IO EGLIF Parameters Recheck

NICCOLÒ FRASSETTO

July 15, 2025

1 Physiological Electroresponsivness Features

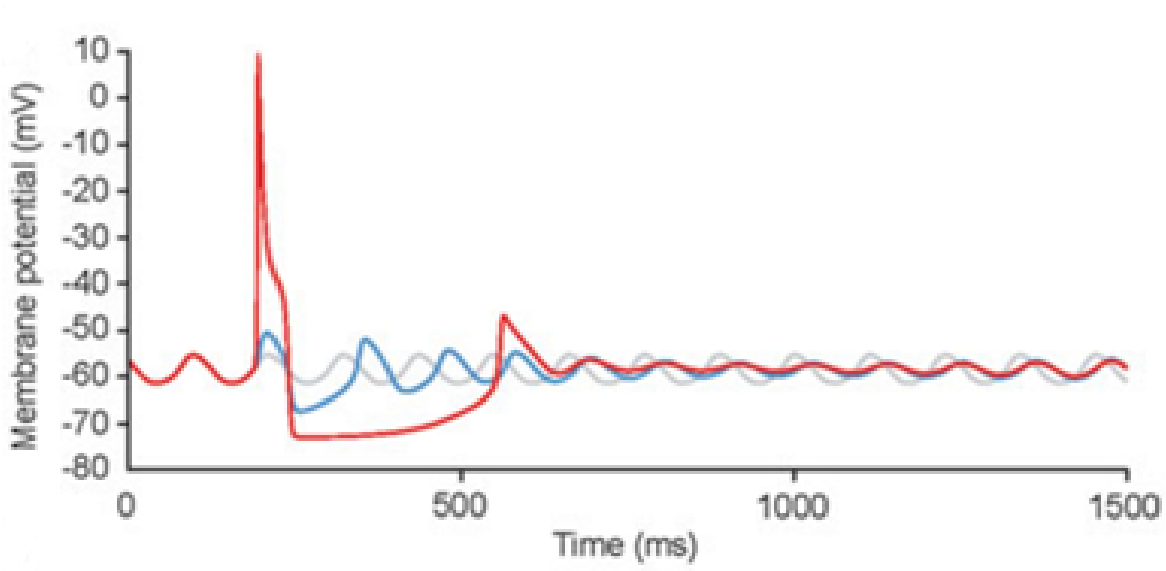


Figure 1: Spiking behavior of Inferior Olive neurons illustrating their intrinsic and network dynamics. Adapted from Loyola et al. (2022), From "Inferior Olive: All Ins and Outs" [1].

Physiological Features:

- **Membrane Potential**: average of –54.52±5.69 mV.
- **STO types**: LTO 1-3 Hz; Sinusoidal 3-10 Hz.
- **STO frequency**: 4.97 ± 1.85 Hz / range: 1.63 – 9.77 Hz.
- **STO amplitude**: 9.52 ± 5.4 mV / range: 1.13 – 21.14 mV
- **Spike**: fast sodium spike.
- **ADP**: a broad after depolarization follows the spike.
- **Spikelets**: one to seven spikelets at 130-450 Hz are superimposed on ADP.
- **AHP**: the hyperpolarization is long lasting.
- **Spikes Rhythmicity**: lasts for few hundred milliseconds (300 ms).
- **STO & Spike**: an action potential is generated on average every 10 STO cicles.

