4G3 Coursework 2

1 Hopfield Network

In this section, the capacity of the binary Hopfield network was analysed. The Hopfield network describes a stored memory/pattern in terms of the set of neurons that fire when that memory is recalled. Therefore, if the neurons in a network are assumed to take one of two binary states (firing or not firing) then a memory can be described as a point in state space. The capacity is a measure of the reliability of memory recollection. It may be quantified as the number of patterns that may be stored in the network before bit errors become significant. Using this definition, when the networks were intialised from memory states, it was seen that the capacity of the network was roughly 10% of its size. That is, if you imagine that the curves in figure 3 are approximated by a ramp, the capacity would be the point at which the upward slope of the ramp begins. Memory recollection error was found to increase with 'distance' of initial states from memory states. This 'distance' from memory states, i.e. noise, was formed by flipping state bits randomly according to a Bernoulli distribution with parameter, p_{flip}. When the networks were initialised from noisy states the probability of error during memory recollection was higher as expected.

The Hopfield network abides by the Hebbian rule of synaptic plasticity i.e. "neurons that fire together wire together". This is modelled by the following equation:

$$\begin{aligned} W_{ij} &= \sum_{m=1}^{M} \left(r_i^{(m)} - \frac{1}{2} \right) \left(r_j^{(m)} - \frac{1}{2} \right) \\ W_{ii} &= 0 \end{aligned} \tag{1}$$

where W_{ij} is the weight (strength) of the connection between pre-synaptic neuron j and post-synaptic neuron i, and r_j, r_i are the (binary) states of the corresponding neurons, m is a stored pattern/memory, and M is the number of stored patterns. Note one peculiarity of the Hopfield network: neurons also wire together if neither of them fire.

1.1 Analytical Error Predictions

In this subsection, error predictions were made from analytical approximations. Firstly, the state-space representation of each memory was drawn randomly and independently from a uniform Bernoulli distribution. Reliable memory recollection implies that memory states are stable. That is, if at time t, bit k is initialised with memory m i.e. $r_j(t) = r_j^{(m)} \forall j$, we require that at time $t + \Delta t$ that the value of bit t is unchanged i.e. $r_k(t+1) = r_k^{(m)}$.

How do we show that stable states exist in our Hopfield network, and if they do, how do we show that memories are stables states? In the lecture slides, it was shown that stable states must exist if a time-dependent Lyupanov (energy) function could be defined on the state space. To ensure the existence of stable points, this function has two main criteria:

1. It is non-increasing.
2. It is bounded from below i.e. it approaches a lower limit.

In his paper, Hopfield defined the following energy function:

$$E(\mathbf{r}(t)) = -\frac{1}{2}\sum_{i}\sum_{j\neq i} W_{ij} r_i(t) r_j(t) \tag{2}$$

where W_{ij} is the weight of the connection between pre-synaptic neuron j and post-synaptic neuron i, $r_j(t)$ and $r_i(t)$ are the states of those respective neurons at time t, and $\mathbf{r}$ is a vector representing the state of all neurons in the network. From this equation, it was then shown that if a single neuron k is updated, the change in the energy function after a single time-step, Δt is given as:

$$E(\mathbf{r}(t+\Delta t)) - E(\mathbf{r}(t)) = -[r_k(t+\Delta t) - r_k(t)] \underbrace{\sum_{j\neq k} W_{kj} r_j(t)}_{H_k(t)} \tag{3}$$

where $H_k(t)$ is known as the field of neuron k. Note that the field is simply the weighted sum of the inputs into the neuron. It was shown in the lecture slides that if the value of neuron k is activated when the field is positive, the two criteria given above for a Lyupanov function are satisfied. The value of bit k at the next time-step is given as $F(H_k(t))$ where F is the activation function. The activation function is commonly chosen to be a sigmoid function such as the inverse tangent function and the logistic function. However since the network analysed in this report is binary, the Heaviside step function was used.

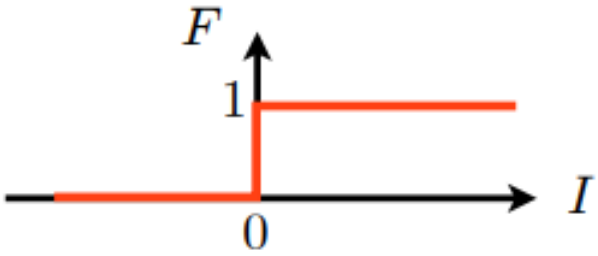

Figure 1: Heaviside step function. Neuron is activated if its field is positive and otherwise remains inactive. This figure is sourced from Mate's module handouts [1].

This function turns on a neuron when its input is positive, and turns it off when it is negative.

It has just been shown that a Lyupanov function can be defined on the Hopfield network i.e. stable states exist, though it must now be shown that memories are stable points in the state-space.

Using the expression in equation 3, it can easily be shown that the following conditions are required to induce a change in the energy function from one time-step to the next:

$$\begin{aligned} H_k(t) > 0 \ ... \ r_k(t) = 0 \\ H_k(t) < 0 \ ... \ r_k(t) = 1 \end{aligned} \tag{4}$$

The conditions above will be important in later proofs. Let us initialise the network at one of the memory states, i.e. $r_k(t) = r_k^{(\mu)}$. Then, re-writing $H_k(t)$ as the sum of contributions from a single chosen memory μ and other memories m, then using equation (1), the following equation is obtained [1]:

$$H_k(t) = \underbrace{\left(r_k^{(\mu)} - \frac{1}{2}\right) \sum_{j \neq k} r_j^{(\mu)} \left(r_j^{(\mu)} - \frac{1}{2}\right)}_{signal} + \underbrace{\sum_{j \neq k} r_j^{(\mu)} \sum_{m \neq \mu} \left(r_j^{(m)} - \frac{1}{2}\right) \left(r_k^{(m)} - \frac{1}{2}\right)}_{noise} \tag{5}$$

