Comp3402 Hexapod CPGs

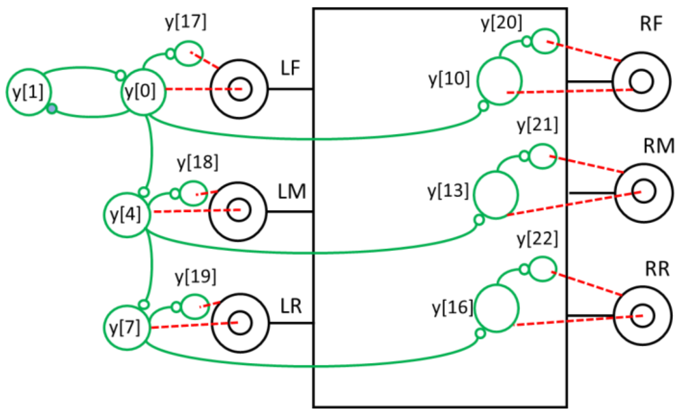
C.B.Price January 2021

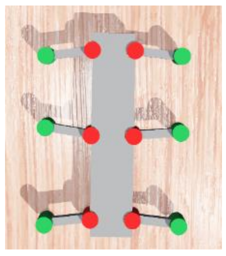
Purpose	(i) To create a neural Central Pattern Generator to drive a hexapod robot (ii) To investigate the principal hexapod gaits produced by this circuit
Files Required	Webots and Octave project folders on website.
ILO Contribution	3
Send to Me	
Assignment Info.	
Homework	Read Chapter 7

Activities

1 The 'Wiring Diagram'

Here's the diagram of the neurons which drive the six leg motors and the six foot motors.





	Leg	Foot
LF	17	0
LM	18	4
LR	19	7
RF	20	10
RM	21	13
RR	22	16

Most of your coding will be in the right-hand-side file **CBP_rhs_hexapod.c.** When you run the compiled code, it will wait 10 seconds before the neurons' outputs are sent to the motors, to let the circuit stabilize. Lots of useful data is written to an Octave file which will automatically make you some nice plots. There will be one file for each robot labelled with its name e.g, name "Hex_1" taken from the scene tree. Here's what's on the plots

Fig1	Combined outputs of neurons 0,4,7 and 10,13,16 (feet)
Fig2	Individual outputs of these neurons
Fig3	Outputs of left side, neurons 0,4,7 (feet) plus 17,18,19 (legs)
Fig4	Outputs of right side, neurons 10,13,16 (feet) plus 20,21,22 (legs)

Coding each neuron is straightforward if you start with the maths. Here's an example, note how the subscripts in math transfer to array indices in code.

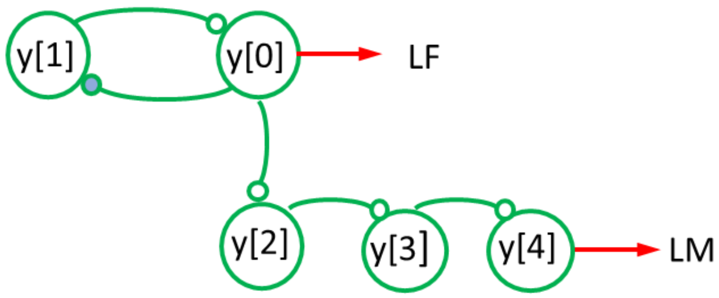
$$\frac{du_2}{dt} = \frac{1}{\tau}(-u_2 + ku_1)$$

```
dudt[2] = (-u[2] + k*u[1])/tau;
```

2 Coding the left feet

(a) The oscillator has been coded, so if you compile and run the code you should see the left foot moving since this is driven by neuron 0. Also, the plots will show this oscillation, everything else will stay zero.

