[**https://acn.ugr.es/project/spikeage/**](https://acn.ugr.es/project/spikeage/)

**SPIKEAGE - Cerebellar Spiking Modelling Accounting for Human Ageing Degradation in Motor Adaptation**

**(**[**https://brian2.readthedocs.io/en/stable/**](https://brian2.readthedocs.io/en/stable/)**)**

**https://github.com/brian-team/brian2/blob/b44119f161c19f63bca65010d9118102bfd1e6fc/docs\_sphinx/index.rst**

To create a computational model of the cerebellum and cerebellar peduncle, you can use a combination of neural network modeling, mathematical modeling of the neural dynamics, and biophysically realistic representations. Below is a simple example using Python with the Brian2 simulator, which is useful for spiking neural network models:

from brian2 import \*

# Parameters for the cerebellum model

tau\_m = 20 \* ms # Membrane time constant

v\_rest = -65 \* mV # Resting membrane potential

v\_reset = -70 \* mV # Reset voltage after spike

v\_th = -50 \* mV # Spike threshold

refractory\_period = 5 \* ms # Refractory period

# Neuron model equations

eqs = '''

dv/dt = (v\_rest - v + I) / tau\_m : volt

I : amp

'''

# Define the neuron group

neurons = NeuronGroup(1, eqs, threshold='v > v\_th', reset='v = v\_reset', refractory=refractory\_period)

# Initialize the neurons

neurons.v = v\_rest

neurons.I = 0.5 \* nA # Example input current

# Set up the network to monitor spike times

monitor = SpikeMonitor(neurons)

# Run the simulation

run(100 \* ms)

# Plot the results

plot(monitor.t / ms, monitor.i, '.k')

xlabel('Time (ms)')

ylabel('Neuron index')

show()

For a more detailed model, you would need to incorporate synaptic dynamics, various cerebellar cell types (e.g., Purkinje, granule cells), and the cerebellar peduncle's input/output pathways. Advanced models like those in the *spiking neural network* framework or using specific cerebellum modeling tools (e.g., Neuron or NEST) would also be appropriate depending on your complexity needs.

-----------------------------------------------------------------------------------------------------

Creating a computational model of the cerebellum and its cerebellar peduncles is quite an involved task, as the cerebellum's functions are highly complex and not fully understood. However, I can guide you through the process of creating a simplified model based on common computational frameworks used in neuroscience, such as the spiking neuron models or simplified population models for the cerebellum's circuitry.

in this case, we can model the **cerebellar cortex**, which consists of different types of neurons (e.g., granule cells, Purkinje cells, and interneurons) and their interactions. The **cerebellar peduncles** are pathways that carry information between the cerebellum and the brainstem, so they can be represented by the inputs and outputs that affect the cerebellar model.

Let’s start by outlining the steps for a simple spiking neuron model (using an example with **Brian2**, a popular Python library for spiking neural networks), which could be used for the cerebellum and the cerebellar peduncle pathway.

**Step-by-step guide to create a simple cerebellar spiking neuron model:**

1. **Install Dependencies** First, you need to install Brian2, a simulator for spiking neural networks:

pip install brian2

1. Cerebellar Model Setup We'll use a simple spiking neuron model to simulate the neurons in the cerebellum and the cerebellar peduncle, such as Leaky Integrate-and-Fire (LIF) neurons, which are commonly used in computational neuroscience.
2. Define the Neuron Parameters and the Network Here's a basic example in Python using Brian2 to simulate a small subset of the cerebellar circuit. This will model excitatory neurons (granule cells) and inhibitory neurons (Purkinje cells), as well as the connections between them.

from brian2 import \*

# Neuron parameters

tau\_m = 20\*ms # Membrane time constant

v\_rest = -65\*mV # Resting potential

v\_threshold = -50\*mV # Spike threshold

v\_reset = -65\*mV # Reset potential

# Define the neuronal equations

eqs = '''

dv/dt = (v\_rest - v + I)/tau\_m : volt

I : volt

'''

