General thoughts
1. We can get accurate integration with a very discrete model (in this case 8 discrete EPG neurons) without the need for artificial tiling of EB tiles and smoothening functions. The real connectivity also includes neighboring connections, which I have not yet implemented but will likely give more smooth behavior.
2. Bottom-up or top-down; should we attempt to implement the full connectivity matrices straight away and get things working from there, or start by building on top of Herve's model, slowly adding new elements inspired by the EM data? (this is the approach I've taken)
3. We can probably simulate shibire experiments by reducing all output weights of a given type of neurons by a fixed percentage (do we have experimental data on how efficient shibire is?)
4. In the present model, I have included PEG-¿EPG connectivity even though we have no basis for this from the EM data. This connectivity could plausibly be present through either gap junctions or local circuits in the EB, but it could also not exist.

Structure and dynamics

I have used only tile neurons in the EB and contracted each half of the PB to 8 glomeruli that are circular in connectivity space (figure 1). Each tile has only a single explicit EPG neuron, but its connectivity corresponds to that of two EPG neurons projecting to the left and right PB respectively (their input and thus activity would be identical so no reason to model them separately). There are both local recurrent EPG connections (as in the EM data) and connections via PEG neurons from the PB. Global inhibition is achieved via D7 neurons which are innervated by EPG neurons in two glomeruli (one in the left PB, one in the right PB) and inhibit PEG and PEN1 neurons in all other glomeruli. Angular velocity input enter the model via PEN1 input.

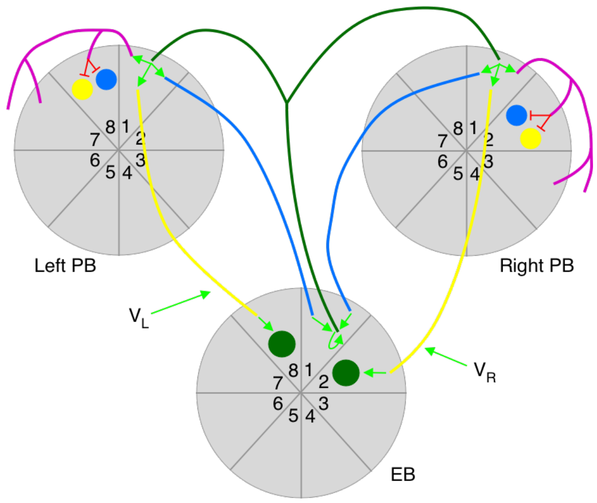


Figure 1: Structure of the firing rate model. Green: EPG, blue: PEG, yellow: PEN1, purple: D7.

I have used a tanh transfer function throughout and we constrain all firing rates to be ≥ 0.

$$f(x) = tanh(x) \tag{1}$$

Activities are then given by:

$$\tau_{EPG}\frac{dEPG}{dt} = -EPG + f(W_{PEG,EPG}\cdot PEG + W_{PEN1_L,EPG}\cdot PEN1_L + W_{PEN1_R,EPG}\cdot PEN1_R + W_{EPG,EPG}\cdot EPG) \tag{2}$$

$$\tau_{PEG}\frac{dPEG}{dt} = -PEG + f(W_{EPG,PEG}\cdot EPG - W_{D7,PEG}\cdot D7) \tag{3}$$

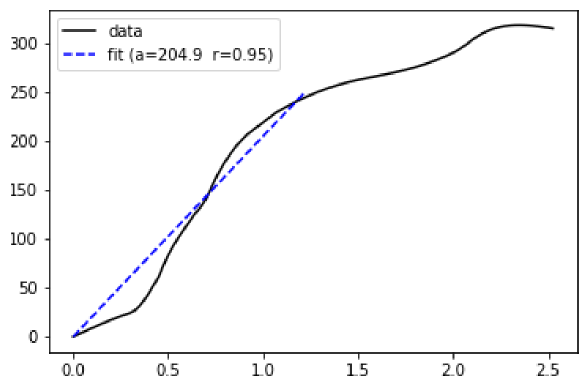
$$\tau_{D7}\frac{dD7}{dt} = -D7 + f(W_{EPG,D7}\cdot EPG) \tag{4}$$

$$\tau_{PEN1}\frac{dPEN1_L}{dt} = -PEN1_L + f(W_{EPG,PEN1_L}\cdot EPG - W_{D7,PEN1_L}\cdot D7 + v_L) \tag{5}$$