(b) Now code neurons 2,3, and 4 which produce a phase delay into the second foot, driven by neuron 4. You will see the **dudt[i]** for each neuron is set to zero. All you need to do it to replace the zero by some code. Here's the neurons and the maths



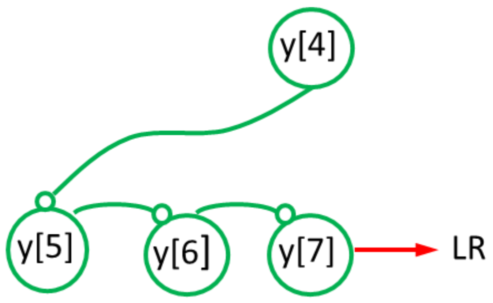
$$\frac{dy_2}{dt} = \frac{1}{\tau}(-y_2 + ky_1)$$

$$\frac{dy_3}{dt} = \frac{1}{\tau}(-y_3 + ky_2)$$

$$\frac{dy_4}{dt} = \frac{1}{\tau}(-y_4 + ky_3)$$

Compile and run and you should see two of the three left feet working.

(c) Now code the remaining left foot like this



$$\frac{dy_5}{dt} = \frac{1}{\tau}(-y_5 + ky_4)$$

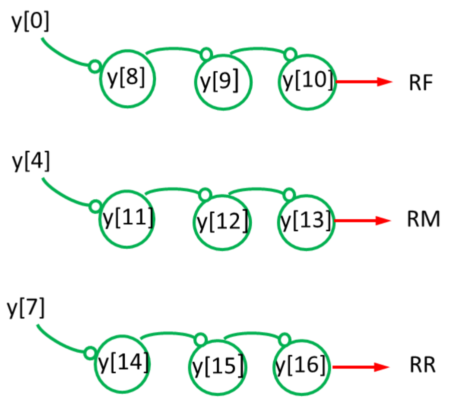
$$\frac{dy_6}{dt} = \frac{1}{\tau}(-y_6 + ky_5)$$

$$\frac{dy_7}{dt} = \frac{1}{\tau}(-y_7 + ky_6)$$

Now you should see all three left feet working.

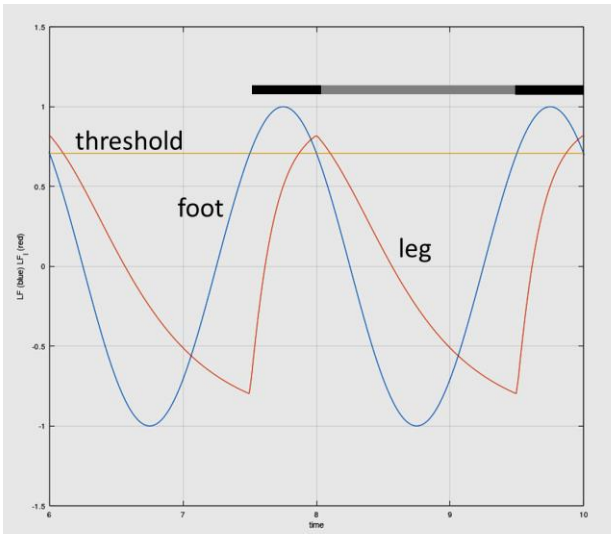
3 Coding the right feet

Coding the right feet. Here we are going to take the outputs of the tree left feet neurons, 0, 4 and 7 and give each a 180 degree phase lag to drive the right feet neurons. The computations for **k1** and **tau1** have been coded (don't use k and tau). We need three neurons to interconnect each pair of left and right foot neurons. Here's what we need to connect each left foot to its opposite right. If all is well there should only be two feet off the ground at any time (one on each side).



4 To generate the leg motion, we need to generate a sawtooth' signal which rises quickly when the foot is off the ground (the swing forward time) then falls slowly while the foot is on the ground. Refer

The blue curve shows the neural signal sent to the left-front (LF) foot. The foot rises when the value is above the threshold (black bar). This is when the leg must swing forwards. The leg signal is the red curve; when the foot lifts it starts to rise from a negative value (rotated backwards) and quickly rises to a positive value (rotated forwards). When the foot drops (start of the grey bar) then the leg slowly moves backwards due to the slowly falling neural signal which becomes negative



to Chapter 7 for the details. Here's the code to generate the neural signal for the left-front leg (LF-l). The other legs have similar code.