# Create the population of Granule cells and Purkinje cells

num\_granule\_cells = 100

num\_purkinje\_cells = 30

granule\_cells = NeuronGroup(num\_granule\_cells, eqs, threshold='v > v\_threshold', reset='v = v\_reset', method='euler')

purkinje\_cells = NeuronGroup(num\_purkinje\_cells, eqs, threshold='v > v\_threshold', reset='v = v\_reset', method='euler')

# Initial conditions

granule\_cells.v = v\_rest

purkinje\_cells.v = v\_rest

# Synaptic connections

syn\_granule\_to\_purkinje = Synapses(granule\_cells, purkinje\_cells, on\_pre='v += 5\*mV')

syn\_granule\_to\_purkinje.connect(p=0.1) # 10% probability of connection

# External input to granule cells (simulating cerebellar input)

granule\_cells.I = 1.5\*mV # External current

# Monitor spikes

spikes\_granule = SpikeMonitor(granule\_cells)

spikes\_purkinje = SpikeMonitor(purkinje\_cells)

# Run the simulation

run(1\*second)

# Plot the spike raster plot

figure(figsize=(10, 4))

subplot(211)

plot(spikes\_granule.t/ms, spikes\_granule.i, '.k')

title('Granule Cell Spikes')

xlabel('Time (ms)')

ylabel('Neuron Index')

subplot(212)

plot(spikes\_purkinje.t/ms, spikes\_purkinje.i, '.r')

title('Purkinje Cell Spikes')

xlabel('Time (ms)')

ylabel('Neuron Index')

show()

**Explanation of the Model:**

* **Granule cells**: These are the excitatory neurons in the cerebellar cortex. In this model, their spiking is driven by an external input.
* **Purkinje cells**: These are inhibitory neurons that receive input from granule cells. When a granule cell spikes, it sends an excitatory signal to the Purkinje cells, which inhibits the output signal from the cerebellum.
* **Synapses**: The synapses between granule cells and Purkinje cells are modeled with a connection probability (p=0.1), meaning each granule cell has a 10% chance of being connected to a Purkinje cell.
* **Spike Monitors**: These track the spikes from both granule and Purkinje cells.

**4. Cerebellar Peduncle Input/Output**

To model the cerebellar peduncles, you could create pathways that bring in signals from the brainstem (e.g., from mossy fibers or climbing fibers) into the cerebellum, or send the cerebellum's output to other areas like the deep cerebellar nuclei. You could do this by:

* **Mossy fibers**: Input from the brainstem to granule cells.
* **Climbing fibers**: Input to Purkinje cells directly, possibly with complex dynamics like "error correction."
* **Deep cerebellar nuclei**: Output neurons that integrate Purkinje cell inhibition and send signals to motor systems.

# Example of adding mossy fiber input to the granule cells

mossy\_fibers = PoissonGroup(1, rates=30\*Hz) # Random Poisson spike train

syn\_mossy\_to\_granule = Synapses(mossy\_fibers, granule\_cells, on\_pre='I += 0.2\*mV')

syn\_mossy\_to\_granule.connect(p=1.0)

# Example of adding climbing fiber input to Purkinje cells (error signal)

climbing\_fibers = PoissonGroup(1, rates=10\*Hz)

syn\_climbing\_to\_purkinje = Synapses(climbing\_fibers, purkinje\_cells, on\_pre='I += 0.5\*mV')

syn\_climbing\_to\_purkinje.connect(p=1.0)

# Now run the model with this added input

run(1\*second)

**Additional Considerations:**

* The cerebellum's function (e.g., motor control, coordination) could be tested by modeling specific patterns of input/output to see how well the network can replicate known cerebellar behaviors like learning, timing, or error correction.
* More detailed models could include **plasticity rules** (such as Long-Term Depression or Potentiation) to simulate learning within the cerebellum.
* You might also want to explore multi-compartment models or include more complex cell types, such as inhibitory interneurons (stellate and basket cells).

**Conclusion:**

This simple model provides a starting point for simulating cerebellar circuits using spiking neuron models. If you need to explore more complex or specific cerebellar structures, such as cerebellar peduncles' role in motor output, you may need to extend the network by adding deeper layers or integrating additional brainstem structures into the model.