Then averaging over all possible values of the bits for the patterns $m \neq \mu$, $\mathbf{r}^{(m)}$:

$$\begin{aligned} \langle H_k(t) \rangle_{\mathbf{r}^{(m)}} &= \left(r_k^{(\mu)} - \frac{1}{2}\right) \underbrace{\sum_{j \neq k} r_j^{(\mu)} \left(r_j^{(\mu)} - \frac{1}{2}\right)}_{K_+ \geq 0} + \sum_{j \neq k} r_j^{(\mu)} (M-1) \underbrace{\left\langle \left(r_j^{(m)} - \frac{1}{2}\right) \right\rangle}_{=0} \underbrace{\left\langle \left(r_k^{(m)} - \frac{1}{2}\right) \right\rangle}_{=0} \\ &= \left(r_k^{(\mu)} - \frac{1}{2}\right) K_+ \end{aligned} \tag{6}$$

Variance:

$$\begin{aligned} \mathrm{Var}[H_k(t)]_{\mathbf{r}^{(m)}} &= \underbrace{\mathrm{Var}[\mathrm{signal}]_{\mathbf{r}^{(m)}}}_{0} + \mathrm{Var}[\mathrm{noise}]_{\mathbf{r}^{(m)}} \\ &= (N-1) \cdot r_j^{(\mu)} \cdot (M-1) \, \mathrm{Var}\left[\left(r_j^{(m)} - \frac{1}{2}\right) \left(r_k^{(m)} - \frac{1}{2}\right)\right] \end{aligned} \tag{7}$$

Now using the fact that

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \tag{8}$$

for some random variable, X, and noting that

$$\mathbb{E}[(r_j^{(m)})^2] = \frac{1}{2} \tag{9}$$

and

$$\begin{aligned}\mathbb{E}[r_j^{(m)} r_k^{(m)}] &= \mathbb{E}[r_j^{(m)}]\,\mathbb{E}[r_k^{(m)}] \\ &= \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}\end{aligned} \tag{10}$$

where equation (10) comes from noting that $r_j^{(m)}$ and $r_k^{(m)}$ are generated independently, it can easily be shown that

$$\text{Var}\left[\left(r_j^{(m)} - \frac{1}{2}\right)\left(r_k^{(m)} - \frac{1}{2}\right)\right] = \frac{1}{16} \tag{11}$$

Therefore equation (7) becomes

$$\text{Var}[H_k(t)]_{\mathbf{r}^{(m)}} = \frac{(N-1)(M-1)}{16}\cdot r_j^{(\mu)} \tag{12}$$

Note that the field of neuron k, $H_k(t)$, is made up of the sum of many stochastic terms. Therefore, by the central limit theorem, its distribution can be approximated as Gaussian, with mean and variance given in equations (6) and (12). However these equations give the mean and variance of neuron k's field in terms of the states of neurons other than k in the same pattern μ i.e. these equations are written in terms of $r_j^{(\mu)}$ for $j \neq k$. It would be nice if the parameters of the distribution over k's field only depended on $r_k^{(\mu)}$ and not $r_j^{(\mu)}$ for $j \neq k$. Thus, we average these parameters over $r_j^{(\mu)}$:

$$\begin{aligned}\left\langle \langle H_k(t)\rangle_{\mathbf{r}^{(m)}}\right\rangle_{r_j^{(\mu)}} &= \underbrace{\left(r_k^{(\mu)} - \frac{1}{2}\right)}_{\pm 1/2}\langle K_+\rangle_{r_j^{(\mu)}} \\ &= (-1)^{r_k+1}\cdot\frac{N-1}{8}\end{aligned} \tag{13}$$

$$\begin{aligned}\left\langle \text{Var}[H_k(t)]_{\mathbf{r}^{(m)}}\right\rangle_{r_j^{(\mu)}} &= \frac{(N-1)(M-1)}{16}\cdot\underbrace{\left\langle r_j^{(\mu)}\right\rangle}_{1/2} \\ &= \frac{(N-1)(M-1)}{32}\end{aligned} \tag{14}$$

where equation (13) is obtained by noting that $\langle K_+\rangle = (N-1)(\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot 0) = \frac{N-1}{4}$.

Thus the distribution over H_k given r_k is given by

$$p(H_k|r_k) \approx N\left(H_k; (-1)^{r_k+1} \cdot \frac{N-1}{8}, \frac{(N-1)(M-1)}{32}\right) \tag{15}$$

Using condition (4) the probability of error in bit k in memory m is given as

$$\begin{aligned} \Pr(\text{error}) &= \frac{1}{2} \cdot [p(H_k < 0|r_k = 1) + p(H_k > 0|r_k = 0)] \\ &= p(H_k < 0|r_k = 1) \\ &= \int_{-\infty}^{0} N\left(u; \frac{N-1}{8}, \frac{(N-1)(M-1)}{32}\right) \mathrm{du} \end{aligned} \tag{16}$$

where the second line in equation (16) comes from noting the symmetry between the distributions for the $r_k = 0$ case and the $r_k = 1$ case.