2 IO EGLIF Parameters

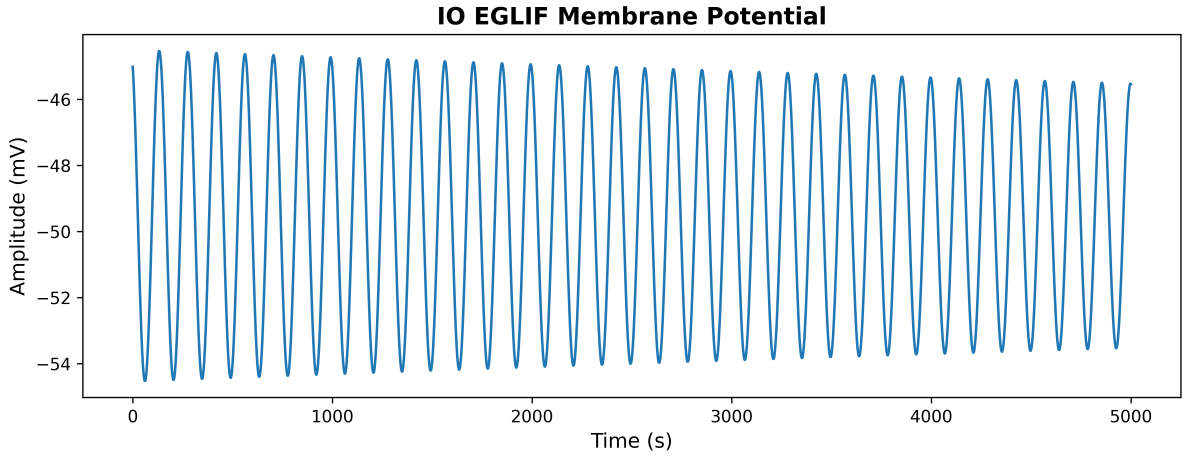
```
io:
  model: eglif_cond_alpha_multisyn
  constants:
    # Membrane Potential
    V_m: -45
    E_L: -45
    C_m: 189
    tau_m: 11
    I_e: -18.101
    k_adap: 1.928
    k_1: 0.191
    k_2: 0.091

    # V Threshold, Refractory Period and Escape rate
    V_th: -35
    t_ref: 1
    tau_V: 0.8
    lambda_0: 1.2

    # Reset on Spike
    V_reset: -45
    A1: 1810.923
    A2: 1358.197

    # Postsynaptic receptors
    tau_syn1: 1
    tau_syn2: 60
    E_rev1: 0
    E_rev2: -80
```

3 IO EGLIF STO



- Estimated STO frequency: 4.88 Hz
- Mean STO amplitude: 8.96 mV

The frequency and amplitude appear consistent with physiological values. However, the amplitude tends to decrease over time and no spikes are triggered in the absence of a stimulus. There is no variability in frequency or amplitude across neurons. Finally, STOs remain within the SSTO regime, while LTOs are not represented.

Idea: Would it be possible to introduce greater variability in amplitude so that in 1 out of 10 cycles, the membrane potential (V_m) reaches the threshold potential (V_th)?

4 DC Generator Background Noise

To analyze the spiking behavior under the influence of external input, a background noise generator was introduced. This generator was implemented as a `dc_generator` device with an amplitude of 20 and a synaptic weight of 1. Its activity was restricted to a brief 50 ms window, starting at 500 ms and stopping at 550 ms.

```
background_noise:
    device: dc_generator
    amplitude: 20
    weight: 1
    start: 500.0
    stop: 550.0
    delay: 0.1
```

4.1 IO EGLIF Rhythmic Firing

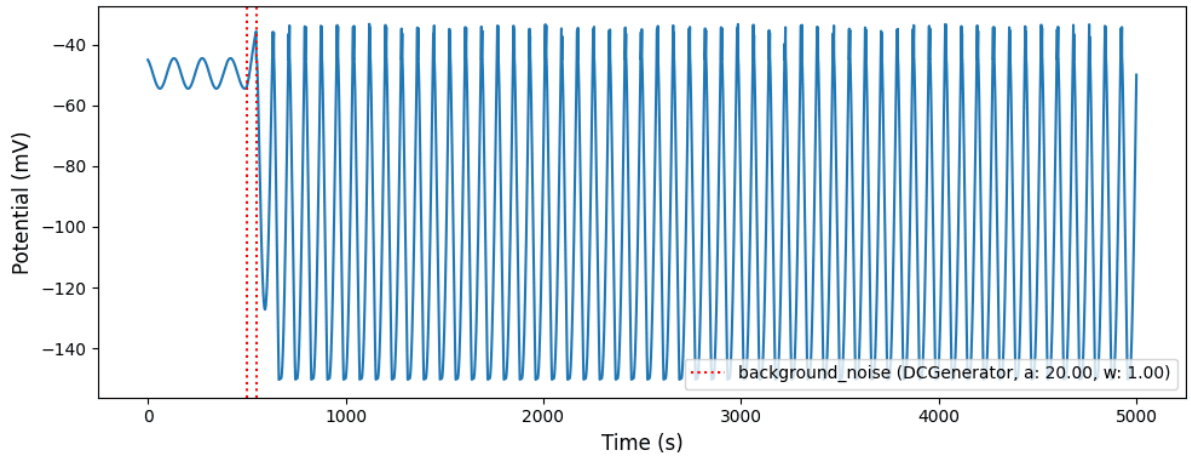


Figure 2: Complete time series of the DC generator output.

