Brief description of the simulation code as used in Durstewitz (2003), Self-organizing neural integrator … J Neurosci 23:5342-5353.

This package contains the following files:

1) LAConfigPar.mat

Contains various line attractor parameter configurations, including those on which the figures in the paper were based (e.g. LAPar1 was used for Figs. 3 & 6). The configurations are ML vectors of length 38 - the meaning of entries 1-31 should be apparent from lines 21-48 of the file LIFNetML.m (see below) + Table 1 of the paper. Rows 36-38 of the vectors contain the average background synaptic conductances for AMPA, NMDA, and GABAA conductances. For simulations where the noisy background synaptic inputs were replaced by their means, these values were directly used in the equations. For simulations with Poisson-like background synaptic inputs, these values were converted to strengths of single synaptic inputs and associated input rates as exemplified on lines 39-53 of the file runSimExample.m, preserving the means. Entries 32-35 were not used for single neuron simulations, and were used for properties of between-neuron synaptic connections in simulations including input or response neurons (as in Fig. 3D; see lines 65-104 of the file CallLIF_C.m).

2) LIFNetML.m

This is the MatLab-version of the simulation tool. It is a special purpose tool for single neuron simulations as described in the paper, and solves eqn. 1-8. Inputs/outputs are explained within the file. This file might be useful for grasping the essentials of the simulation code, but runs much slower than the C-implementation.

3) LIFNetSim1.mexglx/ LIFNetSim1.dll

This is the ML-compiled C simulator for Linux/Windows. It is more general and runs much faster than LIFNetML.m. It requires as inputs:

CtrPar	A vector of length 13 specifying general parameters of the simulation
NeuPar	A 15 x NT matrix (with NT=number of neuron types) where each column specifies the parameters of a particular neuron type
NPList	A column vector of length N (N=number of neurons) which specifies for each neuron the neuron type parameters (from NeuPar) with which it is to be instantiated (with numbers corresponding to the columns of NeuPar)
StypPar	A 15 x ST matrix (with ST=number of synapse types) where each column specifies the parameters of a particular type of synapse
SynPar	A 5 x NumSyn (=total number of synapses) matrix where each column specifies the synapse type (according to STypPar), STD/STF param., and individual weight for each synapse

SPMtx	A N x N x SynPerN matrix which specifies for each connection between the N neurons the synapse(s) to be used (with numbers corresponding to SynPar columns)
EvtMtx	A N x NumEvt (=total number of current 'events') matrix where each column specifies for each neuron the amount of an externally applied current
EvtTimes	A 2 x NumEvt matrix specifying in the 1st row the start time and in the 2nd row the duration of each event
ViewList	A column vector of <=N rows which contains a list of all neurons whose membrane potential, ADP m- gate etc. (dep. on version) should be monitored
BackgrdInp	A N x NumBackgrdSyn matrix specifying for each neuron the synapse numbers (according to SynPar) of constant or noisy synaptic background conductances

For semantics of the most important input parameters see ParSemantics.mat.

Outputs are matrices containing spike times of all neurons (*ST*), matrices containing various simulation variables (like Vm, m- gate of gADP etc.; *Vall*) for neurons specified in ViewList, with their corresponding recording times (*tall*), Ca2+ values (*Cai*), synaptic weights (*WgtMtx*).

Note: If warnings are not suppressed (via CtrPar(13)), the software will output a lot of 'stepsize underflow' (hnext<hmin) warnings. This happens usually when spikes are detected, since the software tries to locate the exact spike time point which often requires to take smaller steps than the minimum step specified - this does not affect simulation results or performance in any way.

4) CallLIF_C.m

This routine converts all simulation parameters as specified for the ML simulator into the format required by the C simulator (see above). So the C simulator can be called the same way as the ML simulator through this routine (see runSimExample.m).

5) runSimExample.m

Shows how to specify parameters for the simulations and how to use the simulators.

6) NullClLIF.m

Computes the nullclines of the LIF system with gADP(Ca2+); inputs as specified in the file. The vector 'mgsyn' usually contains positions 36:38 of LAPar.