The analytical predictions in error probability for increasing number of stored patterns were thus computed using a Python script and plotted in the graph below:

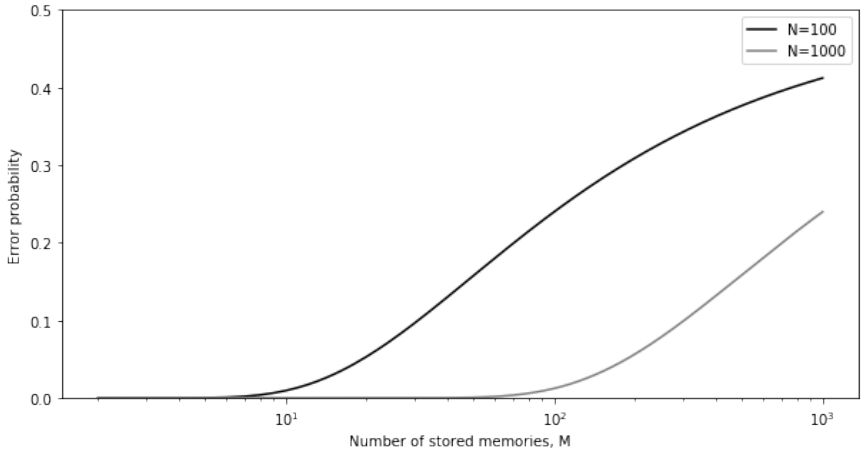

Figure 2: Analytical predictions of probability of recalling a bit indirectly for 100 and 1000 neuron Hopfield networks. It can be seen that the predicted error probability increases as the number of stored patterns increases. Note that the horizontal scale is logarithmic. In each plot there is a threshold below which the error probability is negligible and after which it begins to rise rapidly at roughly an inverse exponential rate (linear in the graph) i.e. exponential of the form $(A - Be^{-CM})$ where A, B and C are arbitrary constants. Both curves approach an error probability of 0.5, as eventually the network would be saturated with errors so that there is an equal likelihood that a memory bit is recalled correctly or not i.e. it is completely random. The error probability is larger for the smaller network (N=100) as an error in any one bit is an error in a larger proportion of bits in the network. From the graph it may be seen that the capacities of the 100 and 1000 neuron networks are about 10 and 100 stored patterns respectively i.e. about 10% of the number of neurons in the network.

1.2 Simulated Error Predictions

In this sub-section, a Python script was used to simulate the behaviour of a 100-neuron Hopfield network. The bit representation of each pattern was drawn from a uniform Bernoulli distribution. After updating the network over a single time-step, changes in any of the one hundred bits were noted. Each experiment was repeated multiple times and the average error probability for each value of M was plotted in the graph shown in figure 3.

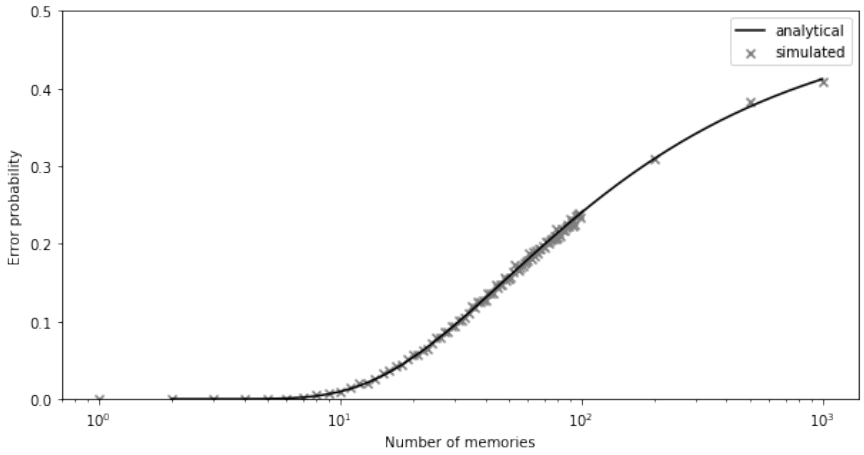

Figure 3: Error probability predictions from a simulated 100-neuron Hopfield network. It can be seen that the simulated predictions closely match the analytical predictions.

1.3 Simulated Error Predictions from Noisy Initialisation

In this part, a similar approach as the previous sub-section was used, except the network was initialised with a noisy version of one of the stored patterns. This was done by flipping the bits of one of the stored patterns randomly according to a Bernoulli distribution with some probability p. These experiments were repeated with p taking on the values: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 1.0. The result of these experiments are shown in figure 4 below.

