Ctrl & Comp in Bio Sys - Westlake Univ., Fall 2025

Lecture 08: Stochastic Kinetics of Markov Chain and Computation Biomachines

2025-10-30

Lecturer: Fangzhou Xiao *Scribe: Xia Yao+Jiacheng Wei+Yihan Gong*

Contents

1 Equilibrium Physics of Bio-regulation

1.1 The Measurement Problem

The most practical motivation for equilibrium physics is the difficulty of measuring kinetic rates compared to thermodynamic energies.

1.The Kinetic Challenge

Consider a standard biological binding reaction, such as a gene (D) binding to a transcription factor (P), or an enzyme (E) binding to a substrate (S).

The dynamic model is described by the Law of Mass Action:

$$E + S \underset{k_-}{\overset{k_+}{\rightleftharpoons}} ES \tag{1}$$

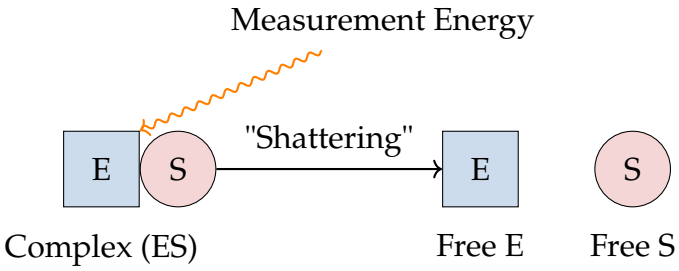
To model this dynamically, we need two parameters:

- k_+: The association rate constant.
- k_-: The dissociation rate constant.

Why is this hard to measure? To find k_+ and k_-, you must measure the concentrations of the free components (E, S) and the complex (ES) in real-time. However, the biological bond between E and S is usually non-covalent (e.g., hydrogen bonds, Van der Waals forces). These are **weak associations**.

The "Shattering" Effect (Mass Spectrometry) Common measurement tools, like Mass Spectrometry, require imparting energy to the molecules to detect them.

- When you apply energy to measure the system, the weak non-covalent bond breaks.
- You cannot distinguish whether the detected molecules were originally a complex (ES) or two separate molecules ($E + S$).
- The act of measurement destroys the structure you are trying to observe.



2.The Equilibrium Solution: Thermodynamics

Instead of trying to measure k_+ and k_- individually, we look at the system in equilibrium. Here, we care about the ratio, the **Dissociation Constant** (K_d):

The Equilibrium Relation

$$K_d = \frac{[E][S]}{[ES]} = \frac{k_-}{k_+} \tag{2}$$

Crucially, statistical physics relates this ratio to **Energy**. The probability of a state is proportional to the Boltzmann factor:

$$P \propto e^{-E_{state}/k_B T} \tag{3}$$

Therefore, K_d is directly related to the Binding Free Energy (ΔG):

Energy-Measurement Link

$$K_d \propto e^{\Delta G/k_B T} \tag{4}$$

Why is this easier?

- Energy changes (ΔG) correspond to the release or absorption of **Heat**.
- We can measure heat changes very accurately (e.g., Isothermal Titration Calorimetry) without needing to count individual molecules or preserve fragile bonds during detection.
- **Conclusion:** Equilibrium is a "Measurement Science." It transforms a hard counting problem into an easier calorimetry problem.

1.2 The Bio-Design Perspective

If you are an engineer or synthetic biologist trying to build a biological function (e.g., a genetic switch), equilibrium mechanisms are often superior.

1.Low Energy Cost

- **Non-Equilibrium:** Requires constant energy input (flux) to maintain a steady state (like keeping a lightbulb on).
- **Equilibrium:** Once the system settles, it stays there forever without consuming fuel (ATP). It is thermodynamically stable.

2.Low Mechanistic Complexity

- Designing a cycle that forces a reaction in one direction (Non-equilibrium) requires complex molecular machinery to couple with energy sources.
- Designing for equilibrium just requires mixing components that have the right binding affinity (ΔG). They will naturally find their way to the desired state.

1.3 The Analytical Perspective

From the perspective of system analysis (Mathematical Modeling), equilibrium provides powerful constraints that make unsolvable problems solvable.

1.Dynamics are Hard

- **Deterministic:** Systems of coupled differential equations are often non-linear and hard to solve analytically.
- **Stochastic:** If we treat it as a random process, the Master Equation is usually infinite-dimensional and impossible to solve exactly.