The resulting behavior deviated from expectations. Once a spike was triggered, rhythmic firing persisted throughout the entire simulation. While such rhythmic activity is physiologically plausible, it typically lasts for no more than **300 ms** under normal conditions.

4.2 IO EGLIF Spike

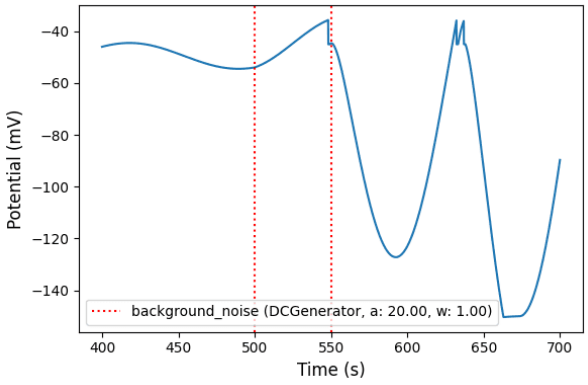


Figure 3: Zoomed-in segment of the DC generator output signal, showing detailed waveform behavior.

A DC input depolarizes the membrane potential to the threshold V_th, triggering a spike followed by an afterhyperpolarization (AHP). The AHP then induces a rebound depolarization, which leads to another spike accompanied by an afterdepolarization potential (ADP). A single spikelet is superimposed on the ADP, followed again by an AHP.

Issues:

- The afterhyperpolarization reaches approximately –150 mV, which is physiologically unrealistic.
- The afterhyperpolarization should last longer than 100 ms.
- Only a single spikelet is generated, whereas physiologically multiple spikelets are typically superimposed on the ADP.

Similar results can also be obtained using a Poisson generator device.

5 DCN-IO

To stabilize the model, a proposed solution involves connecting it to the deep cerebellar nuclei (DCN), following the cerebellar connectivity principles implemented in the BSB-NEST framework. The YAML file describing the resulting network can be found at https://github.com/FrassettoN/io-gap-junctions-network/blob/941b3d6590772abf5eb3f64de7eb9803e1fe19fa/configurations/dcn_io.yaml

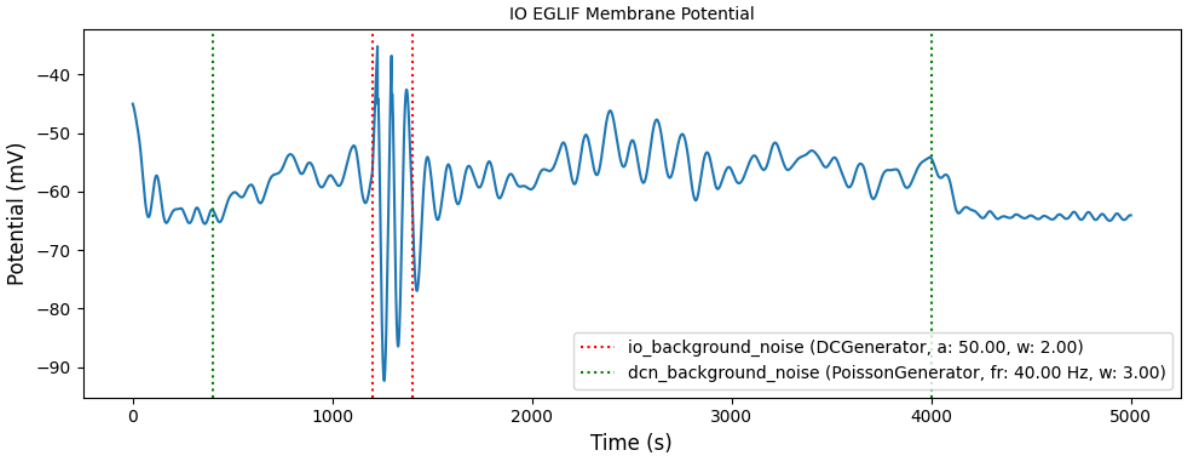


Figure 4: Membrane potential of the IO neuron when connected to the DCN

When connected to the DCN, the average membrane potential of the IO decreases to -65 mV, and the amplitude of STOs is reduced to less than 1 mV.

