Architectures of Intelligence
Assignment 5: Part 2

Matthijs Prinsen *(S4003365)*
Marinus van den Ende *(s5460484)*
Group 75

December 15, 2024

1 Results

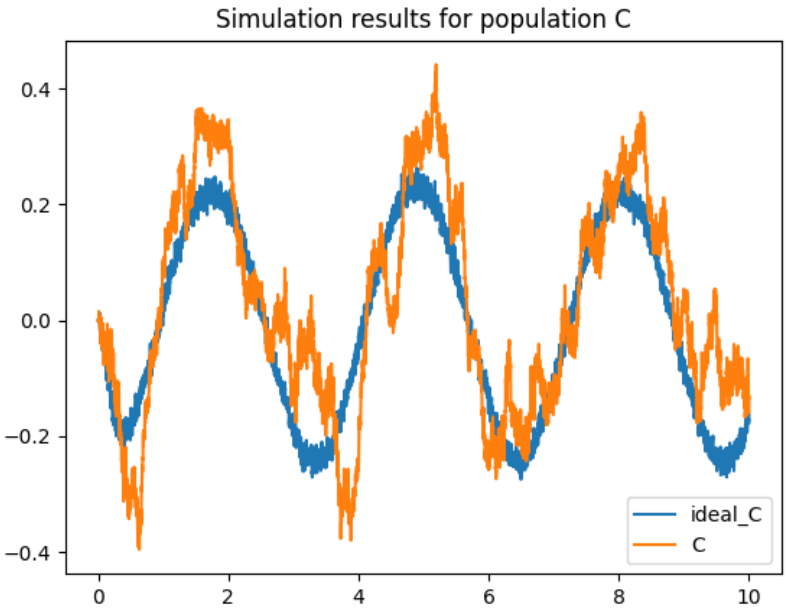

Figure 1: Ensemble C: Ideal and actual values

2 Explanations

In order to dissect the functions for simulating neuronal activity, we first need to start at the `run_neurons` function and then build our understanding of the `compute_responses` function and finally the `compute_training_curves` function.

2.1 Run Neurons

In the function `run_neurons`, we first set the subthreshold evolution of the LIF neuron voltage. This is the value by which the voltage will change for each neuron in the ensemble.

```
dV = dt * (input[i] - v[i]) / t_rc  # the LIF voltage change equation
v[i] += dV # update the voltage of each neuron
```

This Python equation follows the formula below, taken from the book *How to Build a Brain*, page 396 (Eliasmith, 2015).

$$\dot{V}(t) = -\frac{1}{\tau_{RC}}\left(V(t) - \mathbf{J}(\mathbf{x})R\right)$$

Thereafter, we have a couple of `if` statements that define the voltage range and determine whether a neuron fires.

The first `if` statement resets the voltage to zero if it goes below zero.

```
if v[i] < 0:
    v[i] = 0
```

The second `if` statement checks whether we are still in the refractory period, meaning the neuron has fired recently. If we are, we reset the voltage to zero and decrease the refractory time left.

```
if ref[i] > 0:
    v[i] = 0
    ref[i] -= dt
```

The third `if else` statement determines whether that neuron fired or not. The neurons in this implementation fire if their voltage hits 1. If the neuron fires, we record that, reset its voltage, and set its refractory period. If it does not fire, we leave it as is and record that it did not fire.

```
if v[i] > 1:
    spikes.append(True)  # spike
    v[i] = 0  # reset the voltage
    ref[i] = t_ref  # and set the refractory period
else:
    spikes.append(False)
```

Finally, we `return` a list of `True` or `False` values of the neurons that fired.

```
return spikes
```

2.2 Compute Responses

The function `compute_responses` calculates the response rate of the neurons in hertz (Hz). We calculate the response rate over a period of half a second by default, as is used in our implementation.

In the `for` loop, we encode the input `x` into a voltage representation that we can later use to calculate the LIF voltage change. We then randomly assign a voltage for each neuron between 0 and 1.

```
# compute input corresponding to x
input = []
for i in range(N):
    input.append(x * encoder[i] * gain[i] + bias[i])
    v[i] = np.random.uniform(0, 1)  # randomize the initial voltage level
```

This Python equation closely follows the right-hand side of the formula below, taken from the book *How to Build a Brain*, page 396 (Eliasmith, 2015).

$$\delta(t - t_{im}) = G_i\left[\alpha_i\langle \mathbf{x}\mathbf{e}_i\rangle + J_i^{\text{bias}}\right]$$

In the `while` loop, we simulate the neuron activity for a certain period of time, determined by the `time_limit` parameter, and use the `run_neurons` function to get a list of the neurons that spiked.

Thereafter, we can count the number of spikes that occurred and update our spike count.

```
# feed the input into the population for a given amount of time
t = 0
while t < time_limit:
    spikes = run_neurons(input, v, ref)
    for i, s in enumerate(spikes):
        if s:
            count[i] += 1
    t += dt
```

Finally, we can then `return` a list of spike rates for each neuron in the ensemble.

```
return [c / time_limit for c in count]  # return the spike rate (in Hz)
```

2.3 Compute Tuning Curves

In the function `compute_tuning_curves`, we generate a list of test values `x_values` with the same length as `N_samples`.

```
x_values = [i * 2.0 / N_samples - 1.0 for i in range(N_samples)]
```

Thereafter, for each value in `x_values`, we use the `compute_responses` function to find the spike rate of the neurons.

```
A = []
for x in x_values:
    response = compute_response(x, encoder, gain, bias)
    A.append(response)
```

Finally, we return a `tuple` of the inputs and response rates of the neurons.

```
return x_values, A
```

References

Eliasmith, C. (2015). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press. https://www.amazon.com/How-Build-Brain-Architecture-Architectures/dp/0190262125