Spring network allostery on an Erdős–Rényi architecture

Gil Raitses

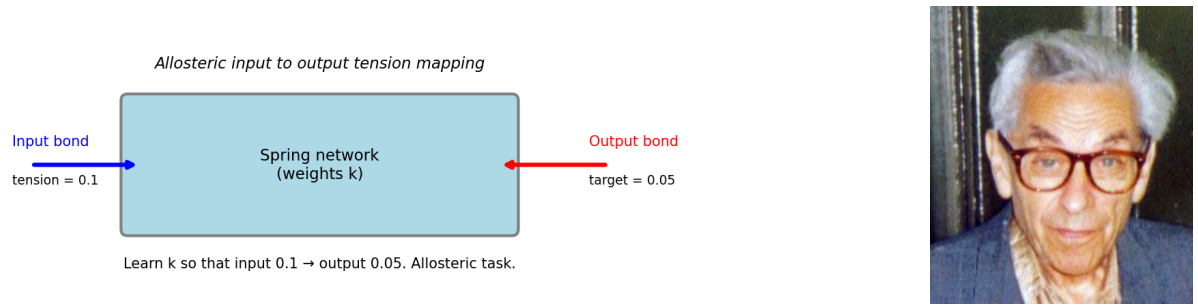
2026

Abstract

What is the simplest system that learns? This report uses one candidate, a spring network in linear response that learns an allosteric input–output map. One bond is driven at fixed tension (input). Another is trained toward a target tension (output). The network updates spring constants k using multi-mechanism learning (MmL). Two equilibrium states, input-driven and feedback-driven, share the same Laplacian. A local rule $k_{\text{new}} = k - \gamma\, t_{\text{in}}\, t_{\text{out}}$ modifies the weights. The architecture is Erdős–Rényi (ER) as a null model for connectivity. The allosteric error decreases with iteration and the learning curve depends on topology and on the choice of input and output bonds. ER serves as a baseline. Connectome-inspired topologies can be studied in the same framework. The setup is a minimal physical learner with a direct analogy to biological networks where plastic weights play the role of k. Key equations and code are in the appendix.

1 The simplest system that learns and the Erdős–Rényi baseline

The course asks what is the simplest system that learns. Flow networks, resistor networks, and spring networks in linear response are candidates. This report uses the spring network, with nodes at positions and edges as springs with constants k. The task is allostery, a long-range input–output map (one bond driven at fixed tension, another trained toward a target tension; see Figure 1).



(a) Task schematic.

(b) Paul Erdős.

Figure 1: Task and ER baseline. (a) Input bond at tension 0.1, output bond trained to 0.05; the learned allosteric map is this input–output coupling. (b) Paul Erdős (Budapest 1992); with Rényi he introduced the random-graph model used here. Topsy Kretts / Wikimedia Commons, CC BY 3.0.

Learning is the incremental update of k so that this allosteric mapping is acquired. The procedure is multi-mechanism learning (MmL), with two equilibrium states (input-driven and feedback-driven), the same physics for both, and a local rule that updates k from the tension patterns. The network architecture is Erdős–Rényi (ER) as a null model. The report gives the algorithm, shows that the allosteric error decreases with iteration, and shows that learning

depends on topology and on which bond pair is input and output. The result is a minimal physical learner analogous to biological networks whose plastic weights correspond to k.

The random-graph model used here has a distinct origin. Paul Erdős (1913–1996) had no permanent job or home. He traveled with a suitcase and showed up at colleagues' doors saying "My brain is open." He did mathematics in intense collaboration with many people, often working through the night and using amphetamine to stay awake. He later quit. When a friend bet he could not stop for a month, Erdős won the bet and said the friend had set mathematics back a month. With Alfréd Rényi he introduced the random-graph model. Each pair of nodes is connected with the same probability p, giving a Poisson degree distribution and no built-in structure. That collaborative, problem-driven culture produced the ER model as a baseline for "random" connectivity, the natural reference against which to compare real networks (e.g. brain connectomes). ER is used here for the same reason, a well-defined minimal case for learning allostery. Figure 1 illustrates the task and the origin of the ER baseline. The next section maps the spring network to biological networks; the following section gives the MmL loop.