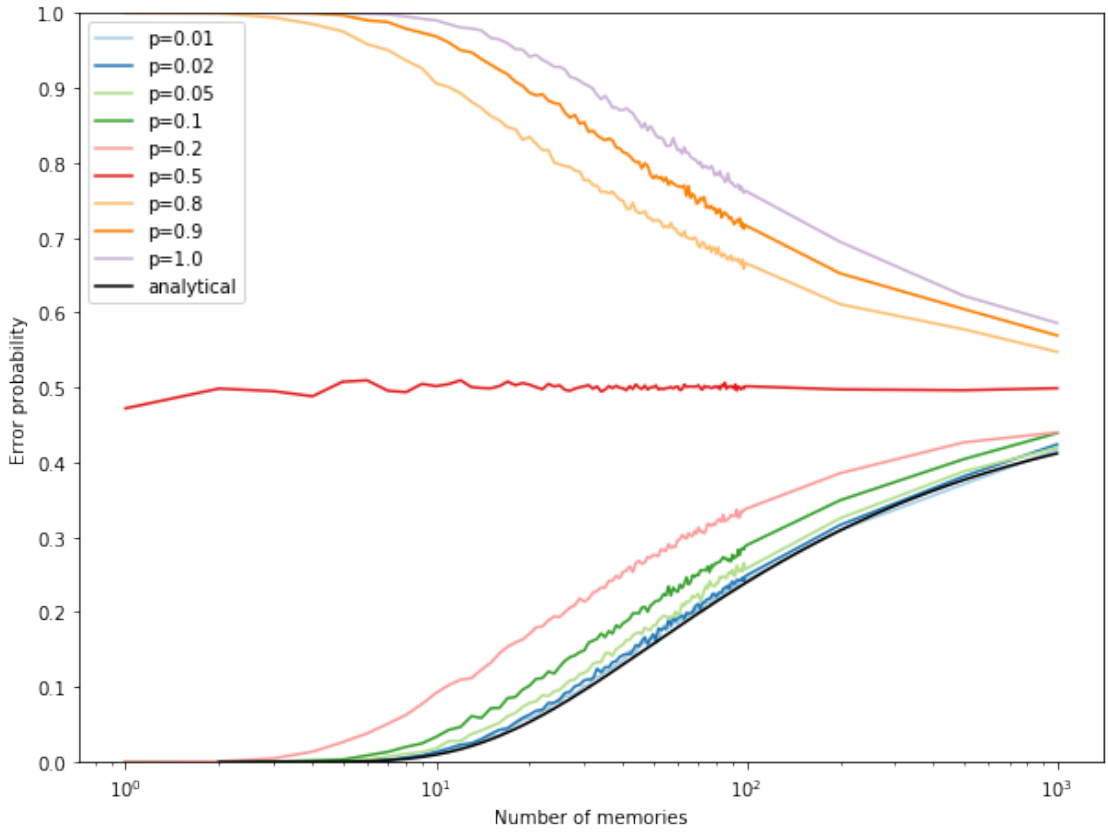

Figure 4: Error probability vs number of stored memories.

Why are the results of figure 4 different from the previous exercises. Recall that the state-space representation for a neuron, k, in pattern, μ, is given as r_k^μ. In figure 4 each bit is flipped according to a Bernoulli distribution with parameter p_{flip}. Therefore there is a probability of $1 - p_{flip}$ that $r_k(t) == r_k^\mu$ and a probability of p_{flip} that $r_k(t)$ is the converse. If it assumed that the network undergoes very little in a small time-step, then it is expected that the each bit will retain the same value at the next time-step. Note that this assumption is valid for a small number of stored patterns in the network. It can be seen that for p ¡ 0.5, the error probability increases as the number of stored patterns increases. When $p = 0.5$ it is found that the error probability is 0.5. This is expected, as assuming the network undergoes very little change in a small time-step, if a single bit is either flipped or not flipped with equal probability, then using the assumption above, the probability that is in the right state or not at time $t + \Delta t$ is also equal. It can be seen that for $p > 0.5$ the error probability is a reflection of those below the 0.5 mark on the vertical axis.

For small values of p (little noise), it can be seen that the simulated error predictions are similar to the analytical prediction. Though as p is increases, so does the error probability. Clearly, the noisier the initialised state, the further it is from the true memory state, and the less likely it converges on the true memory state in one time-step, hence the higher error probability.

For $p > 0.5$ it is seen that the error probability decreases from 1.0 to a limit of 0.5 as

the number of stored patterns increases. Meanwhile for $p = 1.0$ the bits are flipped with certainty. Therefore using the assumption above it is expected that the error probability is near-1 for a low number of stored patterns (which the assumption is valid for).

2 Exploration of a physiological model of a spiking neuron

The classic Hodgkin-Huxley model of the action potential is described by the following equations:

$$\begin{aligned} \dot{v} &= -g_{N_a} m^3 h(v - e_{N_a}) - g_K n^4 (v - e_K) - g_L (v - e_L) + I_{ext} \\ \dot{m} &= \alpha_m(v)(1 - m) - \beta_m(v) m \\ \dot{h} &= \alpha_h(v)(1 - h) - \beta_h(v) h \\ \dot{n} &= \alpha_n(v)(1 - n) - \beta_m(v) n \end{aligned} \tag{17}$$

The parameters used to simulate this model are given in table 1. The model was simulated using the Python excerpt in listing 1. The maximal conductances and reverse potentials were given in the coursework handout, then reasonable values were chosen for the model initialisers. These reasonable values were taken based on guidance from [3]. These values are expected to be reasonable for a few reasons:

- v_0: No current has been injected into the model yet so the initial membrane potential should be the resting membrane potential = -65 mV. This equilibrium membrane potential is the electric force that balances the force induced by the ionic concentration gradient across the membrane [2].
- m_0: The sodium channel activation, which is dependent on v, increases with depolarization as the positive feedback effect between m and v is the mechanism that triggers an action potential [3]. Therefore, it is expected to be near 0 at resting potential values.
- h_0: The sodium channel inactivation, which is dependent on v, decreases with depolarization as it is the mechanism for action potential reset. Therefore, since it decreases with depolarization, it is expected to be significantly non-zero at resting potential. It is chosen to be 0.6 just as in figure 5.11 of source [3].
- n_0: The potassium channel activation, which is dependent on v, increases with depolarization. However, activation leads to an eflux of ions (as opposed to an influx in item 1) which restores the membrane potential to its resting value after an action potential. Therefore it is expected that n is initially moderately low.

	Maximal conductances		Reverse potentials		Model initialisers
g_{Na}	120 μS/nF	e_{Na}	50 mV	m_0	0.08
g_K	36 μS/nF	e_K	-77 mV	h_0	0.6
g_L	0.3 μS/nF	e_L	-54.4 mV	n_0	0.33
				v_0	-65 mV

Table 1: Parameter values and model initialisers. The model initialisers were sourced from reasonable values given in figure 5.11 in source [3].

```python
def integrate_v(v0, m0, h0, n0, I_ext):
    # Note that I_ext is a vector, representing the current injected at each time-step
    v = v0
    m = m0
    h = h0
    n = n0
    vs = [v0]
    for I in I_ext:
        vdot = -g_Na * m**3 * h * (v - e_Na) - g_K * n**4 * (v - e_K) - \
            g_L * (v - e_L) + I
        mdot = alpha_m(v) * (1-m) - beta_m(v) * m
        hdot = alpha_h(v) * (1-h) - beta_h(v) * h
        ndot = alpha_n(v) * (1-n) - beta_n(v) * n
        v += vdot * dt
        m += mdot * dt
        h += hdot * dt
        n += ndot * dt
        vs.append(v)
    return vs
```

Listing 1: Python excerpt for forward Euler integration of Hodgkin-Huxley equations

2.1 Aside: How action potentials are generated

α is the voltage-dependent rate constant at which a gate opens
[2] β is the voltage-dependent rate constant at which a gate closes [2]

In the extracellular medium there is a high concentration of Na_+ and Cl_- ions, as a result from ingested table salt. Inside the cell (in the cytoplasm) there is a higher concentration of K_+ ions.

