Lecture 8.2 Exercises

8.2.1 Hodgkin-Huxley (HH) lamprey segmental oscillator model (Andrej Bicanski)

These exercises should familiarize the student with the following concepts. A mechanistic model of spike frequency adaptation via intracellular calcium and calcium dependent potassium channels. The use of HH models to simulate pharmacological effects (one of the benefits of using as complex a model as the present one). Slow and fast oscillatory subsystems. Intrinsic versus network mediated oscillations, i.e. tonically spiking neurons vs. pacemaker neurons (here dependent on the mode of stimulation, AMPA vs NMDA), or escape vs. release from inhibition as network mechanisms for oscillations.

1. Write a short Matlab script which calls the neuron model (m-file callnetwork.m) repeatedly, simulating a single neuron, each time with an increased input current.
    a. Measure the inter-spike-interval between the first and second, the second and third and between the last two spikes of a 2 second spike train. Plot the frequency-current (FI) curves similar to Figure 8.6D of chapter 8.

    b. Reduce the strength of the after-hyperpolarization current by increasing value of the 5HT variable (simulating the effect of serotonin on the network) and observe the changes to the FI curves.

    c. Pick one input current and plot the calcium concentration ([Ca]_[AHP}) over time, with and without a moderate 5HT level. How do they compare and why? Consider why the calcium concentration over time differs despite the fact that in both cases the inflow and decay rates for the intracellular calcium are the same.

    d. D: Gently change the values of the variables rhoAP and deltaAP, controlling calcium inflow and decay (m-file HHneuron_Wallen1992.m) and observe the effect on the FI curves and the calcium concentration. Compare your observations with C.

2. The NMDA based calcium subsystem in the model acts on a slower time scale as compared to the calcium which flows into the cell with each action potential (see exercise 1). Test the capability of the model to reproduce NMDA induced oscillations.
    a. Run the HH model (m-file callnetwork.m) in single neuron mode, driven by an NMDA bath strength of 1.0, 2.0 and 3.0 units. Repeat the simulations with a TTX block of action potential generation, simulating the action of a pharmacological agent.

    b. Gently change the values of the variables rhoNMDA and deltaNMDA controlling calcium inflow and decay (m-file HHneuron_Wallen1992.m) and observe the effect on NMDA induced oscillations (with and without TTX blockade).

    c. Why do the oscillations have a higher frequency in one of the two cases (TTX vs. no TTX)? Consider all factors which contribute to the hyperpolarization at the end of

burst/plateau. Note that the fact that the two calcium pools (action potential related (exercise 1) and NMDA receptor related (exercise 2)) are modeled separately relies on the notion that their sources (the respective calcium channels) are far apart on the cell membrane. If this is not the case the slow and fast calcium systems would interact through a joint calcium pool in addition to their indirect interaction through the effect of separate calcium pools on membrane potential dynamics.

3. Test the effects of AMPA and NMDA baths on the whole network. Examine how network oscillations can be generated with release or escape from inhibition.
    a. Run the HH model (m-file callnetwork.m) in network mode at three AMPA bath concentrations (2.0, 3.5, 5.0). Record and plot the membrane potential for all neurons in the network and compare the frequency of the network oscillation.

    b. Remove the contribution of LIN neurons (e.g. by setting their connection weights to zero) and observe their contribution to burst termination at strong AMPA stimulation levels (>4.0).

    c. Run the HH model (m-file callnetwork.m) in network mode at an AMPA bath concentration of 3.0 units with 3 different levels of 5HT (0, 0.25, 0.5) and observe the neuromodulatory effect on network frequency.

    d. Run the HH model (m-file callnetwork.m) in network mode at three NMDA bath concentrations (1, 2, 3). Record and plot the membrane potential for all neurons in the network and compare the frequency of the network oscillation.

    e. Sever the link between left and right hemi-segments of the segmental network by setting the contralateral inhibitory synaptic connection weight to zero, and run the model in network mode, once with an AMPA bath strength of 3.5 units, once with a NMDA bath strength of 2 units. What is the main qualitative difference and why?

8.2.2 Oscillator-based model of the salamander CPG (Jeremie Knuesel)

1. Running the CPG model in Matlab

    Here we will see how to use the provided code for an oscillator-based model of the salamander CPG. The Matlab function in 'salam_cpg_osc.m' implements the model itself. Simulation results can be rendered in a plot similar to Fig. 2 of the paper (Ijspeert et al. 2007) using the Matlab function in 'plot_salam_cpg.m'. The inputs and outputs of these two Matlab functions are described at the beginning of the respective files.

a. Pick a CPG drive strength in the range corresponding to the swimming behavior. (The limb oscillators saturate for drive levels above 3, while body oscillators saturate for drive levels above 5. Swimming occurs when only the limb oscillators saturate.)

b. Using this drive strength, run a 20 seconds simulation of the CPG activity with a timestep of 0.01 seconds. Record the output of the CPG oscillators, along with their instantaneous frequencies.

c. Plot the outputs of the body and limb CPG oscillators, together with the oscillators' instantaneous frequencies.

d. Repeat steps a,b,c using a drive strength corresponding to the walking behavior. (Walking occurs when neither the limb nor body oscillators saturate.). Observe the differences to the swimming case.

e. Repeat steps a,b,c using a non-constant, linearly increasing drive strength covering both the walking and swimming behaviors. Observe the walking-swimming transition on the plot.