2 Spring network learns allostery by updating k

The spring network has nodes at positions and edges as springs. Tension on each bond is the observable (the force along that spring, or equivalently the deviation from rest length scaled by k). The input bond is driven at fixed tension (e.g. 0.1). The output bond is trained toward a target tension (0.05). Spring constants k are the modifiable weights. They are updated by MmL. The procedure solves two equilibrium problems (input-pinned and output-pinned), both governed by the same Laplacian, and applies a local rule that uses the product of tensions at each bond in the two states. The details are in the next section. The network learns the allosteric map by changing k, not by backpropagation or hidden layers. It is a minimal physical learner. The graph is Erdős–Rényi so that connectivity is a random-graph baseline and null model. Later sections discuss how real connectomes deviate and how topology affects learning.

Table 1: Mapping of spring network to biological networks.

	Spring network	Biological networks
Weights	Spring constants k	Synaptic strength w
Plasticity	Δk (two equilibria)	Δw (LTP/LTD, reward $\times$ eligibility)
Task	Allosteric map (input $\rightarrow$ output tension)	Input $\rightarrow$ output (for example, stimulus $\rightarrow$ response)
Readout	Error, Δk per bond	Learning curve, Δw
Structure	Graph topology, which bond in/out	Circuit connectivity, synapse location

Table 1 maps this spring-network setup to biological networks. In many biological systems the input–output map involves sensory transduction and adaptation. A concrete example from mechanosensation and navigation in *Drosophila* larvae appears in §9.2.

3 MmL algorithm, two equilibria and same Laplacian

In the first equilibrium, pin the input bond at tension 0.1, minimize energy, and read tensions on all bonds. In the second equilibrium, pin the output bond at a desired tension that pushes toward the target and minimize again. The same Laplacian and same physics apply to both. Because the system is linear, a single stiffness matrix and its factorization are reused for both

equilibria. Only the boundary conditions differ. The weight update uses the product of the tension at each bond in the first equilibrium and the tension at the same bond in the second equilibrium, scaled by a learning rate γ. The weight update for non-pinned bonds is given in Appendix A. The loop consists of solving equilibrium 1, solving equilibrium 2, updating k, and repeating until the allosteric error is small. With the algorithm fixed, the graph is chosen. It is Erdős–Rényi so that connectivity is a random-graph baseline. Figure 2 shows that graph in the spatial layout used by the simulation. Section 5 shows the algorithm loop and ER connectivity (Figure 3).

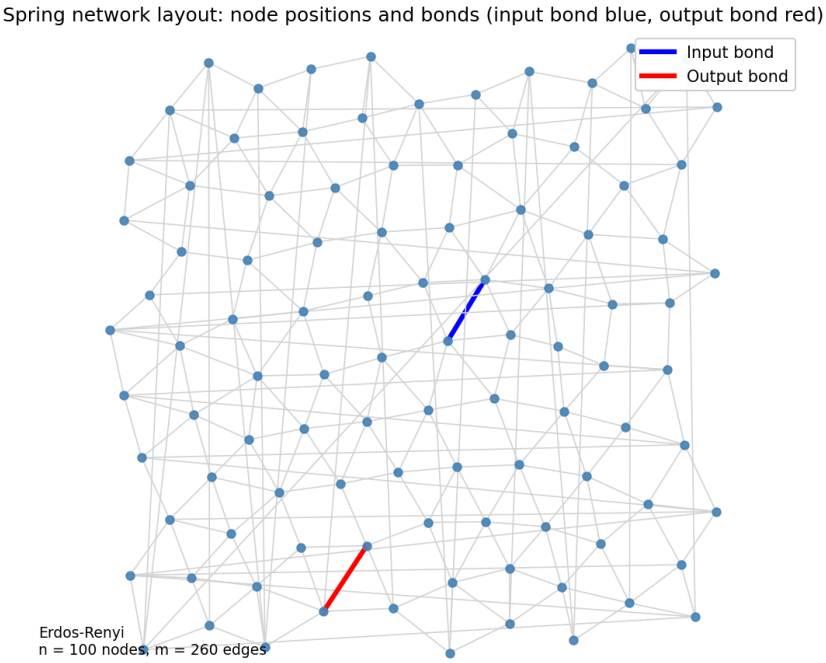


Figure 2: Same ER network in the spatial layout used by the simulation. Node positions (x, y) live in the $L_x \times L_y$ box. Edges are unchanged from the connectivity view. Blue and red mark the input and output bonds, where the drive (0.1) is applied and where the tension trained toward 0.05 is read. This is the geometry on which the spring energy is minimized and the allosteric signal propagates. Topology and bond choice together determine how well the network learns.

