**A multi-level model of the *Drosophila* flight system**

In recent years, myriad empirical studies have shed light on the neurobiology underpinning flight control in *Drosophila melanogaster*. From the measurement and classification of flight muscles [1] to the imaging of neural circuits encoding heading direction [2], there are many isolated results that we could understand better if we could test how they interact with other parts of the system. To that end, we are building an integrated predictive model to simulate the flight system at many levels, including neural control, wing kinematics, and aerodynamics.

We start by building an articulated rigid body representation of a fly based on high-speed video data, and render its inertial dynamics using the PyBullet physics engine (Fig. XA). However, a full computational fluid dynamic simulation of the wings would be too computationally intensive for an integrative system, so instead we implement a quasi-steady model which, at each time step, applies the forces that would be generated by the wing if it were moving at a constant velocity [3]. We have shown via dynamically scaled experiments that this is a good approximation to the forces generated by the wing’s complex vortex dynamics.

Then, we run a regression on the wing kinematics of flies in free flight to extract a basis set of wingstrokes that produce forces and torques in the principal directions, but these raw traces are high-dimensional and vary nonlinearly. So, we decompose each basis wingstroke into its Legendre polynomial components—analogous to Fourier decomposition—allowing us to treat the kinematics as a low-dimensional linear system. Control of linear systems is a well-studied problem, often solved using a proportional-integral-derivative (PID) controller (Fig. XB), and it turns out that a PID controller can be implemented as a neural circuit model.

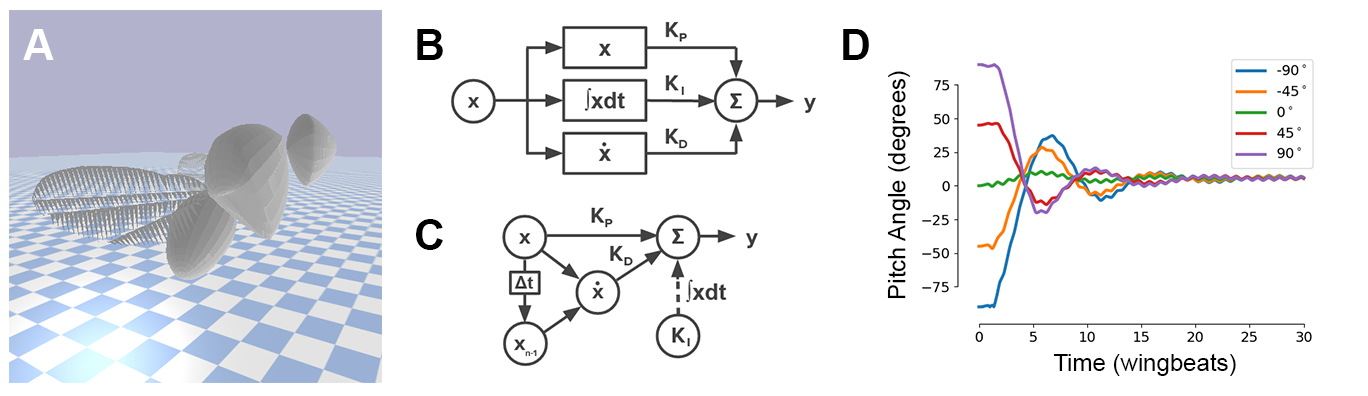
We structure the circuit as a directed graph, where at each time step a node’s value is calculated as the sum of the products of the nodes that connect to it and the weights of those connections (Fig. XC). We assume that the fly is operating in a context where it is able to sense its rotational state, so the raw proportional error signal is simply the input to the model, and we calculate the derivative by introducing a delayed error signal and subtracting it from the current error. In both cases the constant gain terms KP and KD are encoded in the weights of the downstream connections. The integral term, then, could be calculated by introducing recurrence, but here instead we use a Hebbian learning rule adapted from the classical Hopfield network model [4]:



where wijn is the weight of the connection between neurons i and j at timestep n, and xin is the activation strength of neuron i at timestep n. During steady flight, this results in a connection weight that is proportional to the time integral of the error term. The gain term KI is stored in a node with constant value.

To test this circuit in the virtual fly, we attempt to stabilize the axis that is most unstable in open-loop flight: pitch. The virtual fly senses its pitch angle and uses the PID circuit to choose the magnitude of the pitch-torque Legendre mode that will most efficiently move it toward an upright orientation, repeating this process at each timestep. The resulting wing kinematics succeed in stabilizing the fly within a handful of wingstrokes, even after they receive instantaneous perturbations up to and exceeding 90 degrees (Fig. XD). Even a fly perturbed to 180 degrees, upside-down, is able to recover to stable hover flight—albeit having lost some altitude in the process. Other axes are similarly controllable, though the system’s ability to respond to large perturbations in multiple axes simultaneously is limited.

There are many additional modules that we intend to integrate into the model, such as biomimetic sensor dynamics and explicit modeling of the flight muscle system, but it is already sufficiently complete to interface with empirical studies. This relationship is bidirectional, with experimental perturbation data refining the model and putting bounds on its application, and the model acting as a platform to test and generate hypotheses to guide experimental work.



**Figure X** (A) A render of the virtual fly in the PyBullet physics engine. (B) A classical representation of PID control, where x is the error signal and y is the control response. The system calculates three error terms and then actuates based on a sum weighted by the gains KP, KI, and KD. (C) A bioplausible neural representation of PID control, where nodes are neurons (or groups of neurons), solid lines are fixed-strength connections, the dotted line is a connection whose strength varies dynamically via Hebbian learning, and the Δt block is a time delay. During steady flight, this is mathematically equivalent to the classical representation. (D) The response of the virtual fly to a range of initial pitch perturbations. In all cases, the fly quickly returns to a stable upright hover.

[1] Lindsay, T., Sustar, A., and Dickinson, M. “The Function and Organization of the Motor System Controlling Flight Maneuvers in Flies.” Curr. Biol. 27 (3): 345-358.

[2] Turner-Evans, D., Wegener, S., Rouault, H., Franconville, R., Wolff, T., Seelig, J., Druckmann, S., Jayaraman, V. (2017). “Angular velocity integration in a fly heading circuit.” eLife 2017;6:e23496

[3] Dickson, W. B., Straw, A. D., Dickinson, M. H. (2008). “Integrative Model of Drosophila Flight.” AIAA 46 (9).

[4] Hopfield, J. J. (1982). “Neural networks and physical systems with emergent collective computational abilities.” Proc. Natl. Acad. Sci. USA. 79 (8): 2554–2558.