Solution:

a. We can pick e.g. a drive strength of 3.7
b. times = 0:0.01:20;

   [theta, r, x, dtheta] = salam_cpg_osc(times, 3.7);

c. plot_salam_cpg(times, x, dtheta, 3.7);
d. We can pick a drive between 1 and 3, for example 2.2:

   [theta, r, x, dtheta] = salam_cpg_osc(times, 2.2);

   plot_salam_cpg(times, x, dtheta, 2.2);

   Compared to the swimming case, the output of the limb oscillators is not flat, and the frequency of oscillations is lower. The synchronization pattern of the body oscillators has changed from a travelling wave to a standing wave.

e. Let's generate a ramp of drive strengths between 0 and 6, sampling with as many values as we have in the 'times' vector:

   drives = linspace(0, 6, length(times));

   Then run the model with these drives and plot the result:

   [theta, r, x, dtheta] = salam_cpg_osc(times, drives);

   plot_salam_cpg(times, x, dtheta, drives);

2. Issues with numerical modeling

   The outputs of the CPG model are calculated by numerical integration of the model's differential equations. The Matlab code in 'salam_cpg_osc.m' uses a simple Euler integration. The accuracy of the result increases as the integration timestep decreases (but the computations take more time and more memory). It is important to know if a simulation result is due to the properties of the model, or if it is an artefact due to the method of numerical integration which is always inexact. A simulation run with a large timestep can produce invalid results. To demonstrate this, repeat exercise 1e using a timestep of 0.05, and observe the consequences of using an excessively large integration timestep.

   Solution:

   ```
   times = 0:0.05:20;

   drives = linspace(0, 6, length(times));

   [theta, r, x, dtheta] = salam_cpg_osc(times, drives);

   plot_salam_cpg(times, x, dtheta, drives);
   ```

   Some oscillations are still visible but with a lot of noise. One might conclude that the model generates noisy outputs, but the noise has nothing to do with the model, it is an artefact due to integaration errors caused by a bad choice of timestep.

3. Initial conditions, unstable fixed points and robustness of convergence

   Any simulation based on differential equations requires a choice of initial conditions, i.e. the initial state of the model. The Matlab code in 'salam_cpg_osc.m' picks random initial conditions everytime the function is called. Ideally our CPG model would quickly converge to the expected oscillation pattern, regardless of the initial conditions. In practice it is possible to find particular values that leave the model stuck in a different oscillation pattern. This occurs when the model finds itself in the state of an unstable fixed point: the phase lags between oscillators remain constant instead of converging to the expected phase lags. This can happen due to the initial conditions, but also following a perturbation of the state variables. These unstable fixed points are properties of the model, and as described in section 1.2 of the supplementary materials of the article (Ijspeert et al. 2007), we can remediate this by introducing some randomness in the integration step.

   a. Repeat exercise 1e, running the model and plotting the results several times with the same parameters. Observe how each time, the oscillations start at different phases, but the network always converges to the same rhythm: on the left side of the plot (walking behavior), the trunk and tail oscillators (blue and green respectively) oscillate in antiphase.

   b. Edit the file 'salam_cpg_osc.m'. Find where the initial values are defined to random values (the initial values are held in vector variables named 'theta0', 'r0' and 'r_dot0'). Change the

code to initialize all values to zero instead. Run again the simulation and observe the new oscillation pattern: the trunk and tail oscillators are now in phase. The left forelimb and left hind limb (oscillators 17 and 19) are also now in phase.

c. In the file 'salam_cpg_osc.m', find where the phase derivatives are calculated at every integration step (they are stored in the variable 'dtheta', calculated in the 'get_derivatives' function). Add a small random term to the calculation of every element 'dtheta(i)' (e.g. a random value uniformly distributed between -0.01 and 0.01). Run the model again and plot the results. Observe the oscillation pattern during the walking behavior: the oscillators are again in antiphase. The model is now more robust to bad initial conditions or perturbations.

d. Undo the changes made in (b) to restore the random initial conditions.

4. Playing with the model parameters

The oscillation pattern produced by the model and the stability of the rhythm depend on the model parameters, in particular the weights and phase biases of the couplings and the intrinsic frequencies of the oscillators. Changing the values of these parameters and observing the results can give a better sense of what the parameters represent.

Different values for the model parameters can also make numerical integration more difficult, requiring an adaptation of the integration timestep.

Below are some suggestions of changes you can make to the model parameters. Feel free to experiment with other parameters or values!

a. Edit the file 'salam_cpg_osc.m' and find where the coupling weights are defined. Change the coupling weights from limb to body oscillators (variable 'w_limb_axis') to a much larger value and repeat exercise 1e. With a value large enough (e.g. 300 instead of 30), the numerical integration produces noise instead of oscillations, as seen in exercise 2. Try adapting the integration timestep to produce valid results when using such large coupling weights.

b. Change again the coupling weights from limb to body oscillators, this time using a lower value (e.g. 10, the same as for the other couplings). Repeat exercise 1e and observe how the oscillation pattern has changed: the couplings are now too weak to impose a standing wave on the trunk oscillators during the walking behavior (the trunk oscillators are no longer in phase, instead they produce a travelling wave, similar to the wave produced during the swimming behavior).

c. Leave the value of 'w_limb_axis' at 10. In 'salam_cpg_osc.m', find where the saturation function is called to calculate the intrinsic frequencies of the body oscillators based on the

drive strengths. Replace the value for 'c_nu_0' (originally 0.3) to something higher, such as 2. This will increase the intrinsic frequencies of the body oscillators, as they will now start at 2 Hz instead of 0.3 Hz for low drive strengths. Repeat exercise 1e and observe the effect of the limb-body couplings on the trunk oscillators. The limb oscillators have their original, low intrinsic frequencies. The body oscillators however now have much larger intrinsic frequencies. With such a difference between the limb and body oscillators, the weakened limb-body couplings fail to impose the slow limb frequency on the trunk. However, they do perturbate the trunk oscillators: their outputs are now aperiodic and unlike sines. When you are done, revert the coupling weights and saturation parameters to their original values.

d. Try changing the phase biases and see how it affects the oscillation pattern.