4 Erdős–Rényi as null model, biological model systems

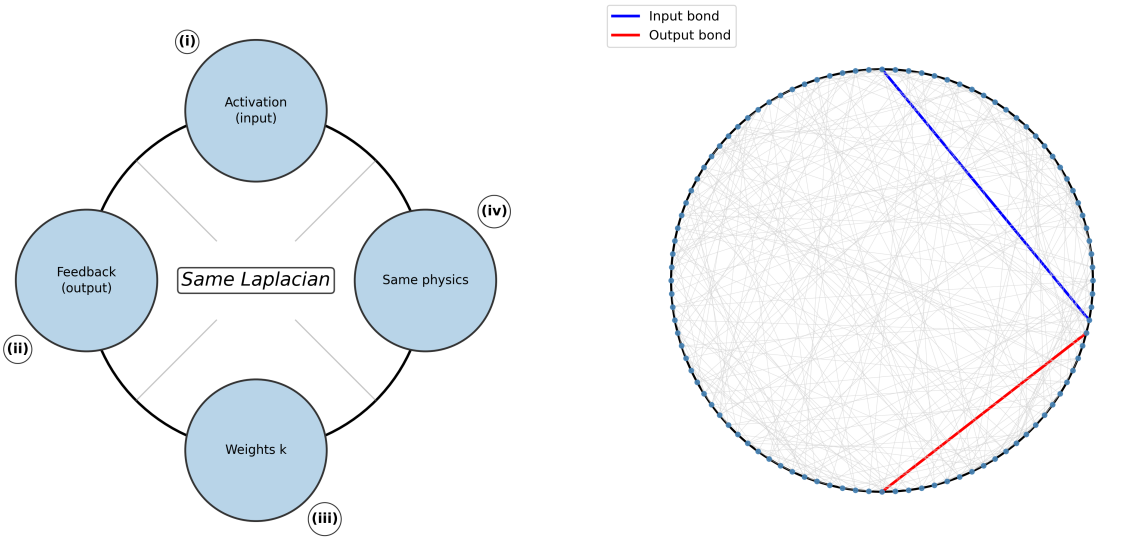
The architecture is the Erdős–Rényi random graph. Section 1 gives its origin and context. Connectivity is random. Degree distribution is Poisson. ER is the standard null model for cortical and large-scale neural connectivity. Biological model systems such as *C. elegans*, Drosophila, and cat and macaque cortex have connectomes that deviate from ER in degree distribution and correlations. Figure 3 shows the algorithm loop and the ER connectivity.

The circle-and-chords drawing is a convenient way to visualize an ER graph without committing to physical positions. The important information is the adjacency structure. The same graph is shown in the spatial layout in Figure 2; that layout is used when the simulation minimizes spring energy and computes tensions. That layout determines where the input and output bonds sit and how the allosteric signal propagates.

5 ER network and learning loop in the provided notebook

The network has $N = 100$ nodes and $L_x = L_y = 10$. The graph is ER with $p \approx 0.06$, giving about 300 edges. The largest connected component is used if the graph is disconnected. The Network_Generation_and_Learning notebook writes `positions.txt` and `connections.txt`, then runs the learning loop. Parameters include the learning rate γ, the feedback scaling η, target output tension 0.05, and number of iterations. See Appendix B for the exact parameter block.

$$N = 100, \quad L_x = L_y = 10, \quad p \approx 0.06, \quad \gamma = 2500, \quad \eta = 1, \quad \text{target } t_{\text{out}} = 0.05.$$



(a) MmL algorithm loop.
MmL: activation → feedback → weight update $k_{\text{new}} = k - \gamma\, t_{\text{in}}\, t_{\text{out}}$.

(b) ER network connectivity.
Nodes on a circle, edges as chords; input and output bonds highlighted.

Figure 3: Algorithm and ER connectivity. (a) MmL algorithm loop. (i) Activation: input bond pinned; (ii) feedback: output bond pinned toward target; (iii) weights k updated from tension products; (iv) same physics (same Laplacian) for both equilibria. The map (i)→(ii) uses the same stiffness matrix; (iii) is the local rule $k_{\text{new}} = k - \gamma\, t_{\text{in}}\, t_{\text{out}}$. (b) The random graph used in this report. Nodes on a circle in arbitrary order, edges as chords. This view shows who is connected to whom, the structure that determines how tension propagates from input to output. ER gives a random baseline (each pair connected with probability p, Poisson degrees). Real connectomes are often compared to this null. The circle layout makes connectivity and clustering visible.

