1 Parameters

Most of the parameter can be defined within the `settings.xls`-file. A detailed description of the parameters can be found in my master thesis. This file is in the same folder of each code and has multiple sheets:

- **Connection Weights**
  This table defines the connection weights (in mV) between the neurons, where each row defines the type of pre-synaptic cell and each column defines the post-synaptic cell. The values within the table could be positive (excitatory) as well as negative (inhibitory). If the value is zero, then no stimulation occurs.
- **Connection Probabilities**
  This table defines the connection probabilities between the neurons, where each row defines the type of pre-synaptic cell and each column defines the post-synaptic cell. The values within the table has to be in the interval $[0, 1]$. A value of 1 means that each pre-synaptic cell of the corresponding type will be connected to each post-synaptic cell of the corresponding type. If the value is zero, then no connection occurs.
- **Resting Potential**
  This list defines the membrane resting potential for each type in mV. Standard values are -65mV for excitatory and -63mV for inhibitory cells.
- **Neuron Ratios**
  This list defines the number of cells for each cell population.
- **Noise Weights**
  This list defines the weights for noise stimulation. The first line defines the cell populations for which noise should occur. The second line defines the weights.
- **Noise Reversal Potential**
  This list defines the noise reversal potential, for each type of cell population defined within the Noise-Weights-sheet

Further parameters can be defined within the `spikenet.m`-File, the `prosthesis.m`-File, and while calling a function:

- The `MAX_WEIGHTS_SCALE` variable allows to define the maximum weights scale factor that can be reached while rewarding.
- The spiking threshold can be defined while constructing a network: `spikenet(THRESHOLD,...)`.

- The number of EM spikes causing 1° of angle change can be defined with the variable `PEAKS_PER_ONE_DEGREE`. The variable has to be a positive integer value.
- The variable `PLOT_INSTANTLY` defines whether (true) or not (false) the angle-plot should be shown while the simulation is running.

The remaining parameters are specific for the ongoing learning approach and the static learning approach and will be described in the following.

2 Ongoing Learning Approach

The code for the ongoing learning approach is within the folder `ongoing`. The `Main.m`-file shows an example how to run a simulation. Further parameters that can be defined are:

- `num_models`: the number of models that should be generated.
- `run_time`: the simulated runtime for each trial in *ms*.
- `run_settings`: defines the target angles (and the corresponding start angles) that should be simulated.
- `trials_per_settings`: the number of trials that should be simulated for each model and each target angle.
- `model_number`: can be set to any value in order to identify the model. This value is also used within the names of the output-files.

The simulation of a network `net` can be started with the command:

```
net.simulate(model_number,run_time,run_settings,trials_per_settings)
```

The return-value is a variable containing the average RMSDs of all trials separated by the target angles.

3 Static Learning Approach

In order to get the best runtime, the code for the static learning approach is split into two parts, one for the training of a model and one for the simulation of a trained model. The latter only contains the code which is required to run a simulation for given target angles.

3.1 Training

The code for the training is within the folder `static-training`. The `Main.m`-file shows an example how to train models. Further parameters that can be defined are:

- `num_models`: the number of models that should be trained.
- `max_run_time`: the maximum simulated runtime (in ms) of the training. This is required, because it might be possible that a model is unable to learn and without the usage of this parameter, the training would run indefinitely.
- `model_number`: can be set to any value in order to identify the model. This value is also used within the names of the output-files.

The training of a model `net` can be started with the command:

```
net.train(model_number,max_run_time)
```

The return-value can be 1 or 0, where 1 means that the model is trained successfully, whereas 0 means that the model is not trained successfully.

3.2 Simulation

The code for the simulation of a statically trained model is within the folder `static-simulation`. The `Main.m`-file shows an example how to start the simulation. Further parameters that can be defined are:

- `angle_start`: the initial angle at which the virtual prosthesis should start.
- `angles_to_simulate`: a list of target angles that should be simulated (in this sequence).
- `time_per_angle`: the simulated runtime (in ms) for each target angle. (total runtime = number of target angles · `time_per_angle`)

The simulation of a statically trained model `net` can be started with the command:

```
net.simulate(model_number,angle_start,angles_to_simulate,time_per_angle)
```

The return-value is a list containing one entry for each target angle. The values could be 1 (angle reached) or 0 (angle not reached).