------------------------------------------------------------------------------------------------------------

Integrating \*\*motor control\*\* into the cerebellar computational model adds another layer of complexity, as it requires modeling the interaction between sensory feedback, motor outputs, and the cerebellar circuits responsible for motor learning and coordination. The cerebellum plays a key role in fine-tuning motor commands, adjusting movements based on sensory feedback, and learning motor tasks (like coordination, balance, and precision).

To model motor control, we need to consider several factors:

1. \*\*Motor Output\*\*: The cerebellum communicates with motor centers in the brainstem, such as the deep cerebellar nuclei (DCN), which then influence motor areas like the spinal cord and muscles. These outputs are typically related to movements like muscle tone, timing, and coordination.

2. \*\*Motor Learning (Cerebellar Plasticity)\*\*: The cerebellum is thought to modify motor outputs based on feedback using learning mechanisms like \*\*Long-Term Depression (LTD)\*\* at the synapses between granule cells and Purkinje cells, which adjusts the strength of connections over time to optimize motor control.

3. \*\*Sensory Input\*\*: Motor control is based on continuous feedback from the body (via sensory inputs like proprioception, vision, etc.) to adjust motor commands and correct errors.

To keep things manageable, let's expand the previous cerebellar model to simulate a basic \*\*motor control loop\*\* with sensory input, cerebellar processing, and motor output (via deep cerebellar nuclei). This will include:

- A sensory input representing external stimuli (e.g., motor intention).

- Cerebellar neurons (granule and Purkinje cells) that process this input.

- Deep cerebellar nuclei neurons that produce motor commands to an actuator or motor system.

- A feedback loop simulating error correction based on sensory feedback.

### Key Features of the Updated Model:

1. \*\*Sensory Input\*\*: Represent sensory feedback as a fluctuating signal that influences the cerebellar model.

2. \*\*Motor Output\*\*: Deep cerebellar nuclei (DCN) produce output signals that simulate motor commands.

3. \*\*Feedback\*\*: Sensory input and motor output are fed back into the cerebellum to modify movement over time.

### Python Code Example with Motor Control

Here's an updated \*\*Brian2\*\* model that includes basic motor control and feedback loops:

```python

from brian2 import \*

# Simulation parameters

defaultclock.dt = 0.1\*ms

duration = 1000\*ms

# Neuron parameters

v\_rest = -65 \* mV

v\_reset = -70 \* mV

v\_thresh = -55 \* mV

tau\_m = 20 \* ms

R = 1 \* MOhm

# Create neurons

# Granule cell parameters

granule\_eqs = '''

dv/dt = (v\_rest - v + I)/tau\_m : volt

I : amp

'''

# Purkinje cell parameters

purkinje\_eqs = '''

dv/dt = (v\_rest - v + I)/tau\_m : volt

I : amp

'''

# Deep cerebellar nuclei (DCN) neurons for motor output

dcn\_eqs = '''

dv/dt = (v\_rest - v + I)/tau\_m : volt

I : amp

'''

# Create Granule, Purkinje, and DCN neurons

granule\_neurons = NeuronGroup(100, granule\_eqs, threshold='v > v\_thresh', reset='v = v\_reset', method='exact')

purkinje\_neurons = NeuronGroup(20, purkinje\_eqs, threshold='v > v\_thresh', reset='v = v\_reset', method='exact')

dcn\_neurons = NeuronGroup(10, dcn\_eqs, threshold='v > v\_thresh', reset='v = v\_reset', method='exact')

# Connect neurons (mossy fibers to granule cells, granule cells to Purkinje cells, Purkinje cells to DCN)

synapses\_granule = Synapses(granule\_neurons, purkinje\_neurons, on\_pre='I += 0.1 \* nA')

synapses\_granule.connect(p = 0.1) # 10% connectivity

synapses\_purkinje = Synapses(purkinje\_neurons, dcn\_neurons, on\_pre='I += 0.2 \* nA')

synapses\_purkinje.connect(p = 0.5) # 50% connectivity

# Initial conditions

granule\_neurons.v = v\_rest

purkinje\_neurons.v = v\_rest

dcn\_neurons.v = v\_rest

# Create a sensory input (simulated as a sinusoidal signal to represent a varying motor demand)

sensory\_input = TimedArray(np.sin(2 \* np.pi \* (1/500.) \* np.arange(0, duration/ms)) \* 0.5 \* nA, dt=defaultclock.dt)

# Monitor the spikes

granule\_spikes = SpikeMonitor(granule\_neurons)

purkinje\_spikes = SpikeMonitor(purkinje\_neurons)

dcn\_spikes = SpikeMonitor(dcn\_neurons)

# Add sensory input to granule neurons (representing sensory feedback affecting motor control)

granule\_neurons.I = sensory\_input

# Run the simulation

run(duration)

# Plot the results

figure(figsize=(12, 9))

subplot(311)

plot(granule\_spikes.t/ms, granule\_spikes.i, '.k')

xlabel('Time (ms)')

ylabel('Granule Neuron Index')

title('Granule Cell Spikes')

subplot(312)

plot(purkinje\_spikes.t/ms, purkinje\_spikes.i, '.r')

xlabel('Time (ms)')

ylabel('Purkinje Neuron Index')

title('Purkinje Cell Spikes')

subplot(313)

plot(dcn\_spikes.t/ms, dcn\_spikes.i, '.b')

xlabel('Time (ms)')

ylabel('DCN Neuron Index')

title('Deep Cerebellar Nuclei Spikes (Motor Output)')

show()