As a consequence, the behavior was further studied under conditions where the DCN were inhibited using a Poisson generator, replicating a state similar to Purkinje cell-mediated inhibition of the DCN in the cerebellar network. In this condition, STOs exhibited greater variability, although their mean amplitude still differed from physiological values. With DCN inhibition, spikes became more controlled; however, they were promptly interrupted in the absence of input (no spike rhytmicity).

These results suggest that the EGLIF IO model may function correctly only when embedded within a larger network, where DCN inhibition is dynamically regulated by the olivocerebellar circuit. This behavior complicates the investigation of the IO volume stand alone.

References

[1] S. Loyola, L. W. J. Bosman, J. R. De Gruijl, M. T. G. De Jeu, M. Negrello, T. M. Hoogland, and C. I. De Zeeuw, *From Inferior Olive: All Ins and Outs*, Handbook of the Cerebellum and Cerebellar Disorders. https://doi.org/10.1007/978-3-030-23810-0_43

NESTML Gap Junctions

Niccolò Frassetto

July 15, 2025

1 Generating NESTML Models with Gap Junctions Enabled

The following Python function uses NESTML to generate a neuron model with gap junctions support. It enables gap junctions during code generation by specifying the appropriate code generation options.

Listing 1: Python function to generate NESTML code with gap junctions enabled

```
import os
from pynestml.frontend.pynestml_frontend import generate_nest_target

def generate_code(neuron_model: str):
    codegen_opts = {
        "gap_junctions": {
            "enable": True,
            "gap_current_port": "I_stim",
            "membrane_potential_variable": "V_m"
        }
    }

    files = os.path.join("./", "nest_models", neuron_model + ".nestml")
    generate_nest_target(
        input_path=files,
        module_name="nestml_gap_" + neuron_model + "_module",
        suffix="_nestml",
        codegen_opts=codegen_opts
    )

    return neuron_model
```

2 Gap Junctions in `hh_psc_alpha_neuron`

To generate a gap junctions enabled model for the `hh_psc_alpha_neuron`, call the function:

```
generate_code(neuron_model="hh_psc_alpha_neuron")
```

The model was compiled using the original NESTML source code available at the following GitHub repository: hh_psc_alpha_neuron.

The model was tested by following the **Gap Junctions: Two Neuron Example** in the NEST documentation.

```
import matplotlib.pyplot as plt
import nest
import numpy

nest.Install("nestml_gap_hh_psc_alpha_neuron_module")
nest.resolution = 0.05

neuron = nest.Create("hh_psc_alpha_neuron_nestml", 2)
```

```
neuron.I_e = 650.0
neuron[0].V_m = -10.0

vm = nest.Create("voltmeter", params={"interval": 0.1})
nest.Connect(vm, neuron, "all_to_all")

with_gaps = True
if with_gaps:
    nest.Connect(
        neuron, neuron,
        {"rule": "all_to_all", "allow_autapses": False},
        {"synapse_model": "gap_junction", "weight": 0.5}
    )

nest.Simulate(250.0)

senders = vm.events["senders"]
times = vm.events["times"]
v_m_values = vm.events["V_m"]

plt.figure(1)
plt.plot(
    times[numpy.where(senders == 1)], v_m_values[numpy.where(senders == 1)], "r-",
    label="Neuron 1 (V_m = -10.0)")
plt.plot(
    times[numpy.where(senders == 2)], v_m_values[numpy.where(senders == 2)],
    "g-",
    label="Neuron 2 (V_m = -65.0)")
plt.legend(loc='upper right')
plt.xlabel("time (ms)")
plt.ylabel("membrane potential (mV)")
plt.show()
```