It is the positive feedback loop in m and v generates an action potential. As the membrane potential rises to a threshold of about 55 V, an increase in m which causes an influx of Na_+ ions i.e. a rapid current influx. This causes a sharp rise in membrane potential which further causes a rise in m and so on. This depolarization causes a decline in h though which causes the conductance of the sodium channels to drop and therefore cause a decline in the current influx and therefore a drop in membrane potential. Note that the depolarization also causes a rise in the eflux of K_+ ions which also acts to decrease the membrane potential and bring it closer to rest values.

2.2 Simulating system response to steady injected current

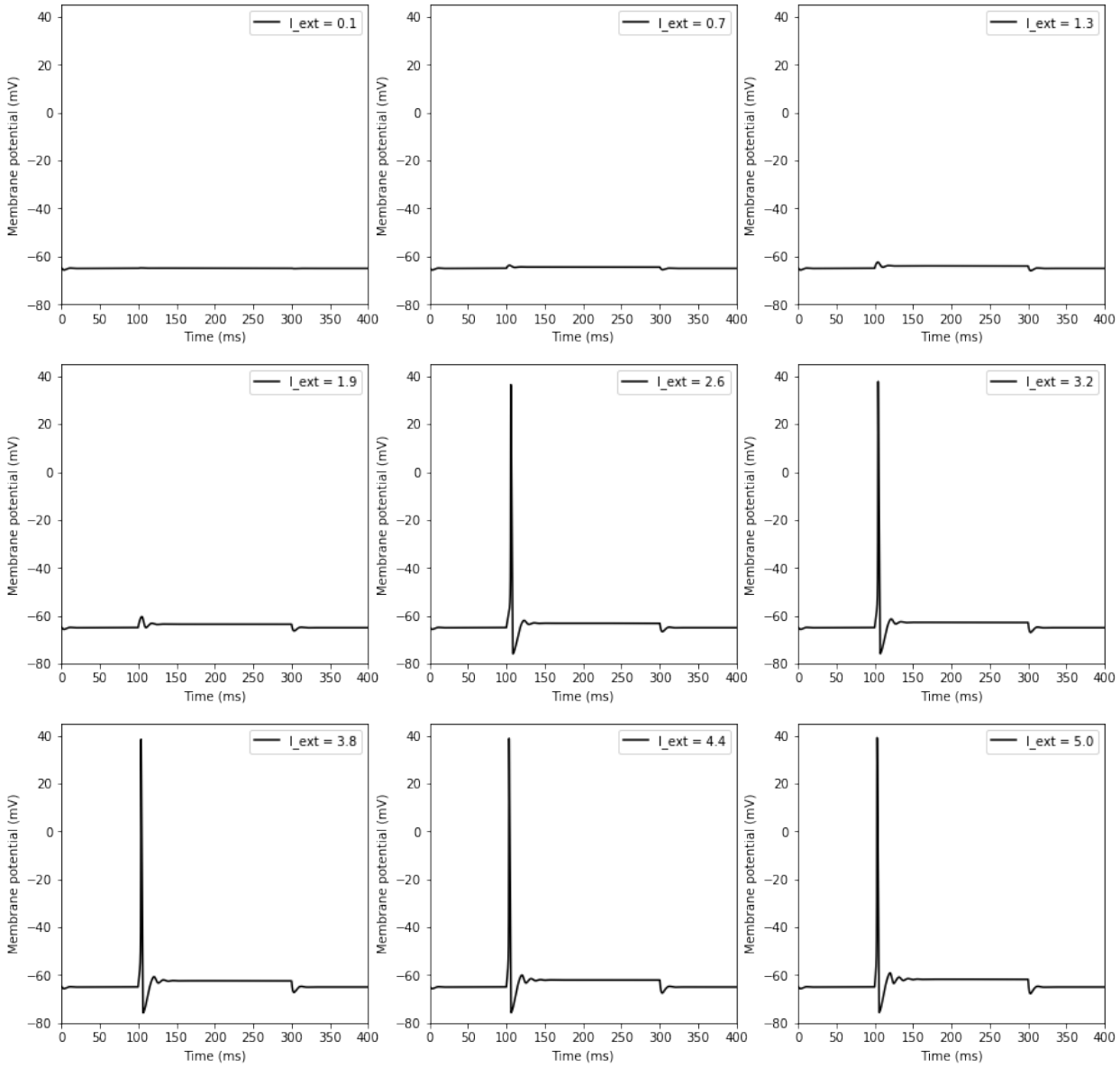

Figure 5: Plot of membrane potential against time for injection currents (I_ext) in the range 0.1-5mA/nF. Current injection begins at 100ms and ends at 300ms.

In figure 5 above it can be seen that action potentials begin to fire after a current in the range 1.9 to 2.6 mA/nF is injected. Figure 6 below models the behaviour for currents in this range for a more precise estimate of the injection current required to trigger action potentials.