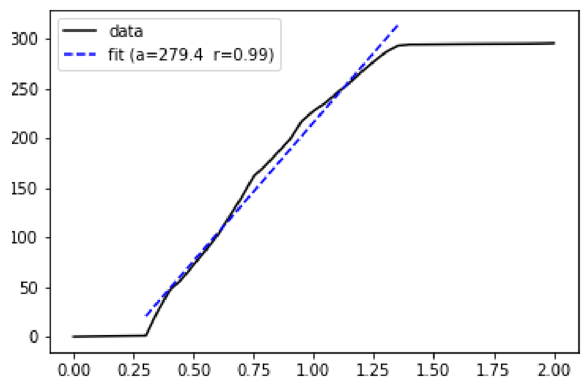
$$\tau_{PEN1}\frac{dPEN1_R}{dt} = -PEN1_R + f(W_{EPG,PEN1_R}\cdot EPG - W_{D7,PEN1_R}\cdot D7 + v_R) \tag{6}$$

Velocity integration

Velocity tuning curves are linear-ish and can be used to translate between real velocity in degrees/s and input velocity.



(a) equilibrate for 2000 ms, measure mean veocity over 500 ms → early velocity.



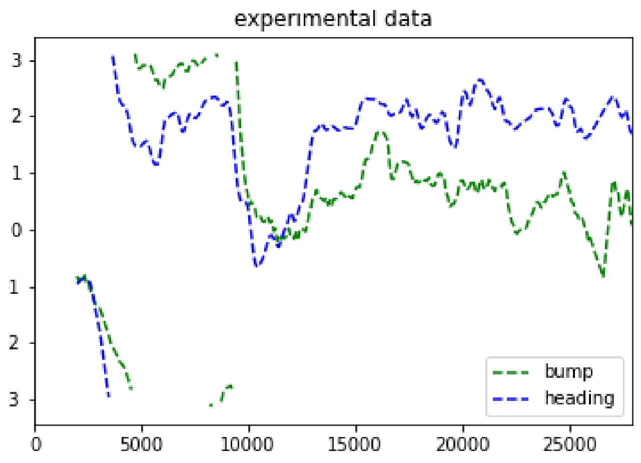
(b) For small inputs, turning is not sustained. If we equilibrate for 2000ms and measure rotational velocity over 10000 ms, this threshold behavior is more evident.

Figure 2: x-axis is input velocity (v_R), y axis is bump angular velocity (degrees/s)

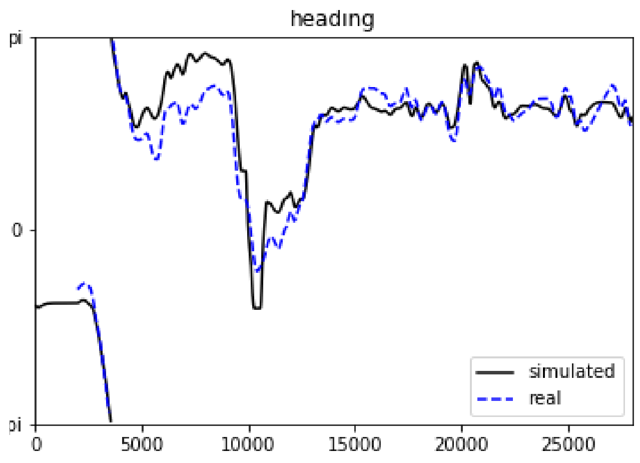
Turns are generally short and slower than 200 degrees/s in the experimental data, and I therefore use a conversion of 200 based on figure 2a to go from experimental velocity to input velocity in equations 5 and 6.

