# 2024 Boulder School on Self-Organizing Matter
# Problems on "Autonomous Learning Metamaterials"

Doug Durian <djdurian@physics.upenn.edu>

1. Compute the local coupled learning rule, from the gradient of clamped minus free total power or energy, for the following edge elements and adjustable parameters. For each, keep in mind the two conjugate physical degrees of freedom and the learning degrees of freedom.

   (a) Resistors with resistances $R_i$, for current input/output tasks

   (b) Fluidic pipes with diameters $D_i$, for pressure tasks at low Reynolds number.

   (c) Capacitors with gaps $d_k$, for voltage tasks.

   (d) Springs with constants $k_i$ and rest lengths $L_i$, for motion tasks.

   (e) Springs with constants $k_i$ and rest lengths $L_i$, for force tasks.

2. Let's consider the humble voltage divider as a contrastive learning network. This device consists of two linear resistors with conductances $\{\kappa_1, \kappa_2\}$ that obey Ohm's law, $\Delta V = I/\kappa$, connected in series. One end of $\kappa_1$ serves as the input node to be held at voltage $V_{in}$, while the far end of $\kappa_2$ is grounded to zero volts; these are fixed boundary conditions. The middle node between the resistors is the output, whose voltage $V_{out}$ under "free" conditions is computed (or "inferred") physically. The learning goal is to adjust the conductances so that the output node learns the desired target voltage $V_{out} \to V_t = aV_{in}$ where $a$ is some chosen number between zero and one. Under "clamped" conditions, during training, the middle node is nudged toward the target by fixing its voltage to $V_c = \eta V_t + (1-\eta)V_{out}$ where $\eta$ is a hyperparameter between zero and one. These situations are simple enough to exactly solve for the learning rules in terms of the circuit parameters:

   (a) Under free conditions, draw a circuit diagram and find the output voltage two ways: by Kirchhoff's laws and by minimizing total power dissipated in the two resistors. Find the how the conductances must be related in order to achieve the desired target output voltage, and find total free power $P_f$.

   (b) Under clamped conditions, away from learning, draw a circuit diagram and find the total clamped power $P_c$. Note that this involves current flowing into or out of the middle node.

   (c) Compute the coupled learning rules for each conductor as given by gradient descent on the total power difference cost function $\mathcal{C} = (P_c - P_f)$. That is, compute $d\kappa_j/dt = -\gamma\partial(P_c - P_f)/\partial\kappa_j$ for the two edges, using your above results for clamped and free powers, and verify that these vanish at learning.

   (d) The local coupled learning rules for use in the lab are $d\kappa_j/dt = -\gamma\eta(V_{jc}^2 - V_{jf}^2)$, where $V_{jx}$ is the measured voltage drop across edge $j$ under boundary condition $x$. Compute these rules explicitly in terms of circuit parameters. Why are they not identical to gradient descent on the total power difference found in the previous part?

(e) Compute the global learning rules given gradient descent on the traditional loss function $\mathcal{L} = (V_{out} - V_t)^2$, and compare to all the above.

(f) (Open ended) Does it matter to what extent such learning rules coincide? Relevant issues are training speed, avoiding local minima, and finding generalizable solutions.

As an alternative or supplement, you could do this for a current divider (two resistors connected in parallel across a voltage source) or a bridge (four resistors in a ring, with a voltage source connected across one pair of opposite nodes and the goal of bringing the voltage difference across the other pair to zero).

3. (Open ended; there's probably an answer, which I'd like to know!) The minimal feedforward artificial neural network for computing the XOR function (0 if the two inputs are the same, 1 if they are different) has two nodes in the input layer, two nodes in a hidden layer, one node in the output layer, and six edges. What might be a minimal contrastive local learning network (CLLN) that can learn analog computation of XOR? The general issue is how to choose network architecture, input/output node selection, and type of nonlinearity.