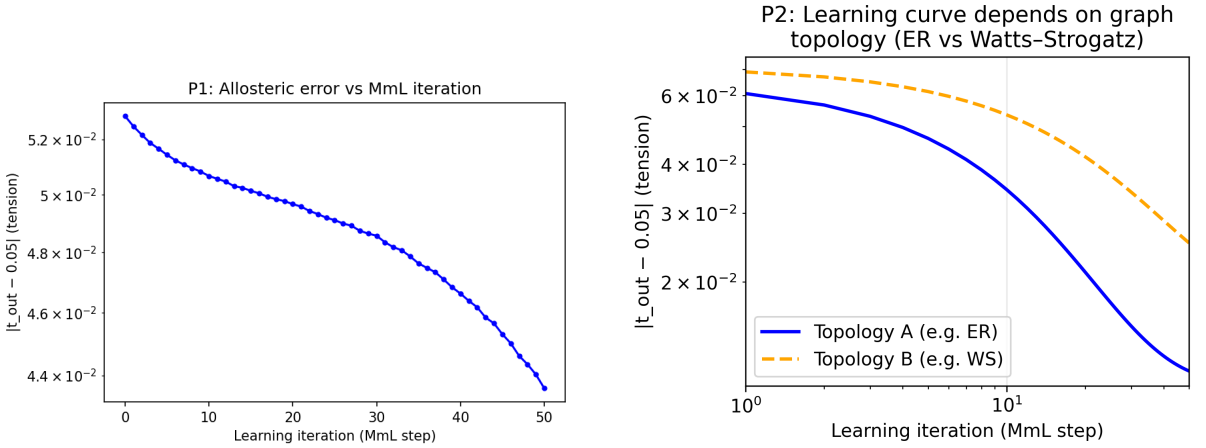
Running the notebook produces a sequence of equilibrium solves and k updates. The natural readout is how fast the allosteric error drops and whether it reaches a small value. The network does learn the map and the learning curve depends on topology. §7 and Figure 4 show the result.

6 Allosteric error decreases with iteration, prediction tested

The quantity $|t_{\text{out}} - 0.05|$ decreases with iteration and the network learns the allosteric mapping. Learning-curve shape depends on topology. Figure 4 shows the error-vs-iteration curve and the topology comparison. The figure is produced by the attached notebook.

7 Predictions as architecture or size change

The same network is used with two different input-output bond pairs. Error vs iteration is computed for each. Convergence speed and final error can differ. The choice of which bond is input and which is output changes the learning curve. Learning curve shape depends on graph topology. As in the proposal, P1 is that the network learns (error decreases with iteration) and P2 that topology affects learning; Figure 4 shows both. The notebook also compares P3 (input–output bond choice), P4 (gamma and eta), and P5 (plasticity on path, with larger $|\Delta k|$ on bonds between input and output). Convergence and final error vary with these choices.



(a) P1: error vs MmL iteration.

(b) P2: topology affects learning.

Figure 4: Learning curves from the MmL loop. (a) P1: allosteric error $|t_{\text{out}} - 0.05|$ versus MmL iteration. The vertical axis is the absolute deviation of the output-bond tension from the target 0.05. The horizontal axis is iteration number. Error decreases as the network updates spring constants k, so the allosteric input–output map is acquired. (b) P2 compares learning curves for different graph topologies. Same task and update rule. Only the connectivity differs. Convergence speed and final error depend on topology, showing that learning in this minimal physical system is structure-dependent. Produced by the attached notebook.

The curves show that the minimal physical learner succeeds. The allosteric error goes down. Different topologies or bond choices give different convergence. The rest of the report ties the result back to MmL, ER as null model, and biological analogy.

8 MmL as minimal physical learner, allostery as input-to-output map

The course asks what is the simplest system that learns. Flow networks, resistor networks, and spring networks in linear response are candidates. The spring network is that minimal physical learner. It has no hidden layers or backpropagation, only nodes, edges as springs, and a local update rule that uses two equilibrium states (input-driven and feedback-driven) to change the weights k. Learning is the incremental modification of those connection strengths so that the network's input–output behavior acquires the desired functionality.