Tracking experimental data

We can simulate experimental data reasonably well even with this simple model.



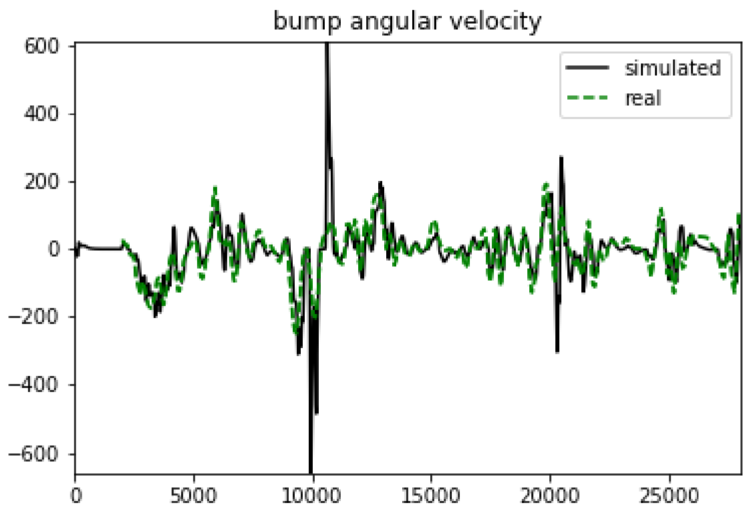
(a) real heading and bump position



(b) real heading and simulated bump position

Figure 3

The heading curves in figure 3b match reasonably well, as do the velocity curves in figure 4a.

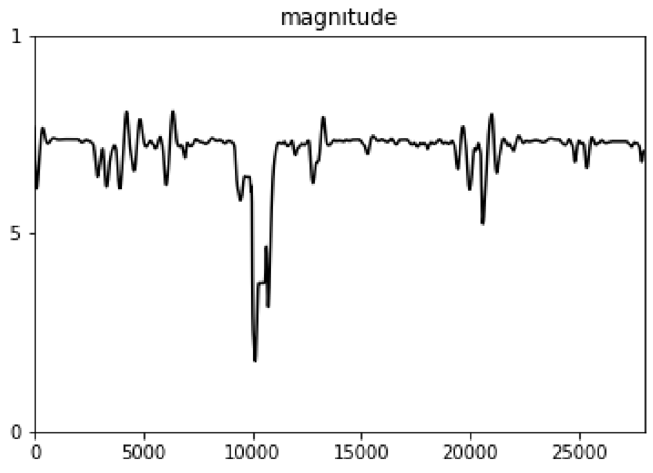


(a) real and simulated velocity curves

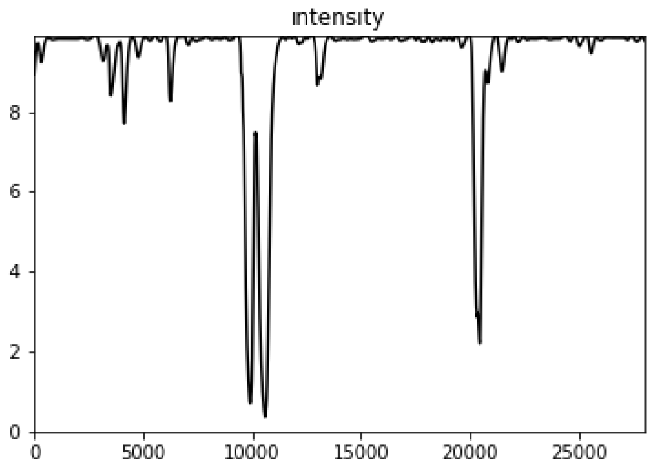
Figure 4

We see that our integrator does not quite follow the long rapid turns at 5,000ms and 10,000ms; but neither does the real bump (figure 3a) which actually does a worse job recovering from the turn.