```

### Explanation of the Updates:

1. \*\*Sensory Input\*\*:

- The `sensory\_input` variable simulates the feedback coming from external sensory sources, such as proprioception, vision, etc. In this example, it's a simple sinusoidal function representing a fluctuating motor demand.

- This input affects the \*\*granule neurons\*\*, which process sensory feedback and influence the downstream cerebellar neurons.

2. \*\*Granule, Purkinje, and Deep Cerebellar Nuclei Neurons\*\*:

- \*\*Granule cells\*\* receive sensory input and pass information to the \*\*Purkinje cells\*\*.

- \*\*Purkinje cells\*\* inhibit the \*\*deep cerebellar nuclei (DCN)\*\*, which simulate motor output neurons.

- The \*\*DCN neurons\*\* generate motor commands based on the output of the cerebellar circuit.

3. \*\*Synaptic Connections\*\*:

- Granule cells connect to Purkinje cells with simple excitatory synapses.

- Purkinje cells inhibit the DCN neurons, which simulate motor output (e.g., controlling motor actions).

4. \*\*Motor Output\*\*:

- The motor output is recorded from the \*\*DCN neurons\*\*. The spikes from DCN neurons represent motor commands sent to motor control areas like the brainstem and spinal cord.

### Motor Learning and Error Correction:

In the cerebellum, motor learning is thought to occur through \*\*synaptic plasticity\*\*, specifically \*\*Long-Term Depression (LTD)\*\* at the synapse between granule and Purkinje cells. The basic idea is that when sensory feedback does not match the expected motor outcome, the cerebellum adjusts its output to reduce the error in future movements. This can be modeled by introducing changes in synaptic strength over time based on error signals (although we’ve kept this model simple without implementing LTD explicitly).

### Future Extensions:

1. \*\*Motor Learning\*\*: Implement \*\*LTD\*\* (or Long-Term Potentiation, LTP) based on the timing of spikes between granule cells and Purkinje cells to simulate learning and adaptation in motor control.

2. \*\*Dynamic Feedback\*\*: Add more sophisticated models of sensory feedback, such as proprioception, vision, or feedback from muscles, to affect motor control in real time.

3. \*\*Complex Motor Tasks\*\*: Model more complex motor tasks, such as reach-and-grasp movements or locomotion, which involve multiple brain regions and real-time motor adjustments.

This model offers a foundational framework to study cerebellar contributions to motor control and can be extended to simulate more complex motor tasks, cerebellar diseases (e.g., ataxia), or neuroplasticity in motor learning.