The task that is learned is allostery. Allostery is a long-range mechanical input–output map. A perturbation at one site, the input bond where tension is fixed at 0.1, is transmitted through the network and read out at another site, the output bond trained toward tension 0.05. The signal propagates via the same physics that governs both equilibria, the same Laplacian and the same stiffness matrix, so the mapping is determined by topology and by the current k. Updating

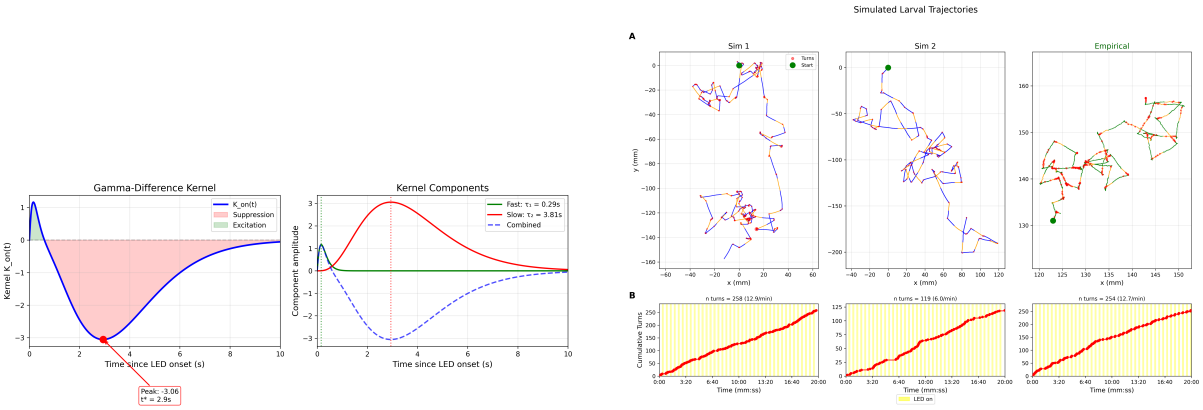
k changes the map. Figure 4 shows that this works. The allosteric error $|t_{\text{out}} - 0.05|$ decreases with iteration. The learning curve depends on topology and on which bond pair is chosen as input and output. The same conceptual framing, input site, output site, mechanical coupling, plasticity of the coupling strengths, appears in the course's protein-allostery discussion and in biological networks where synaptic weights play the role of the spring constants k.

9 Biological context and example

9.1 Biological context, ER and real connectomes

ER is used as a null model for connectivity. It assumes that each pair of nodes is connected with the same probability p, giving a Poisson degree distribution and no built-in community structure or hubs. The question is whether real networks look like that. Empirical brain networks, *C. elegans*, Drosophila, cat and macaque cortex, are often reported to deviate from ER. For example, degree distributions may be heavy-tailed or more regular than Poisson, or show higher clustering and modularity than random graphs typically show. Comparing real connectomes to ER separates structure that is "just random" from structure that may reflect development, function, or constraint. Using ER here is a deliberate baseline. The claim is not that brains are ER, but that learning allostery on a random graph is a well-defined minimal case. Richer, connectome-inspired topologies such as those from larval wiring diagrams or cortical atlases can be plugged into the same learning loop to ask how structure shapes the input–output map.

Extensions beyond the single input–output pair include multi-target allostery, signal routing, and robustness. Those directions, and the use of data from biological model systems such as Drosophila larva to set graph topology, connect this minimal physical learner to the broader theme of learning in networks of interactions.



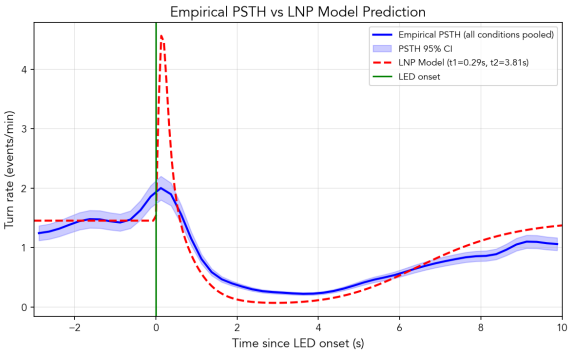
(a) Stimulus–response kernel.

(b) Example trajectories.

Figure 5: Drosophila larva as a biological input–output system. (a) Fitted hazard kernel. Gamma-difference kernel captures biphasic dynamics, with fast excitation ($\tau_1 \approx 0.3\,\text{s}$) and slow suppression ($\tau_2 \approx 4\,\text{s}$). Input (LED) $\rightarrow$ output (reorientation hazard). (b) Example trajectories. Simulated larval paths (5 min). Runs (blue) and turn events (red). Yellow shading indicates stimulus on. Turns are suppressed during stimulation [5].