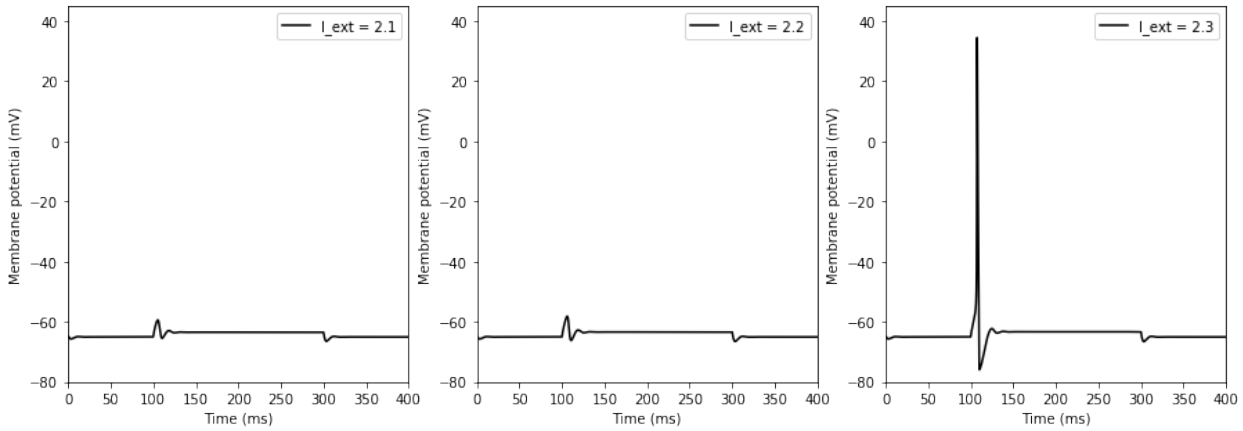

Figure 6: Plot of membrane potential against time for injection currents (I_ext) in the range 2.1-2.3mA/nF. Current injection begins at 100ms and ends at 300ms. It can be seen that action potentials begin to fire at about I_ext = 2.3mA/nF.

This simulation was repeated for 1 to 30 mA/nF. For currents above 6 mA/nF, it was noted that multiple action potentials fired instead of just one. This characteristic 'fast-spiking' behaviour is seen in figure 7 where the spacing between successive action potentials is uniform over a long duration. Many neurons exhibit 'adaptation', a phenomenon where the spacing between successive action potentials increases slowly over time. Though this behaviour is not noted in figure 7 as adaptation occurs too quickly to be observed, hence the term 'fast-spiking'. A plot of action potential frequency against current is given in figure 8.

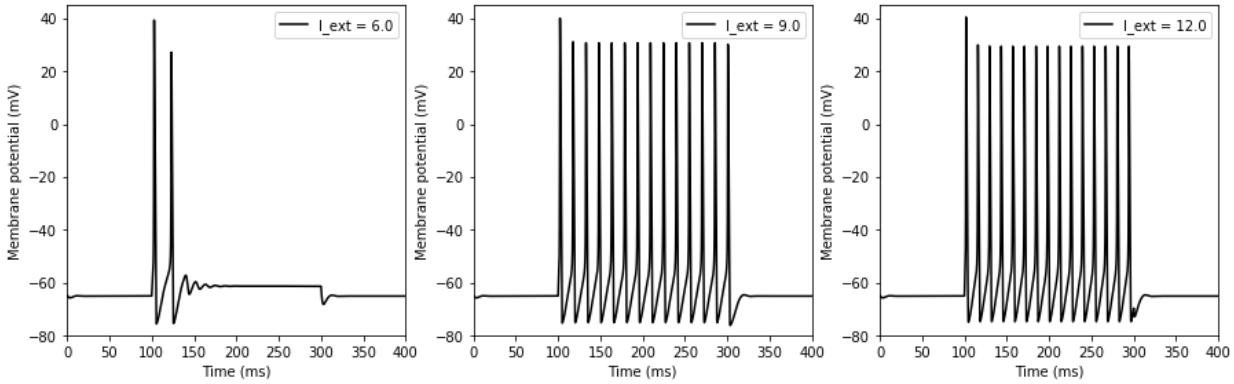

Figure 7: Fast-spiking behaviour of neurons exhibited for steady current injections above 6 mA/nF.

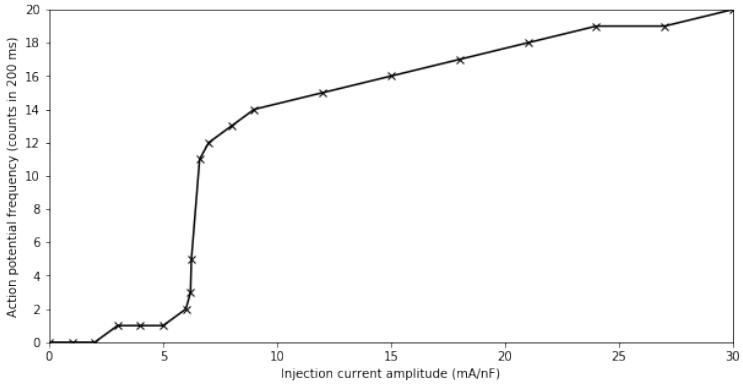

Figure 8: Plot of action potential frequency as a function of injection current amplitude. It can be seen that the input threshold for firing is just below 3mA/nF (actually it was seen to be 2.3 mA/nF in figure 6. Notably, the spike frequency begins to rise dramatically at a current of 6mA/nF. At about 6.8 to 7mA/nF the steepness of this rise in action potential frequency starts to decline.

In figure 6 it can be seen that the threshold for inducing firing is 2.3 mA/nF. Though notably the action potential frequency begins to increase dramatically at I = 6 mA/nF as seen in figure 8. Then the rate of increase in action potential frequency then begins to decrease about I = 6.8 to 7 mA/nF.

