EVE-symnet

0.0.1

Generated by Doxygen 1.8.6

Thu Oct 15 2015 13:57:00

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 LifNetSim Class Reference

Class implementing the network event based simulator.

`#include <lifnetsim.h>`

Public Member Functions

- void set_init_cond (std::vector< double > init_volts, std::vector< double > pulses, std::vector< double > init_deltas)
- void run (std::string limit_run, long unsigned limit)

  Runs the simulation given a trial_time or n_resets (not both, error is given).
- void nearest_reset_t (std::vector< int > &, double &)

  Finds the nearest reset time from the oscillators.
- void receive_net_pulses (bool *who_received)

  Adds incoming pulses from network neurons.
- void update_net_pulses_timings ()

  Updates timings of pulses from network.
- double generate_ext_pulse_t ()

  Randomly generates new noise pulses.
- void receive_ext_pulses ()

  Adds incoming pulses from the outside.
- void update_ext_pulses_timings ()

  Updates timings of pulses outside network (noise)
- void save_saddle_t_and_id ()
- void save_reset_timing (int osc)
- void save_data (std::string file_name)

  Save simulation data to file.
- void check_saddle ()

  Updates saddle DFA and checks if saddle has been reached.
- void set_noise (double noise_rate, double noise_amplitude_squared)
- void check_resets (bool *who_reset)

  1 -> 0 and send pulses from resetting oscillators,
- LifNetSim (std::vector< double > volts, std::vector< double > pulses, std::vector< double > deltas, double noise_rate=0, double ext_psp=0)

  Create new simulator.

Public Attributes

- data_transf::Data_dump data_to_save

  Google protobuf data container.
- std::vector
  < data_transf::Oscillator * > osc_to_save

  Each Oscillator variable contains reset timings for one oscillator.
- unsigned int n_osc

  Number of oscillators in network.
- std::vector< double > osc_volts

  Voltages of oscillators.
- std::vector< std::deque< double > > net_pulses

  Network pulses timings for each oscillaor.
- std::vector< double > ext_pulses

  Pulses from the outside to be applied to each oscillator (noise)
- std::vector< double > deltas

  Delta current for each oscillator.
- const double I = 1.04

  Input current.
- const double gamma = 1

  Dissipative factor.
- const double tau = 1

  Membrane constant.
- const double net_psp = 0.025

  Post-synaptic potential of pulse.
- const double pdelay = 0.49

  Phase delay of pulses.
- const double tdelay = pdelay * tif(gamma, I, tau)

  Time delay of pulses.
- double ext_psp = 0

  Post-synaptic potential of external pulses (noise)
- double noise_rate = 0

  Rate of noise.
- bool * **who_received**
- bool * **who_reset**
- char * **saddle**
- int s_DFA [5][3]
- int s_DFA_curr_state = 0

  current state of DFA
- int s_DFA_accept = 4

  accept state of DFA
- std::vector< std::vector
  < double > > net_conn_mat

  Network connection matrix.
- double t_to_next_event = double(INFINITY)

  Time to next event.
- unsigned long time_units = 0

  Time units elapsed since run start.
- double time_subunits = 0

  Time subunits since last time unit.
- double time = 0

Simulation time since run start.

- bool stop_run = false

  When true, simulation loop breaks.

- unsigned long reset_counter = 0
- unsigned long saddle_counter = 0

  A counter for the number of saddle reached since run start.

Friends

- class **boost::serialization::access**

2.1.1 Detailed Description

Class implementing the network event based simulator.

more stuff

2.1.2 Member Function Documentation

2.1.2.1 void LifNetSim::run (std::string *limit_run,* long unsigned *limit*)

Runs the simulation given a trial_time or n_resets (not both, error is given).

Parameters

trial_time	the total time of the simulation
n_resets	the total number of resets for the reference neuron until the simulation is stopped.

2.1.2.2 void LifNetSim::save_reset_timing (int *osc*)

Save timing of oscillator osc, for later dumping to file via function save_data

2.1.2.3 void LifNetSim::save_saddle_t_and_id ( )

Saves saddle and its timing for later dumping to file via function save_data

2.1.2.4 void LifNetSim::set_init_cond (std::vector< double > *init_volts,* std::vector< double > *pulses,* std::vector< double > *init_deltas*)

Used in initialization and when resetting of initial condition is needed

2.1.2.5 void LifNetSim::set_noise (double *noise_rate,* double *noise_amplitude_squared*)

Used in initialization and when setting noise parameters is needed

2.1.3 Member Data Documentation

2.1.3.1 unsigned long LifNetSim::reset_counter = 0

A counter for the number of resets of reference oscillator since run start

2.1.3.2 int LifNetSim::s_DFA[5][3]

Initial value:

```
=
    {
      {0, 1, 0},
      {0, 0, 2},
      {3, 0, 0},
      {0, 0, 4},
      {0, 0, 0}
    }
```

DFA to check saddle is reached states in rows, symbols in columns symbols is the number of oscillators resetting due to a pulse. If DFA reaches state 4, then a new saddle is reached.

	0	1	2
0	0	1	0
1	0	0	2
2	3	0	0
3	0	0	4
4	0	0	0

The DFA reflects the sequence:

- singleton resets because of a pulse
- stable cluster resets because of pulse
- the new singleton resets naturally
- the new stable cluster resets because of pulse (new saddle reached, accept)

The symbols in the DFA mean: 0: one or more natural resets were observed 1: a single reset because of pulses was observed 2: two resets because of pulses were observed

note that the ambiguity of 0 is not important, as the table of events (see Schittler Neves, Fabio: Universal Computation and Memory by Neural Switching, 2010) assures that at DFA state 2, the only natural reset that can be observed is that of the new singleton. So when the symbol 0 is received in state 2, it can only signify that the new singleton fired.

The DFA transition table is implemented in a bidimensional array.

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

- /home/giovanni/Dropbox/cpp_sims/include/lifnetsim.h

Index