Comp3402 Neural Oscillators and CPGs

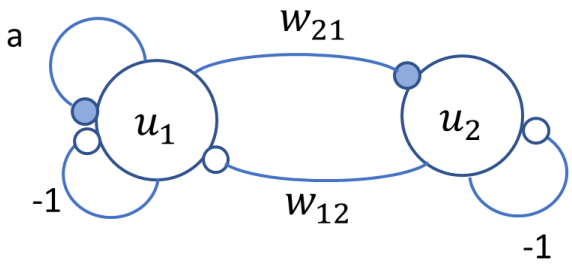
C.B.Price January 2021

Purpose	(i) To investigate a coupled leaky-integrator oscillator and discover why this is doomed to fail (ii) To investigate a non-linear 'limit cycle' oscillator which is a good model for biological oscillators, (iii) to investigate the phase-delay neural circuit
Files Required	Webots and Octave assets.
ILO Contribution	3
Send to Me	
Assignment Info.	
Homework	Read Chapter 7

Activities

1 A coupled leaky-integrator circuit.

Here's the diagram of the coupled leaky integrator circuit we shall investigate.



> Remember empty synapses are inhibitory (negative) and filled synapses are excitatory. The inhibitory 'synapses' labelled -1 are modelling the leak in the bucket. Here the weight from neuron 1 to neuron 2, $w_{21}$is a positive number and the weight from 2 to 1 is negative, i.e., $w_{21} < 0$

As explain in class, and in Chapter 7, theory tells us that this circuit will only oscillate if (i) *a = 2* and (ii) $w_{12}w_{21} < -1$.

(a) Download and unzip the Octave assets to your desktop. Now run the script **Solve_Two_Leakys** and when prompted input the following parameters **a = 2,** $\boldsymbol{w_{12} = -4}$**,** $\boldsymbol{w_{21} = 5}$. You will see the value of u_1 oscillating and also the trajectory in the u_1-u_2 phase plane.

If you want to see the trajectory dynamically then you can run the script **Solve_Two_Leakys_New**

(b) Let's break the condition **a = 2**. Make two further runs, one with a just a little less than 2 and the second with a just a little more than 2. What do you observe? Can you describe how the values of u_1 change with time? If u_1 were the length of a car suspension spring, what would this mean about the car?

The second condition that the product of the weights should be < -1 is *easily* satisfied, since it says nothing about the exact numerical values.

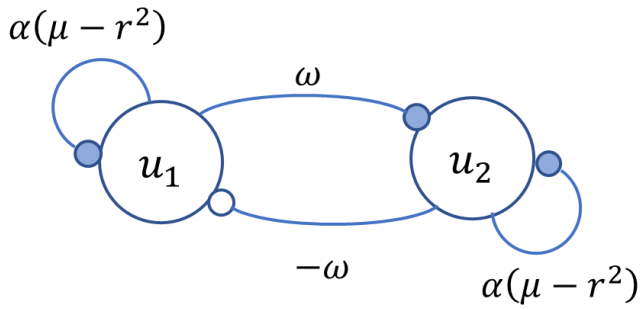
(b) Investigate other values of w_{12} and w_{21} bearing in mind it's the product that's important. What is the effect on the oscillation as the product increases?

(c) Now just for fun, break the second condition. Keep a = 2 but enter values to make the product of w_{12} and w_{21} greater than -1, perhaps even positive.

The point to take away from this is that two coupled leaky integrators can be made to oscillate, but this is certainly not how biology builds its oscillators. We shall now move on to a real biological neural oscillator model.

2 The Hopf Oscillator

Here's the neural circuit for the Hopf oscillator. Details of how this circuit is related to actual neurophysiology is presented in Chapter 6. Note the resemblance with the coupled leaky integrator model, the neurons influence each other in the same 'direction'



Here, the parameters α, μ, ω are all fixed in time, and *r* is a variable given by $r = \sqrt{u_1^2 + u_2^2}$.
The parameter ω is the angular velocity of oscillation in rads/sec and *r* is the radius of the circle in the phase plane

(a) Run the script **Solve_Hopf** or **Solve_Hopf_New** (to get a dynamic solution) with the following baseline parameters

$$\alpha = 5, \qquad \mu = 1, \qquad \omega = 3.1415$$

You should find the phase trajectory approaches a circular limit-cycle of radius *r = 1* and that the period of oscillations is $T = 2\pi/\omega$ which here evaluates to 2 seconds. Sweet!

(b) Investigate the effect of changing α keeping the other parameters unchanged. Make it smaller as well as larger. What happens to the u1-time plot and to the phase plane trajectory? State in Simple English what the parameter alpha does.

(c) Return alpha to its default value. Now investigate the effects of changing μ. HINT: set it to squared values, such as 4, 9, 16, 25. Explain in Simple English what mu actually does.

(d) I hinted above that the parameter ω is related to the period of oscillation *T* through the expression $T = 2\pi/\omega$. Use this to get an expression for ω as a function of period *T.* Then choose some values for *T*, calculate the corresponding omegas and plug these into the model. Check you do get oscillations of period *T*.

