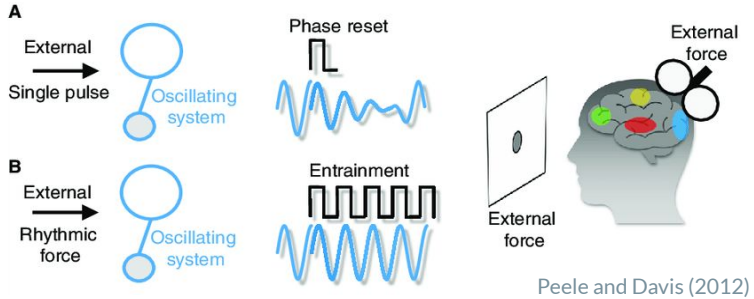
A Computational Model of Auditory Entrainment

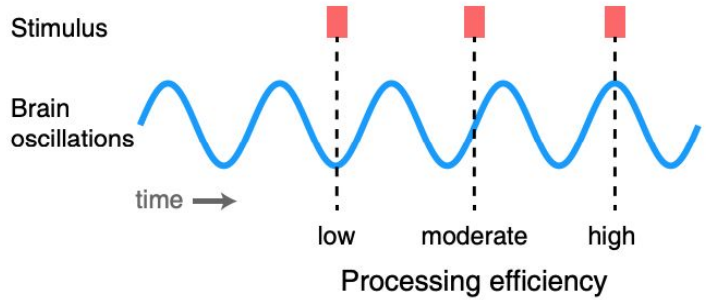
Shaan Verma

Auditory Entrainment

- **Entrainment:** Synchronization of an oscillatory system(s) to an external rhythm/system
  - More efficient processing of stimuli
- Entrainment of auditory neurons depends on properties of input stimuli: amplitude (volume), frequency, and rhythm
- Entrainment is more likely to occur...
  - As stimulus frequency approaches intrinsic frequency and as stimulation intensity increases
- Plays an important role in auditory selective attention



Peele and Davis (2012)



Lakatos et al. (2019)

Project Objective

- Implement the computational model of auditory neurons presented in the paper, A Computational Model of Auditory Entrainment.
- Explore the behaviour of our model when the intrinsic neuronal properties change.
- Give the neuron a pure tone input (sinusoidal current) instead of a step current.

Methods

- Primary auditory neurons were modeled as phase-locked leaky integrate-and-fire (PL-LIF) cells by modifying the standard LIF equation as follows:

$$\frac{dv}{dt} = \frac{-(v - v_{rest}) + r * i(t) + a * sin(2\pi * f * t + b)}{\tau}$$

where ***r*** is resistance and ***i(t)*** is input current, ***a*** is the oscillation amplitude, ***f*** is the oscillation frequency, and ***b*** is the phase.

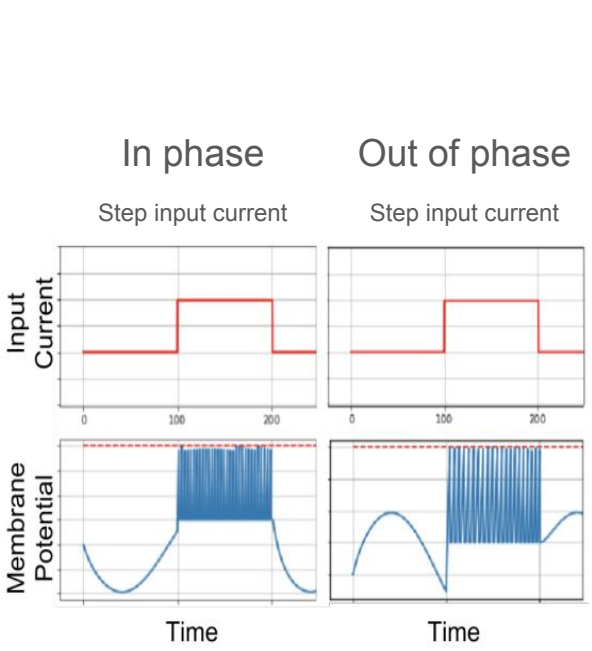
- The authors determined the oscillation frequency and phase by two entrainment functions: greedy entrainment and attended entrainment, we didn't use either.