2.3 Simulating system response to periodic square pulse current

In this section, we model the response of an individual neuron to an alternating current. The properties of this peridioc current inputs are described in figure 9. Figure 10 shows the response to varying periods in the current. Listing 2 gives the Python excerpt used to generate these periodic currents. Neuronal responses can typically be grouped into a few common classes, some of which are mentioned below [4]:

- Regularly-firing neurons: These neurons display adaptation. That is, the spacing between successive action potentials decreases slowly, over several spikes, to a steady value.
- Fast-spiking neurons: These neurons do not display adaptation. That is, the spacing between successive action potentials is fixed.
- Post-inhibitory rebound: The release of an inhibitory current triggers an action potential.
- Bursting/Stuttering: Successions of action potential firing are observed followed by 'silent' periods. This process repeats periodically (bursting) or aperiodically (stuttering).

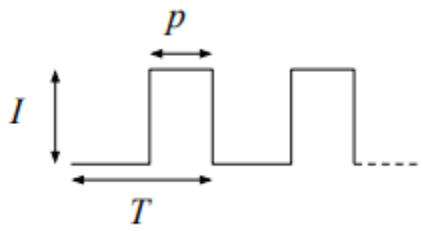

Figure 9: Current input waveform. T: wave period; p: pulse width. This figure has been sourced from the coursework handout [5].

```python
def periodic_square_pulse(amp, p, T, dur, dt=0.001e-3):
    """
    Generate periodic square pulse with a pulse-width, p, and a period, T.
    The input starts at amplitude 0 and rises to magnitude I during a
   pulse.
    The waveform runs for the duration, dur, specified.
    dt gives the discrete time increment
    """
    on = False
    wave = []

    t = np.arange(0, T+2*dt, dt)
    i = 0
    for time in np.arange(0,dur,dt):
        if(t[i] >= T):
            i = 0
        wave.append(amp if t[i] >= T-p else 0)
        i += 1
    return wave
```

Listing 2: Python excerpt to generate periodic square current pulse

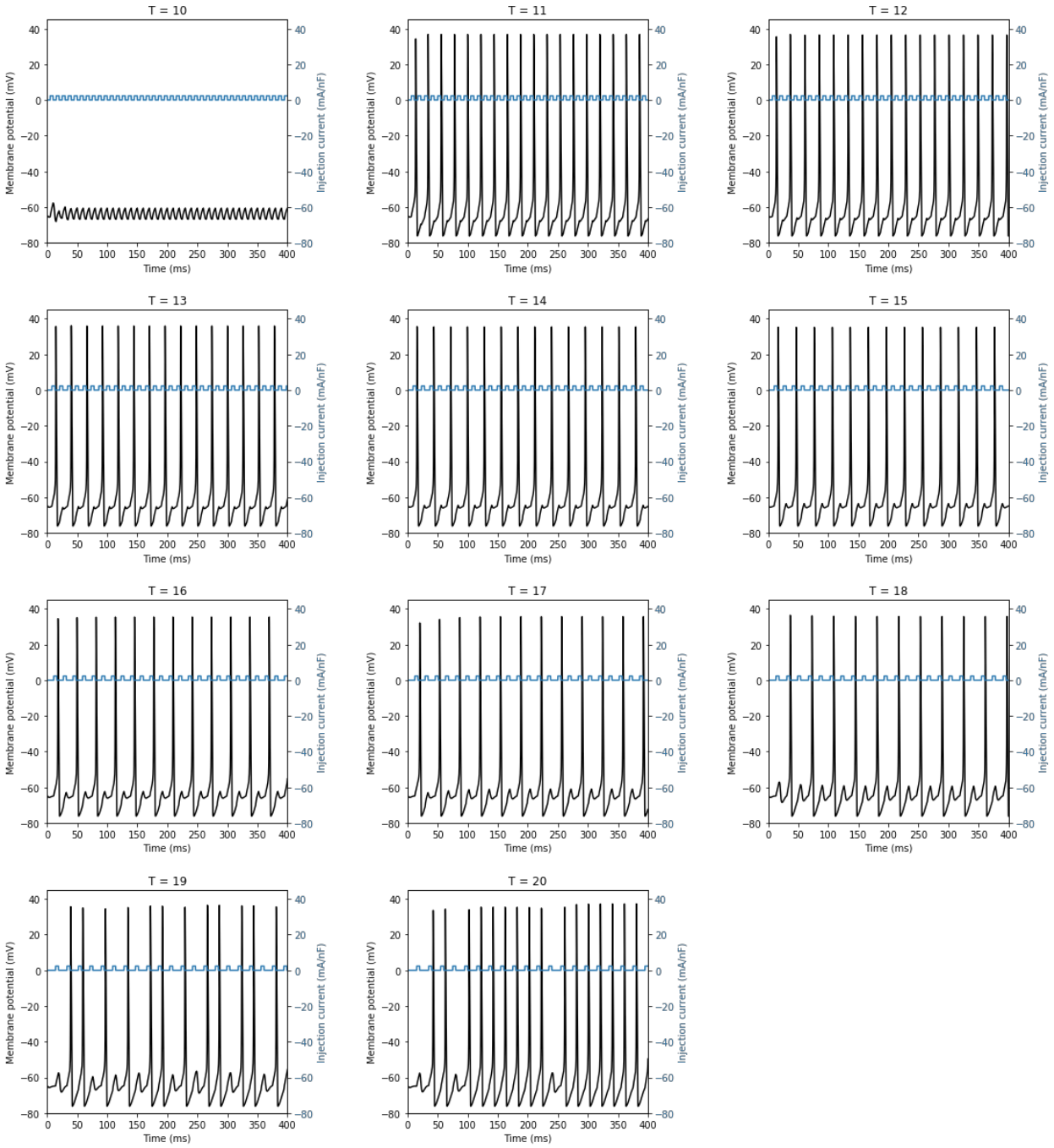

Figure 10: Response of membrane potential to periodic square pulse input current with 2.3 mA/nF amplitude and 5ms pulse width. The wave period is altered from one subplot to the next. It appears that no action potentials fire when current-high time is equal to current low-time (T = 10ms). The simulation was run for several cycles to allow any transients resulting from the initialisation to die down. Most subplots show a regular spike rate though cases T = 19 and T = 20 appears to show some more irregular behavior.