2.1 Results

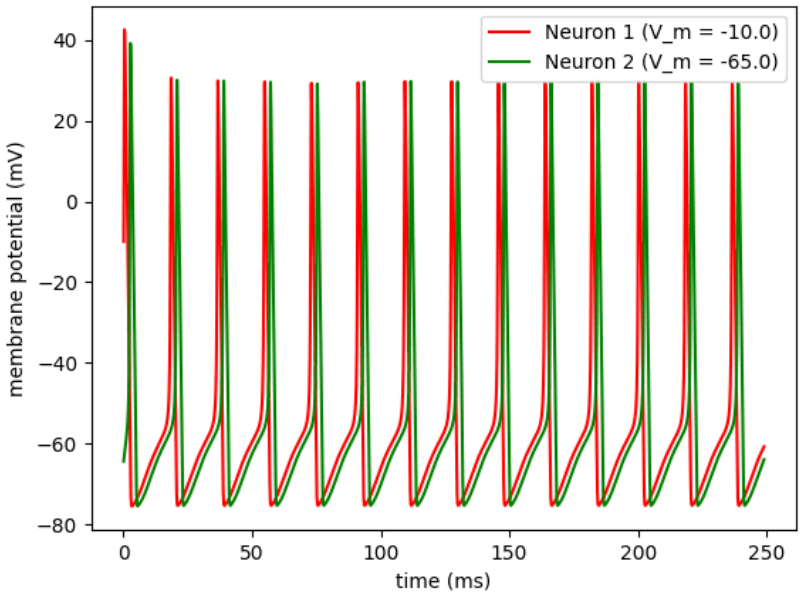


Figure 1: Membrane potential of the HH neurons without gap junctions

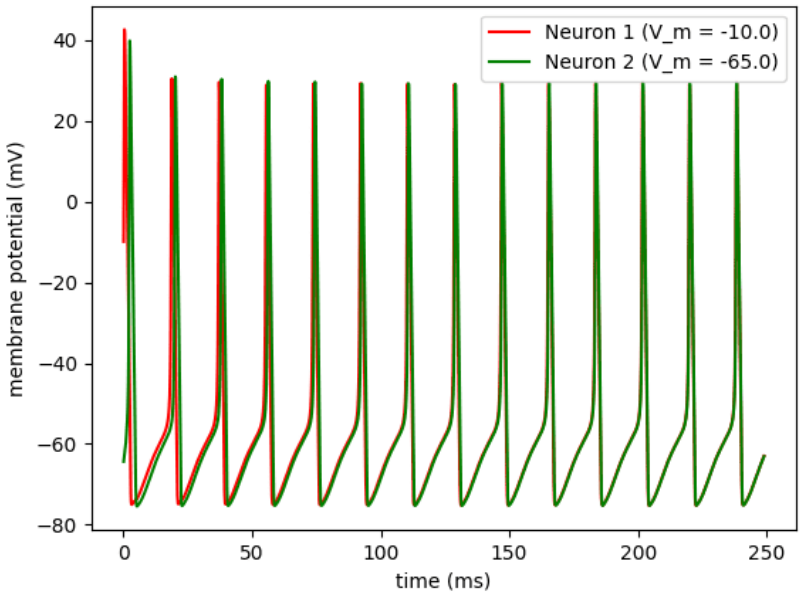


Figure 2: Membrane potential of the HH neurons with gap junctions

The results demonstrate that gap junctions effectively synchronize the membrane potentials of the HH neurons.

3 Gap Junctions in IO `eglif_cond_alpha_multisyn`

To generate a gap junctions enabled model for the `eglif_cond_alpha_multisyn`, call the function:

```
generate_code(neuron_model="eglif_cond_alpha_multisyn")
```

The model was tested by following the **Gap Junctions: Two Neuron Example** in the NEST documentation.

```
import matplotlib.pyplot as plt
import nest
import numpy

nest.Install("nestml_gap_eglif_cond_alpha_multisyn_module")

nest.resolution = 0.05

neuron = nest.Create("eglif_cond_alpha_multisyn_nestml", 2)
neuron.set({
        "V_m": -45,
        "E_L": -45,
        "C_m": 189,
        "tau_m": 11,
        "I_e": -18.101,
        "k_adap": 1.928,
        "k_1": 0.191,
        "k_2": 0.090909,

        # V Threshold, Refractory Period and Escape rate
        "V_th": -35,
        "t_ref": 1,
        "tau_V": 0.8,
        "lambda_0": 1.2,

        # Reset on Spike
        "V_reset": -45,
        "A1": 1810.923,
        "A2": 1358.197,

        # Postsynaptic receptor
        "tau_syn1": 1,
        "tau_syn2": 60,
        "E_rev1": 0,
        "E_rev2": -80,
        })

neuron[0].V_m = -48.0

vm = nest.Create("voltmeter", params={"interval": 0.1})
nest.Connect(vm, neuron, "all_to_all")

with_gaps = True
if with_gaps:
    nest.Connect(
        neuron, neuron,
        {"rule": "all_to_all", "allow_autapses": False},
        {"synapse_model": "gap_junction", "weight": 0.5}
    )

nest.Simulate(1000.0)
```

```
senders = vm.events["senders"]
times = vm.events["times"]
v_m_values = vm.events["V_m"]

plt.figure(1)
plt.plot(
    times[numpy.where(senders == 1)], v_m_values[numpy.where(senders == 1)],
    "r-",
    label="Neuron 1 (V_m = -48.0)")
plt.plot(
    times[numpy.where(senders == 2)], v_m_values[numpy.where(senders == 2)],
    "g-",
    label="Neuron 2 (V_m = -45.0)")
plt.legend(loc='upper right')
plt.xlabel("time (ms)")
plt.ylabel("membrane potential (mV)")
plt.show()
```

