Computational Neurodynamics Coursework 1

Team: Ziyi Wang, Yanjinlkham Temuujin, Junle Yu

Overview

This document provides a Python simulation of small-world modular networks using **Izhikevich neuron models**. The `myNetwork` class simulates complex neuron dynamics with excitatory and inhibitory connections, with functions for network setup, connectivity, simulation, and visualisation.

Code Structure

- `__init__(self, num_ex, num_in, num_ex_module, Dmax)`
  - **Purpose**: Initialises the network, setting the number of excitatory and inhibitory neurons and configuring initial parameters.
  - **Parameters**:
    - `num_ex` : Total excitatory neurons.
    - `num_in` : Total inhibitory neurons.
    - `num_ex_module` : Number of excitatory neuron modules.
    - `Dmax` : Maximum delay (ms) for synaptic transmission.
- `set_Ex_to_Ex(self)`
  - **Purpose**: Defines excitatory-to-excitatory connectivity within each module.
  - **Return**: Returns a connectivity matrix, `ex_to_ex` , specifying the excitatory neuron connections.
- `set_Ex_to_In(self)`
  - **Purpose**: Establishes connections from excitatory neurons to inhibitory neurons, with each inhibitory neuron connecting to four excitatory neurons.
  - **Return**: Returns a connectivity matrix, `ex_to_in` , indicating excitatory-to-inhibitory connections.
- `set_weights(self)`
  - **Purpose**: Constructs a synaptic weight matrix for the network by combining connectivity matrices for excitatory and inhibitory neurons.
  - **Dependencies**: Uses `set_Ex_to_Ex()` and `set_Ex_to_In()` for partial matrices, which are combined and scaled to create the full weight matrix.
- `set_parameters(self)`
  - **Purpose**: Sets the neuron model parameters (`a` , `b` , `c` , `d`) for excitatory and inhibitory neurons with slight variation for realism.
- `set_delays(self)`
  - **Purpose**: Configures synaptic delays for all neuron-to-neuron connections, with specific delays assigned based on neuron types.
- `rewire(self, p)`
  - **Purpose**: Rewires excitatory-to-excitatory connections within each module with a probability `p` , implementing small-world network characteristics.
  - **Parameters**: `p` : Probability of rewiring each excitatory connection.
- `plot_connectivity(self)`
  - **Purpose**: Plots the network's connectivity matrix, showing the structure of all connections.
- `set_all(self)`
  - **Purpose**: High-level initialiser that sets up the network by calling `set_weights` , `set_delays` , and `set_parameters` .
- `plot_raster(self, T)`
  - **Purpose**: Runs the simulation for `T` ms and generates a raster plot of neuron firing events over time.
  - **Parameters**: `T` : Simulation time (ms).
- `plot_mean_fire_rate(self, window_size, step_size)`
  - **Purpose**: Computes and plots the mean firing rate of neurons in each module over a 1000 ms simulation using a sliding window approach.
  - **Parameters**:
    - `window_size` : Size of each window (ms).
    - `step_size` : Time shift between windows (ms).

Dependencies

- **Python Libraries**:
  - `numpy` : For numerical operations.
  - `random` : For random sampling and shuffling.
  - `matplotlib` : For plotting connectivity and simulation results.
- **Custom Module**:
  - `iznetwork` : Custom module to manage neuron network simulation, used for creating the `IzNetwork` object and setting neuron parameters, weights, and delays.