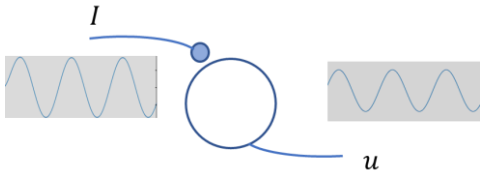
(e) Investigate changing the initial conditions of the oscillator. I suggest you change neuron-1 initial condition. The line of code where these are set are clearly indicated.

You should have found that this oscillator will always oscillate for its parameter values, and unlike the coupled leaky integrator oscillator does not require exact parameter values. It works, biologically.

3 The Phase-Lag neuron.

This is nothing new. It is based on our single leaky-integrator neuron, but instead of using a pulse input we use a sine-wave input. Here's the ODE for the neuron and the idea shown as a sketch

$$\frac{du}{dt} = \frac{1}{\tau}(-u + kI)$$

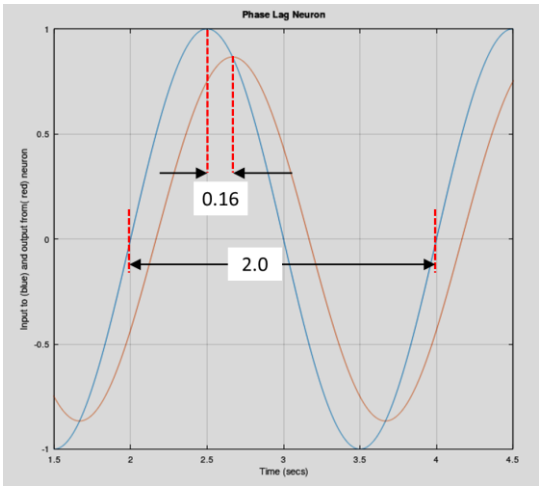


The amount of phase lag is specified by setting a value for tau (or a given oscillator omega) and *k* is fixed so the amplitude of the output is the same as the input. See chapter 7 for details.

(a) Run the script **Solve_Phase_Lag_1** which inputs a sine wave of amplitude 1 to the neuron. You can choose the phase lag (but this must be less than 90 degs). So not select amplitude correction. Look at the output and check that the neuron gives you the correct phase lag. Here's how to calculate the phase lag from the plots, I chose 30 degs.

First, find the time for a complete period (360 degrees). Here this is 2.0 seconds. Then find the time of the lagged peak. I measured 0.16 seconds. The phase lag is calculated by simple proportion:

lag = 360*(0.16/2.0) = around 30 degrees



(b) Now re-run the script for the same desired phase lag and select corrected amplitude. You should get a phase-lagged sine wave with no amplitude change.

(c) Now investigate other phase lags (up to the limit of 90 degrees). Perhaps try 90 degrees and look at the values of tau and k which you get. Try 100 degrees …

(d) Roll back to requesting a 60 degree phase lag. Look carefully at the neuron output. You will see it overshoots on the first cycle or so. Any idea why?

4 Three Cascaded Phase-Lag neurons

To get larger phase differences, we can cascade three neurons, each providing a part of our desired phase lag. So, if we need a phase lag of 180 degrees, we can achieve this with a cascade of 3 neurons each providing a 60 degree lag.

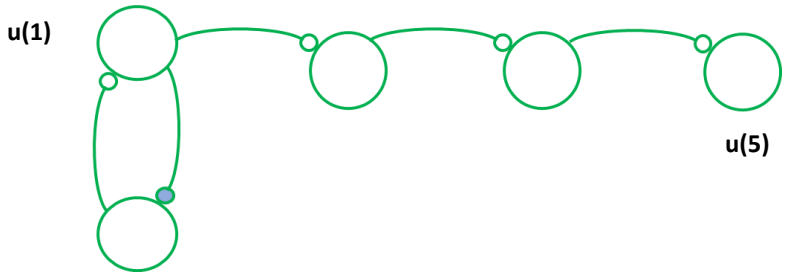
(a) Run the script **Solve_Phase_Lag_3.** Figure 1 shows you the input and all three cascade outputs, Figure 2 shows you the input and the output of the whole cascade. Run for a phase difference of 60 degrees

(b) Now run for 90, 120 and 180 degrees. Observe how each successive neuron initially overshoots its desired phase lag.

(c) Repeat and note down in each case the 'transient time', the time needed for the final output to settle to an amplitude of 1.0

5 Hopf Oscillator coupled with Cascade Phase-Lag neurons

This is in preparation for our Hexapod robot program. It's putting all of the above together. Here's the Hopf oscillator followed by a chain of three neurons, each providing 1/3 of the desired phase delay



Run the script **Solve_Hopf_with_PhaseDelay_New** which plot out the motion of neurons u(1) and u(5) on the phase plane.

Important phase lags to try out are 60, 90, and 180 degrees. You should observe some initial transient (beginning time) where the trajectory is far from the limit cycle, but is attracted to it after some time.