# Comp3402 More Neural Circuits

**C.B.Price January 2021**

| | |
|---|---|
| **Purpose** | (i) To investigate some basic neural circuits (ii) To learn how they can perform fundamental maths operations. |
| **Files Required** | Octave folders on website. |
| **ILO Contribution** | 3 |
| **Send to Me** | |
| **Assignment Info.** | |
| **Homework** | Read Chapter6 |

## Activities

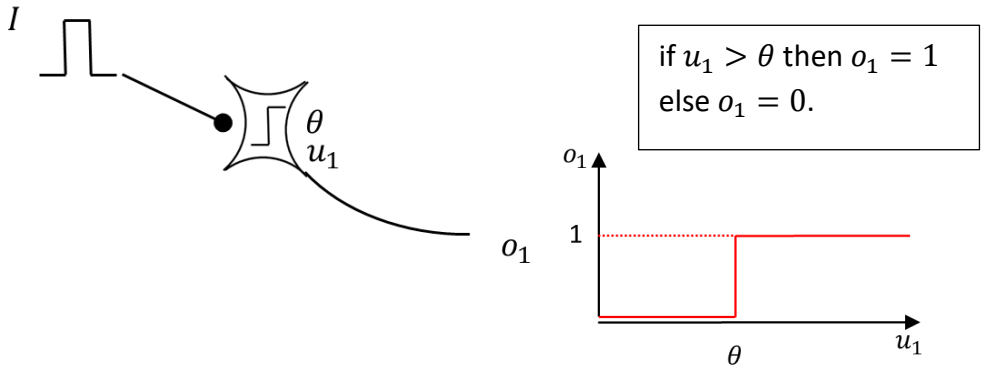
**Configuring Octave**

Depending on your installation you may need to load a couple of packages like this. You will find out when you run a script and it shouts at you.

**pkg load signal**
**pkg load matgeom**

### 1 Investigating a Single Leaky Integrator Neuron with Step Output Threshold.

Here's the neural circuit with constant input *I* and neuron state $u_1$. This state is then passed through the output step function shown below. So, the neuron output is either 0 or 1.




Here's the ODE for this circuit:

$$\frac{du_1}{dt} = \frac{1}{\tau}(-u_1 + I)$$

$$o_1 = g(u_1, \theta)$$

Here, the function $g(u_1, \theta)$ is the threshold function which takes input $u_1$ and produces output $o_1$ shown above.

(a) Type **SingleThresholdTest** in the Octave Command Window. The input pulse applied to the neuron starts at 2 secs and ends at 4 secs. The pulse has a value (height) of 0.8. When asked, set the threshold to 0.5.

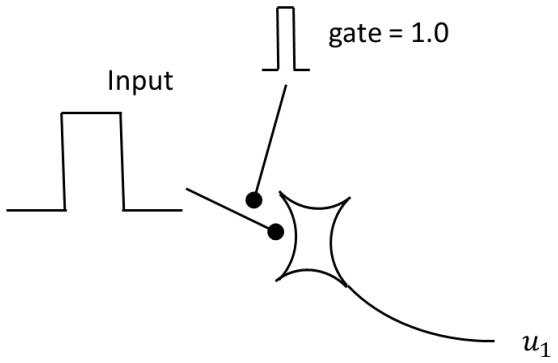
You should see the input pulse to the neuron, the neuron's state variable $u_1$ and its output $o_1$. Look at the output pulse. The neuron's state rises to the input value 0.8, but the neuron output shoots up to 1.0. So we've gone from *analog* to *digital.*

(b) Try to find the time when the output goes from 0 to 1. Can you see how this is related to the threshold? Hint: look at the middle graph.

(c) Now experiment with the threshold value. What happens for values (i) close to 0.0 and (ii) just less than the input pulse value (8.0)? If we are to use this circuit in an engineering situation, then what threshold would you choose?

## 2 Investigating the "Gating" circuit

Here's a neural circuit which receives a long input pulse which is input into the neuron $u_1$, but it passes through a "gate" which only lets it through when the gate pulse is 1.0. So the input gets through when the input pulse and the gate pulse "overlap", at this time the gate is "open". Note that the gate pulse is relatively short.