The poor performance can result from very rapid turns leading to strong global PEN1 and thus EPG excitation, which in turn mediates D7 excitation and global inhibition. Thus broad excitation leads to downstream global inhibition and quenching of the bump (figure 5). This is different from the real system where the bump intensity tends to correlate positively with rotational velocity...
Local EPG - EPG recurrence in the EB somewhat mitigates this by retaining activity during strong inhibition of PEN1 and PEG.



(a) Magnitude of PVA average. This decreases strongly following global inhibition near the rapid turn at 10,000 ms



(b) Intensity of the most active EPG neuron (bump height). This decreases strongly following global inhibition near the rapid turn at 10,000 ms.

Figure 5

Shibire simulations

On the basis of the above model we can simulate 'shibire inhibition' by inhibiting the outputs of a given neuron class. This is exemplified in figure 6 where progressively stronger D7 inhibtion from 0% to 70% decreases the PVA magnitude of the EPG bump.

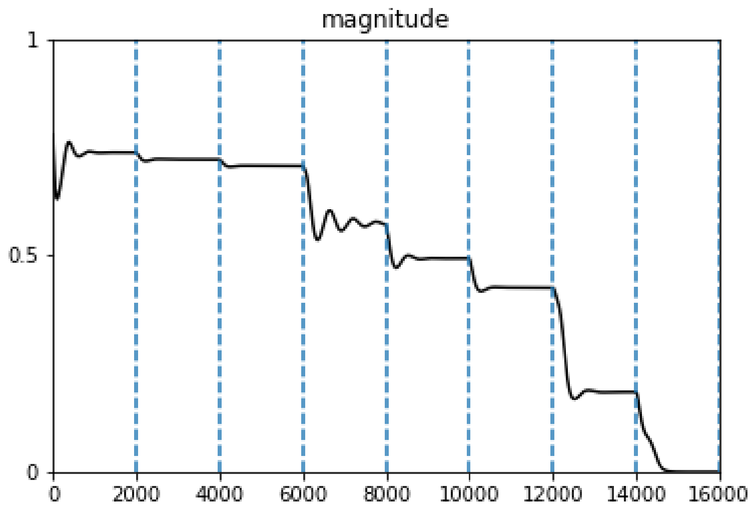
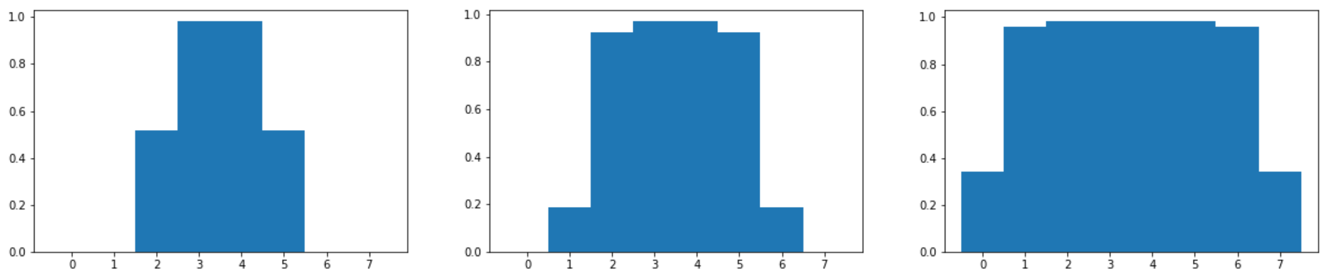


Figure 6: magnitude of PVA average for sequential shibire inhition (0, 10%, 20%, ... , 70%).

We can plot the distribution of EPG acitvity towards the end of each epoch to see how this changes (figure 7), and note that while the maximum bump intensity remains constant, decreased lateral inhibition leads to a broadening of the bump (what does the experimental data look like?).