9.2 Biological example (Drosophila larva)

Mechanosensation and navigation in *Drosophila* larvae have been studied with optogenetics and hazard kernels [5]. The mapping from stimulus to reorientation is characterized by a fitted hazard kernel (biphasic, fast and slow components). Figures 5 and 6 illustrate the kernel, trajectories, and PSTH validation; the supplement gives further validation. A follow-up study [6] addresses



(a) Empirical vs model turn rate aligned to LED onset.



(b) Peristimulus turn rate by condition.

Figure 6: Drosophila larva PSTH validation and condition-dependent dynamics. (a) Peristimulus turn rate (events/min): empirical PSTH with 95% CI (blue) and LNP model prediction (red) with kernel timescales $\tau_1 = 0.29\,\text{s}$, $\tau_2 = 3.81\,\text{s}$. Same input–output framing as the spring network: stimulus in, behavioral readout out. (b) Four experimental conditions (0–250 PWM vs. 50–250 PWM; constant vs. ramp). Turn rate (events/s) aligned to LED onset; pink shading indicates peak-intensity period. Condition-dependent amplitude with same biphasic shape supports a single kernel form with condition-specific scaling [5].

individual-level inference. Connectome-inspired topology could use larval wiring in the same framework.

10 Conclusion

A spring network on an Erdős–Rényi architecture learns an allosteric input–output tension mapping by updating spring constants k with multi-mechanism learning. The procedure uses two equilibria, input-driven and feedback-driven, the same Laplacian, and the local rule $k_{\text{new}} = k - \gamma\, t_{\text{in}}\, t_{\text{out}}$. The allosteric error decreases with iteration. The learning curve depends on topology and on the choice of input and output bonds. ER serves as a null model for connectivity. Real connectomes deviate from it. Connectome-inspired topology can be used in the same framework. This minimal physical learner illustrates the course themes of "simplest system that learns" and allostery as a long-range input–output map, with a direct analogy to biological networks where plastic weights play the role of k. Figures 5 and 6 show the Drosophila larva as a concrete biological system with the same input–output structure, stimulus in, behavioral readout out, and a learned or fitted mapping between them. Open questions remain. Further investigation and experiments with the other predictions (P3–P5, namely input–output bond choice, γ and η, and plasticity on the path) could shed light on how pathway choice, learning-rule strength, and where weights change shape the acquired allosteric map.

11 References

1. Emergent Intelligence course. (2026). *Module 2: Learning with linear networks*. PHY600, Spring 2026. (review)

2. Laydevant, J., et al. (2024). Learning in networks of interactions. *Neuron*. (primary)

3. Erdős, P., & Rényi, A. (1959). On random graphs I. *Publicationes Mathematicae*, **6**, 290–297. (review)

4. Bullmore, E., & Sporns, O. (2009). Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, **10**(3), 186–198. (review)

5. Raitses, G., Goonawardhana, D., & Mihovilovic-Skanata, M. (2025). Temporal dynamics of mechanosensory behavior in *Drosophila* larvae. (source of Figure 5)

6. Raitses, G., et al. (2025). Quantitative guidelines for behavioral phenotyping. Manuscript in preparation.

Appendix

A Key equations

The MmL weight update for non-pinned bonds is

$$k_{\text{new}} = k - \gamma\, t_{\text{in}}\, t_{\text{out}} \tag{1}$$

The allosteric error (P1) is

$$\text{error} = |t_{\text{out}} - 0.05| \tag{2}$$

Total spring energy is harmonic and minimized at each equilibrium. Bond tension is $t = -k(d - \ell_0)$ where d is current length and ℓ_0 is rest length.

In the Erdős–Rényi model, each pair of nodes is connected with probability p. The degree distribution is Poisson.

B Code

The key parameters below are from Network_Generation_and_Learning.ipynb.

```
n = 100
p_edge = 0.06   # edge probability, ~300 edges
Lx, Ly = 10.0, 10.0
iterations = 51
gamma = 2500.0
eta = 1.0
target_out = 0.05
# Input bond pinned at tension 0.1. Output bond trained toward 0.05
```

The MmL update applies to non-pinned bonds only. Pinned bonds keep fixed k. The full implementation is in Network_Generation_and_Learning.ipynb.

```
update_mask = ~(mask_in | mask_out)   # exclude input and output bonds
k_new[update_mask] = k_stiff[update_mask] - gamma * t_in[update_mask] * t_out[
    ↪ update_mask]
k_new[k_new < 0.0] = 0.0001           # clamp minimum k
```

To reproduce the learning curve (Figure 4) and the results in this report, run the attached notebook.