Implementation

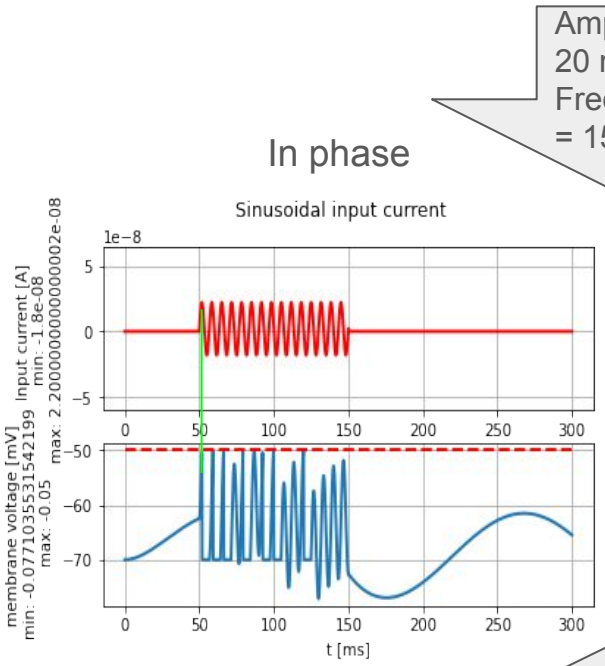
```
def simulate_LIF_neuron(input_current,simulation_time=2*b2.ms,v_rest=-70*b2.mV,
                        v_reset=-70*b2.mV,firing_threshold=-50*b2.mV,
                        membrane_resistance=10*b2.Mohm,
                        membrane_time_scale=20*b2.ms,
                        abs_refractory_period=5*b2.ms,
                        neuron_amp = 10*b2.mV,
                        neuron_freq = 5*b2.Hz,
                        neuron_phase = 2*b2.pi):

    # differential equation of Leaky Integrate-and-Fire model
    pi = b2.pi
    eqs = """
    dv/dt =
    ( -(v-v_rest) + membrane_resistance * input_current(t,i) + neuron_amp * sin(2*pi
    * t * neuron_freq + neuron_phase)) / membrane_time_scale : volt (unless
    refractory)"""

    # LIF neuron using Brian2 library
    neuron = b2.NeuronGroup(
        1, model=eqs, reset="v=v_reset", threshold="v>firing_threshold",
        refractory=abs_refractory_period, method="euler")
    neuron.v = v_rest  # set initial value

    # monitoring membrane potential of neuron and injecting current
    state_monitor = b2.StateMonitor(neuron, ["v"], record=True)
    spike_monitor = b2.SpikeMonitor(neuron)
    # run the simulation
    b2.run(simulation_time)

    #calculate the firing rate in Hz
    fr = state_monitor.num_spikes/sim_time

    # calculate coefficient of variation
    ISIs = state_monitor.spike_trains()[0][1:]-state_monitor.spike_trains()[0][:-1]
    cv = np.std(ISIs)/np.mean(ISIs)

    return state_monitor, spike_monitor, fr, cv
```

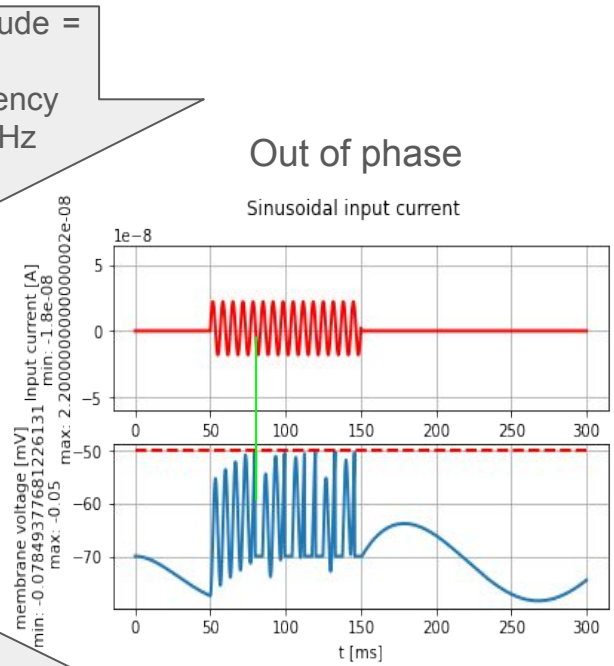
```
# neuron in phase with our input current (entrained/phase-locked)
b2.start_scope()

sinusoidal_current = input_factory.get_sinusoidal_current(500, 1500,
unit_time=0.1 * b2.ms, amplitude=20* b2.namp, frequency=150 * b2.Hz,
direct_current=2. * b2.namp)

# run the LIF model
(state_monitor, spike_monitor,fr,cv) = simulate_LIF_neuron
(input_current=sinusoidal_current, simulation_time=300* b2.ms, neuron_freq =
5*b2.Hz, neuron_phase = 2*b2.pi)
```

Results



Amplitude = 20 nA
Frequency = 150 Hz





Amplitude = 10 mV
Frequency = 5 Hz

Results (continued)

Entrainment	Average Number of Spikes
In Phase	7.55
Out of Phase	5.75

Neuron frequencies determined by: linspace(2*Hz,12*Hz,10)

- Neurons in phase with the current have a higher spike firing rate than those out of phase with the current
- This echoes the results found in the paper for the greedy entrainment function

Q/A and Discussion