3.2.10 void Synapse::step (double *dt*, double *t*)

adjust synaptic conductance value and check if synapse should activate

2.3.2.11 void Synapse::trigger_AP ( )

launch action potential in axon (set up delay)

2.3.2.12 void Synapse::update_weight (double *t*)

adjust the weight of the synapse depending on its plasticity

2.3.3 Member Data Documentation

2.3.3.1 double Synapse::delay [protected]

determine remaining time for an AP to reach the synapse

2.3.3.2 double Synapse::delay_duration [protected]

determine a fixed duration for an AP to reach the synapse

2.3.3.3 double Synapse::g [protected]

current synaptic conductance (S)

2.3.3.4 double Synapse::gMax [protected]

Maximal synaptic conductance (S)

2.3.3.5 unsigned int Synapse::nb [protected]

synapse ID

2.3.3.6 unsigned int Synapse::NB_SYN = 0 [static], [protected]

accounts for the number of synapses in the network

2.3.3.7 Neuron* Synapse::Ppost_syn [protected]

pointer to post-synpatic Neuron

2.3.3.8 Neuron* Synapse::Ppre_syn [protected]

pointer to pre-synaptic Neuron

2.3.3.9 bool Synapse::state [protected]

true if an action potential is running through the axon. Else, false

2.3.3.10 double Synapse::t_last_AP [protected]

is the last time at which the synapse was activated

2.3.3.11 int Synapse::type [protected]

a synapse can be excitatory (+1 or true) or inhibitory (-1 or false)

2.3.3.12 int Synapse::type_of_learning [protected]

accounts for plasticity of the synapse, that is weight adjustements. If type_of_learning = 0, no plasticity. If type_↩ of_learning = 1, hebbian learning. If type_of_learning = -1, anti-hebbian learning

2.3.3.13 double Synapse::weight [protected]

synaptic weight

2.3.3.14 double Synapse::ww [protected]

adjust to scale the value of weight

The documentation for this class was generated from the following files:

- network.hpp
- network.cpp

Index