Coursework for
4G3 Computational Neuroscience
Assignment 2

Máté Lengyel, Timothy O'Leary

assigned: March 10, 2020
due: March 24, 2020, 4pm
submit through Moodle
don't forget to attach a cover page

Question I. Hopfield network **[15 points]**

In this exercise your task will be to analyse the capacity of the (binary) Hopfield network (with unit activites 0 and 1, as discussed during the lectures, *not* -1 and $+1$ as found in some references). Beside using the lecture notes, you may find it useful to read the original paper by Hopfield (1982) downloadable from the module's Moodle page (note the minor differences in notation between the lecture notes and the paper).

Exercises:

1. Based on the calculations on pages 5-7 of the lecture slides, predict the probability of recalling a bit incorrectly ('error probability') in a network of 100 neurons as you store 1, 2, ..., 100, 200, 500, 1000 random patterns in which each bit has 0.5 probability of being 1, and bits are sampled independently from each other and across memorised patterns. Plot the error probability against number of patterns stored on a *logarithmic* scale. Explain how you obtained your result.

2. Plot the same graph for a network of 1000 neurons. Explain how you obtained your result.

3. Check your calculations above by running simulations of a Hopfield network with 100 neurons with the (binary) dynamics discussed in lectures. Store different numbers of memories (try the same numbers as above) and measure the error probability. For each network you create try to recall several of the stored patterns and plot the average error probability across those. For low numbers of stored patterns you might want to create many networks and average across those, too. Plot in the same co-ordinate system the results of your numerical simulations together with your analytical predictions (error probability against log number of stored patterns).

4. In the lectures slides (and the Hopfield paper, and the exercises above), the analysis of error probability assumes that the network is already in one of the originally stored states. The practically more relevant case is when the network starts (as an initial condition) from a noisy and partial version of one of the originally stored memories. Simulate this case again for a network of 100 neurons with different numbers of patterns stored (try the same numbers as above). Start your network by initializing it at one of the stored patterns, but flip each bit randomly with some probability ('input noise level'). Try the following input noise levels: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 1.0.

   (a) Again, plot the error probability as a function of the (log) number of stored patterns for different input noise levels in the same co-ordinate system.

   (b) Comment on your results: why are they different from what you obtained in the previous exercises?

   (c) What is happening in those cases when the input noise level is above 0.5? How do they compare to the performance you see for input levels below 0.5? Explain your results.

Question II. Exploration of a physiological model of a spiking neuron

[15 points]

The following ordinary differential equations describe the classic Hodgkin-Huxley model of the action potential:

$$\dot{v} = -g_{\mathrm{Na}}\, m^3\, h\, (v - e_{\mathrm{Na}}) - g_{\mathrm{K}}\, n^4\, (v - e_{\mathrm{K}}) - g_{\mathrm{L}}\, (v - e_L) + I_{\mathrm{ext}} \tag{1}$$

$$\dot{m} = \alpha_{\mathrm{m}}(v)\, (1 - m) - \beta_{\mathrm{m}}(v)\, m \tag{2}$$

$$\dot{h} = \alpha_{\mathrm{h}}(v)\, (1 - h) - \beta_{\mathrm{h}}(v)\, h \tag{3}$$

$$\dot{n} = \alpha_{\mathrm{n}}(v)\, (1 - n) - \beta_{\mathrm{n}}(v)\, n \tag{4}$$

where v denotes membrane potential (in mV) and n, m and h are state variables for the membrane currents. The α and β functions describe the rates of ion channel gating as a function of voltage and the g_{x}, e_{x} are conductance densities (normalised to membrane capacitance) and reversal potentials (in mV) of the different ionic currents, Na (sodium), K (potassium) and L (leak). I_{ext} is externally applied current (in mA/nF). Time is in units of milliseconds.

The goal of this question is to understand how this physiologically realistic model responds to external input and how this differs from simplified models. You will need to numerically integrate this equation using MATLAB or Python. For all simulations use (forward) Euler integration with a fixed timestep of 0.001 ms. An m-file is supplied (hh.m, downloadable from the module's Moodle page) that defines the α and β functions. The values of the remaining parameters (in appropriate units) are provided in Table 1 below:

	[μS/nF]		[mV]
g_{Na}	120	e_{Na}	115
g_{K}	36	e_{K}	-12
g_{L}	0.3	e_{L}	10.6

Table 1. Parameter values.

Exercises:

1. Using an appropriate initialisation, simulate the response of the system defined by equations 1-4 to 200 ms-long pulses of applied current (I_{ext}) with amplitudes in the range 0.1 – 5 mA/nF.

   Generate example plots showing the membrane potential as a function of time. Plot action potential frequency as a function of input amplitude. Comment on the shape of the plot. What is the input threshold for inducing firing?

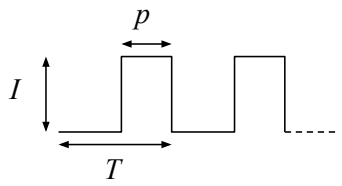

Figure 1. Current input waveform.

2. Write code to generate periodic square pulse input with variable period (T), pulse width (p) and amplitude (I) as shown in Figure 1. Systematically explore the effect of driving the model with such an input. To begin with, include the following input parameters in your exploration:

$$T = 10, 11, 12, ..., 20 \text{ ms with } p = 5 \text{ ms, and } I = 2.3 \text{ mA/nF}$$

   Summarise your data graphically, giving example membrane potential plots of any relevant or noteworthy phenomena. Comment on the biological relevance of your findings and provide a qualitative explanation for the phenomena you observe. How do your findings relate to your answer in the previous exercise? Is it possible to elicit a spike with a negative current pulse?

   Do your findings pose a problem for commonly used simplifications of neurons such as rate models and leaky integrate-and-fire models?