In figure 10, though some irregularity is noted for T=19 and T=20, regular 'fast-spiking'

can be seen to be observed in the other cases.

As mentioned at the start of this subsection, firing may also be observed when an inhibitory current is removed from a neuron. This 'post-inhibitory rebound' is observed in figure 11. In this figure, a simulation was run wit negative periodic square pulses (I = -5mA/nF).

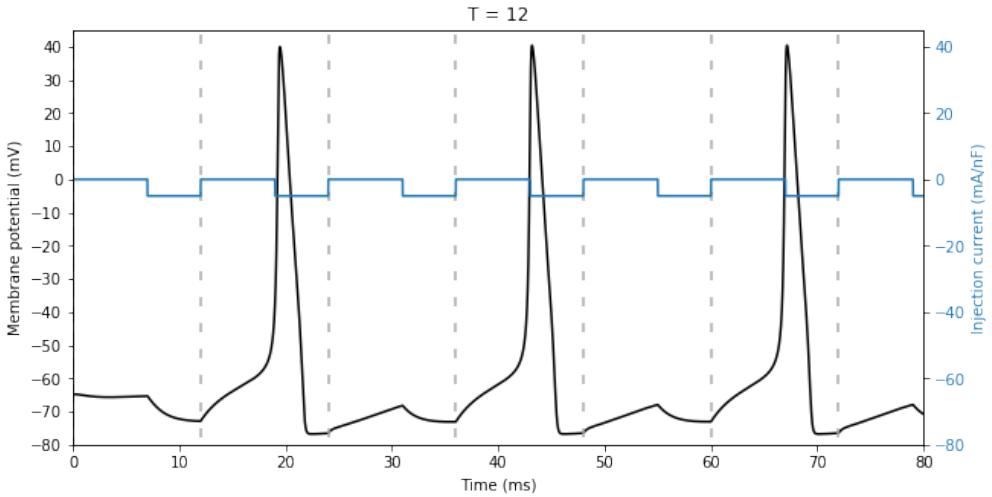

Figure 11: Firing is still observed for negative pulses: I = -5mA/nF, T = 12 ms. Notice that sharp rises in membrane potential are initiated each time the inhibitory current is released (dashed lines). It is the release of the inhibitory current that triggers an action potential. This behaviour is known as 'post-inhibitory rebound'. This phenomenon is not captured in basic LIF or rate models as explained in the next section.

Note that these current injections act as a means to observe the behaviour of a neuron when stimulated with electrodes in a lab. In the brain, a postsynaptic neuron would receive a spike train input from a presynaptic neuron as opposed to a peridic square wave or steady current injection. These experiments do allow interesting neuronal behaviour to be observed though. Additionally, they also reveal shortcomings in some models of neuronal behaviour as discussed in the following sections.

2.4 Limitations of LIF and rate models

Reviewing the list of neuronal response classes at the start of the previous section, it can be seen that the LIF model has a few limitations.

- In figure 10, note that action potentials fire for T=11 though not for T=10, even though the amplitude of the current injected is the same. This behaviour cannot be explained by an 'integrate-and-fire' model. Note that in the T=11 case, the first spike is generated at the end of the first current pulse. If it was true that the postsynaptic neuron simply integrated the current injected into it then the same firing would be expected to be observed at the end of the first current pulse for the T=10 case too, though this is not the case. A neuron in the Hodgkin-Huxley model does not therefore

exhibit 'integrate-and-fire' behaviour though it exhibits 'resonate-and-fire' behaviour. This behaviour is not only seen in this model, but it seen in reality across species.

- It resets the membrane potential immediately after an action potential is fired, therefore adaptation cannot be modelled since it depends on the accumulation of the effect of previous states of the membrane potential. The LIF model may be improved by using a 'filter' for refractoriness with a time constant much slower than that of the membrane potential decay. See reference [4] for a more in-depth explanation.
- In figure 11, action potential firing is observed when a negative pulse is applied. This 'post-inhibitory-rebound' cannot be modelled by an integrate-and-fire neuron as a neuron only fires under this model when the membrane potential rises to a positive threshold. A negative current would only act to drive the membrane potential down in the LIF model, which would therefore not elicit an action potential. In other words, only excitation triggers action potentials in integrate-and-fire models.

References

1. https://www.vle.cam.ac.uk/pluginfile.php/11820271/mod_folder/content/0/Lengyel/ printable/06_assocmem_print2_15.pdf?forcedownload=1

2. Math.pitt.edu, 2020. [Online]. Available: http://www.math.pitt.edu/ bdoiron/assets/ermentrout-and-terman-ch-1.pdf. [Accessed: 19- Mar- 2020].

3. Dayan, P. and Abbott, L., 2005. Theoretical Neuroscience. Cambridge: MIT Press.

4. W. Gerstner,"1.4 Limitations of the Leaky Integrate-and-Fire Model — Neuronal Dynamics online book", Neuronaldynamics.epfl.ch, 2020. [Online]. Available: https://neuronal dynamics.epfl.ch/online/Ch1.S4.html. [Accessed: 20- Mar- 2020].

5. https://www.vle.cam.ac.uk/pluginfile.php/11935851/mod_folder/content/0/cw02.pdf?force download=1