3.1 Results

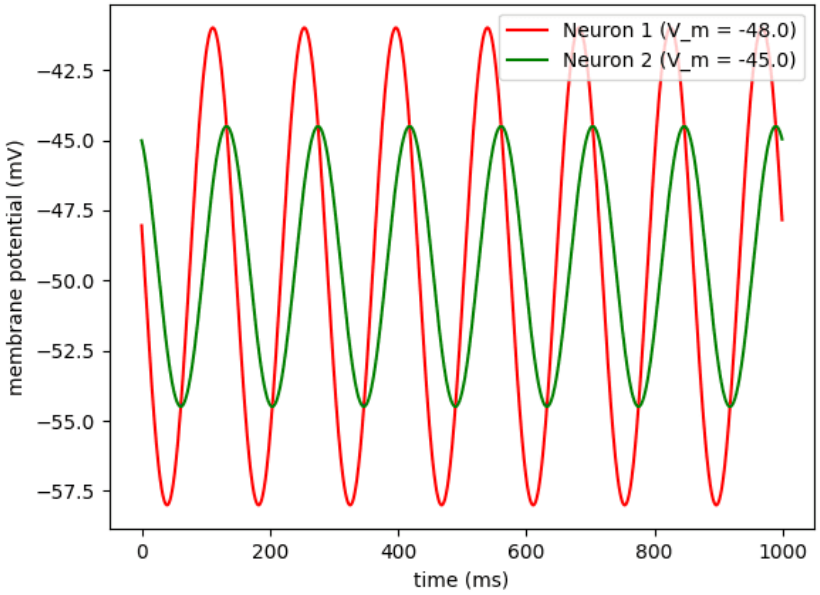


Figure 3: Membrane potential of the IO EGLIF neurons without gap junctions

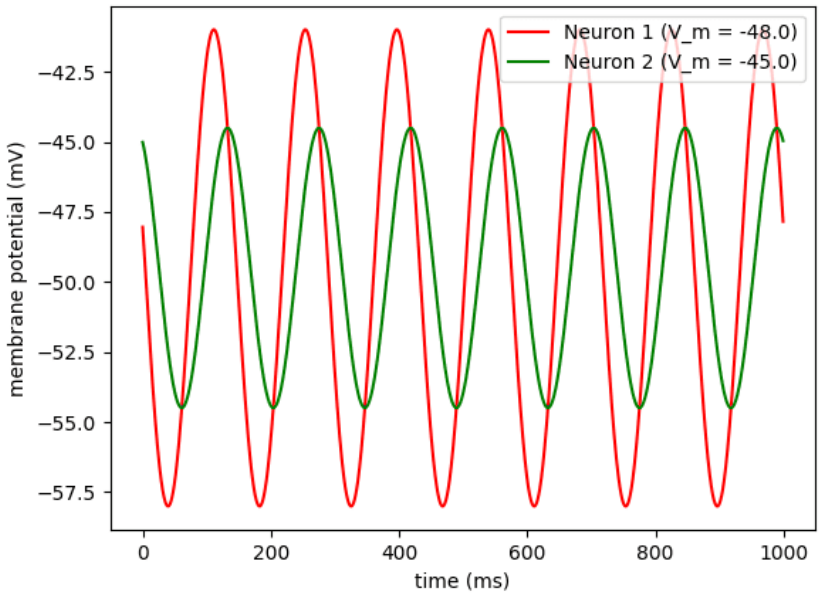


Figure 4: Membrane potential of the IO EGLIF neurons with gap junctions

The results suggest that gap junctions do not appear to synchronize the membrane potentials of the IO EGLIF neurons.