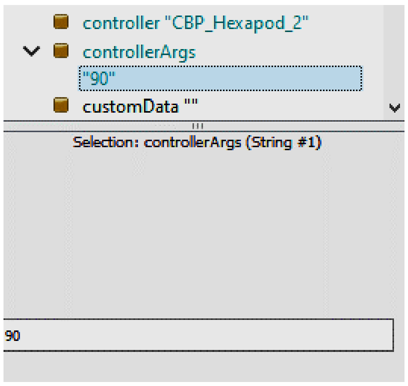
```
if( y[0] > thresh ) drive = 1; else drive = 0;
dydt[17] = (-y[17] + 5.0*drive)/tau1;
```

Make sure you make the if test on the correct leg neuron; here it is **y[0]** since this is the front left leg. Apart from this, the if test is identical for all six leg neurons. Note the input into each neuron is 5.0 in order to give the large rate of rise of the neural signal.

Now code up all the legs and check that your hexapod displays the cored *quadruped* gait.

5 **Investigating the Metachronal, Ripple and Tripod gaits.**

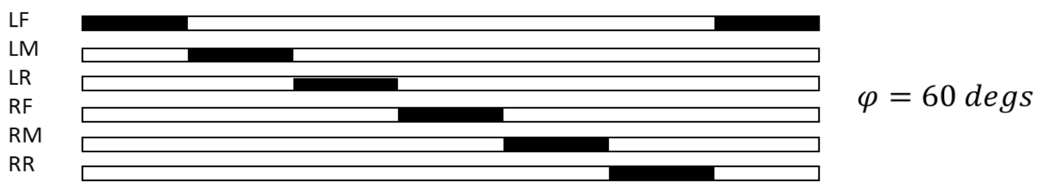
These are the three major gaits shown by hexapods in nature. It is important our neural model is able to replicate these. Each gait corresponds to a particular phase difference between the neurons. This phase can be entered in the scene tree through the **controllerArgs** variable like this.



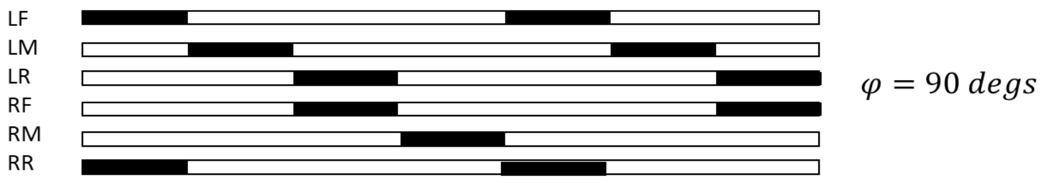
```
controller "CBP_Hexapod_2"
controllerArgs
"90"
customData ""
Selection: controllerArgs (String #1)
90
```

The three major gaits correspond to phase differences of 60, 90 and 180 degrees. The gait patterns corresponding to these angles are shown in the diagram below.

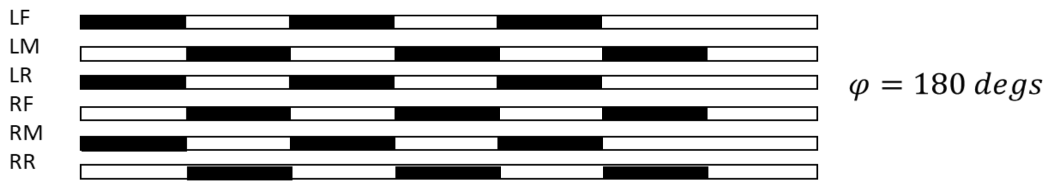
Black bars show when each foot is raised. Time it to the right. Rows are RR, RM, RF, LR, LM, LF



Metachronal (wave)



Tetrapod



Tripod

Investigate each gate in turn. I suggest you take this order: Tripod then Metachronal then Tetrapod.

Webots has been programmed to report the speed of the robot for each gait. You could make a record of this together with your observations.

Gait	**Speed**	**Observations**
Tripod		
Metachronal		
Tetrapod		

6 Making the hexapod move on an arc.

Have a think on a simple modification to the code to get the hexapod moving on an arc instead of a straight line

7 How does the hexapod cope with ramps and bumpy surfaces ?

Investigate this. When you create a bumpy surface in Webots, think carefully about the height of the bumps. Look at the hexapod ground clearance and the reach of its feet.