This diagram shows that the input dendrite is multiplied by the gating dendrite which achieves the desired behaviour. The ODE for this neuron is

$$\frac{du_1}{dt} = \frac{1}{\tau}(-u_1 + gI)$$

where *I* is the input and *g* the gate. While the input can have any value, the gate is either 0 or 1, since it is executing a ***logical*** function. The Octave code that does this is

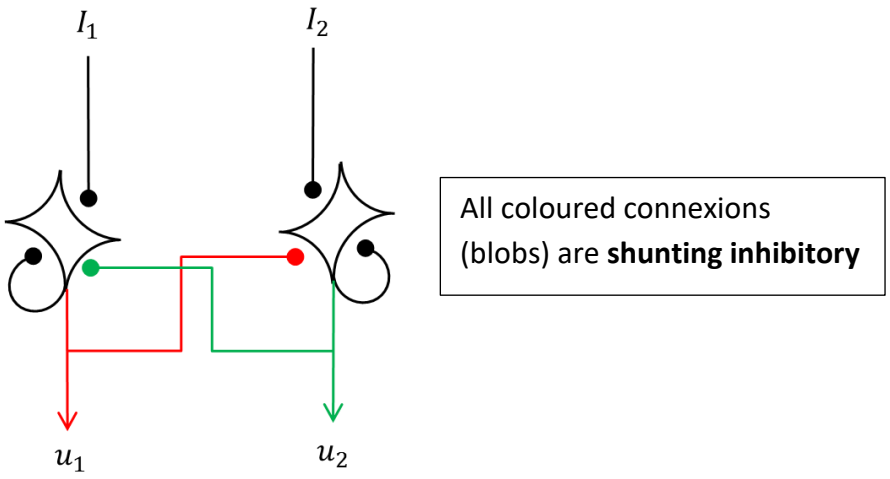
**dudt(1) = ( -u(1) + inputPulse*gatePulse)/tau1;**

where you can see the multiplication of the input pulse and the gate pulse

(a) Run the script **Gating** where you will be asked to choose the height of the input pulse. The gating pulse height is set to 1.0. Check that the gate lets the input in only when the gate is 1.0 and that $u_1$ reaches the input value only when the gate is open.

(b) How could you create a gating pulse from a neuron whose output is not 0 or 1?

## 3 Shunting Inhibition for 2 Neurons




Here the left neuron sends an inhibitory signal to the right neuron (red line) and the right neuron sends an inhibitory signal to the left neuron (green line). In addition each neuron both excites and inhibits itself (black line).

Here's the ODE for this circuit:

$$\frac{du_1}{dt} = \frac{1}{\tau}[u_1 - u_1 u_1 - k u_1 u_2 + u_1 I_1 + \epsilon]$$

$$\frac{du_2}{dt} = \frac{1}{\tau}[u_2 - u_2 u_2 - k u_2 u_1 + u_2 I_2 + \epsilon]$$

Note first that the equations are symmetrical. The second is the same as the first with $u_1$ and $u_2$ swapped.

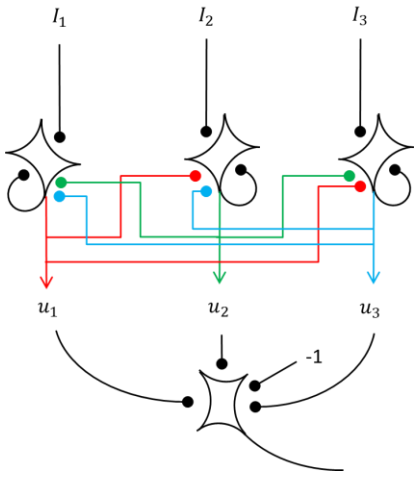
Theory (see Chapter 6) predicts that, due to the shunting inhibition, the neuron with the smallest input will decay to zero, while the neuron with the largest input will rise to the "input value plus one"

(a) Run the script **ShuntingTwo**. You will be asked to input a value for each neuron's input. Look how the circuit selects the larger input. The max values of both neurons are printed to the console. Try a few values, some very different (3.5, 8) and some close (2.0, 2.001).

(b) There will be one combination of inputs where the winner takes all behaviour does not work. I wonder what this is? Theory predicts the output of both neurons in this case is given by the expression $u_1 = (I + 1)/3$ . Make a quick check!

## 4 Finding the Maximum of values in an array

Here we shall extend the above circuit in two ways. First, we will expand the input array to length 3. Second, we shall remove the extra '1' added to the calculation. Here's the resulting circuit and system of ODEs




$$\frac{du_1}{dt} = \frac{1}{\tau}[u_1 - u_1u_1 - ku_1u_2 - ku_1u_3 + u_1I_1 + \epsilon]$$

$$\frac{du_2}{dt} = \frac{1}{\tau}[u_2 - u_2u_2 - ku_2u_1 - ku_2u_3 + u_2I_2 + \epsilon]$$

$$\frac{du_3}{dt} = \frac{1}{\tau}[u_3 - u_3u_3 - ku_3u_1 - ku_3u_2 + u_1I_1 + \epsilon]$$

$$\frac{du_4}{dt} = \frac{1}{\tau}[-u_4 + u_1 + u_2 + u_3 - 1]$$

The first three equations use shunting inhibition, the fourth adds the outputs of neurons 1 to 3 (only one is non-zero) and subtracts 1.

(a) Run the code **ShuntingThreeMax.m** which contains this circuit. Try several combinations of input values, and make sure that the circuit outputs the maximum.

(b) You can find the code for the complete circuit in **rhs_ShuntingThreeMax.m**