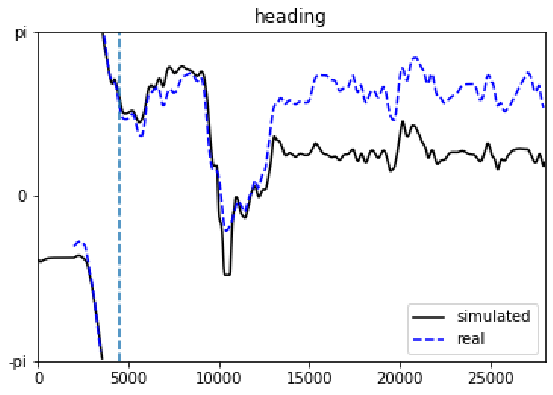
(a) No inhibition

(b) 30% D7 inhibition

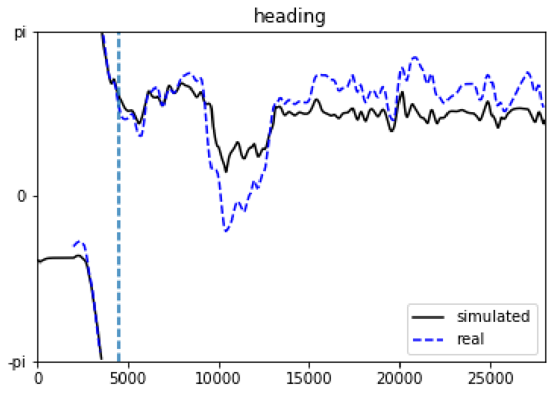
(c) 60% D7 inhibition

Figure 7: x axis is EB tile number, y axis is EPG activity

This also decreases the ability of the bump to track the heading (figure 8), although not as much as we might expect from experimental data (i.e. the model is not very sensitive to changes in this direction of parameter space).



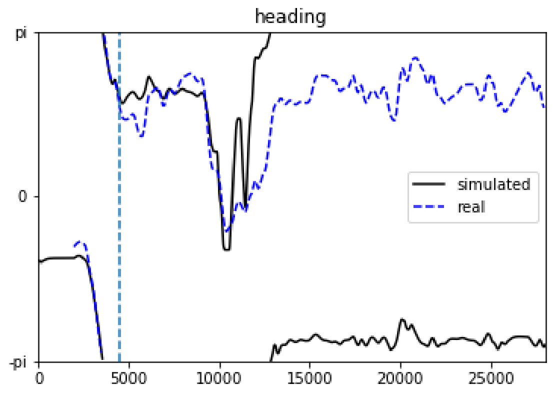
(a) 30% D7 inhibition



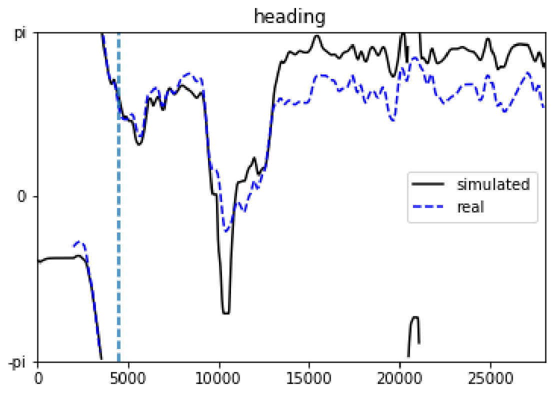
(b) 60% D7 inhibition

Figure 8: vertical line indicates time of inhibition (4500 ms).

The model is more sensive to inhibition of PEN1 outputs and also relatively robust to PEG inhibition (figure 9)



(a) 30% PEN1 inhibition



(b) 30% PEG inhibition

Figure 9: vertical line indicates time of inhibition (4500 ms).

Code

fr_class.py
implements the 'frmodel' class that is used for all simulations, storing weights and activities etc. Also has methods for calculating PVAs and plotting some different aspects of a simulation. The 'add_event' method allows for planning of changes in rotational velocity or shibire inhibition at particular timepoints.

analysis_functions.py
Simple functions for constructing velocity curves and carrying out sequential shibire inhibitions.

run_example_simulation.py
Contains a function for loading experimental data for a single fly and comparing this to simulated data.