2.Equilibrium is Easy

If we assume the system is at equilibrium, we don't need to solve the dynamics. We know the answer immediately because of the **Boltzmann Law**:

$$P(State_i) = \frac{1}{Z} e^{-E_i/k_B T} \tag{5}$$

Where Z is the partition function.

Key Constraints Equilibrium imposes strict physical rules that simplify the math:

1. **Detailed Balance:** The flux between any two states is zero.

$$P_A \cdot k_{A\to B} = P_B \cdot k_{B\to A} \tag{6}$$

2. **No Cycle Fluxes:** There is no net rotation around reaction loops.

Summary of Section 1

Even though biology is dynamic, we use equilibrium physics because:
1. We can measure "Energies" (Heat) much easier than "Rates".
2. Equilibrium systems are energy-efficient and simpler to design.
3. Equilibrium math is solvable via the Boltzmann distribution.

2 Markov Chains: Stochastic Kinetics and non-equilibrium

Traditional thermodynamics often focuses on equilibrium—the state of a system after an infinite amount of time. However, in biological systems (like cells), we are often interested in kinetics (how fast things happen) and non-equilibrium behaviors (energy consumption).

The Kinetics Problem

Even if we know the final state of a cell, we often ask: *"If a signal changes in the environment, how long does it take for the cell to switch states?"*

- This is known as the First Passage Time problem.
- Simply adding up the inverse rates of steps ($1/k_1 + 1/k_2 \dots$) is incorrect because stochastic systems can transition backward.

2.1 Non-Equilibrium Systems

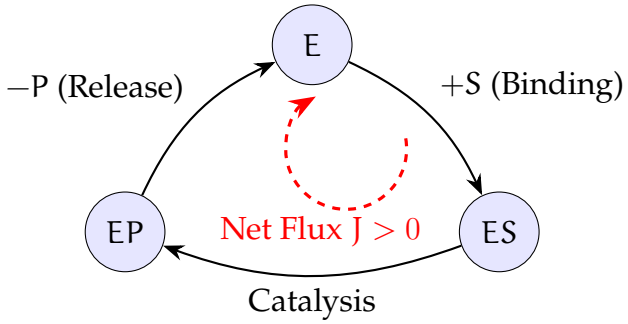
Energy vs. Non-Equilibrium

A common misconception is that if a system consumes energy (e.g., ATP hydrolysis), it is automatically "non-equilibrium."

- Correction: Many energy-consuming systems still exhibit behaviors that fit equilibrium models perfectly.
- True Indicator: A system requires a non-equilibrium model only when it exhibits behaviors impossible in equilibrium, such as non-monotonicity or net cyclic fluxes .

Example: The Enzymatic Cycle

Consider an enzyme E that converts Substrate S to Product P.



- Equilibrium: Forward and backward rates balance perfectly. No net rotation.
- Non-Equilibrium: High concentration of S drives the cycle continuously in a clockwise direction (Flux). This is a "current" in the state space.

2.2 Finite Markov Chains & The Master Equation

To model these kinetics, we use Finite State Markov Chains . This is a specific application of the Chemical Master Equation (CME).

From Concentrations to Probabilities

The lecture demonstrates that for 1st-order reactions , the macroscopic rate equation is mathematically identical to the single-molecule probability equation.

Consider a simple reversible transition:

$$A \underset{k_1}{\overset{k_2}{\rightleftharpoons}} B \tag{7}$$

(Note: Notation assumes k_2 creates A, k_1 consumes A).

1. Macroscopic View (Rate Equation) Let $A(t)$ and $B(t)$ be the number of molecules. The total number $N_{tot} = A + B$ is constant. The rate of change of A is:

$$\frac{dA}{dt} = k_2 B - k_1 A \tag{8}$$

Variable Definitions:

- dA/dt: Change in number of molecules of A per unit time.
- k_1: Rate constant for $A \to B$ (units: $time^{-1}$).
- k_2: Rate constant for $B \to A$ (units: $time^{-1}$).

2. Microscopic View (Probability) We define the probability (or fraction) of being in state A as:

$$P_A = \frac{A}{N_{tot}}, \quad P_B = \frac{B}{N_{tot}} \tag{9}$$

Since the system is linear, we can divide the rate equation by the constant N_{tot}:

$$\frac{1}{N_{tot}} \frac{dA}{dt} = k_2 \frac{B}{N_{tot}} - k_1 \frac{A}{N_{tot}} \tag{10}$$

Substituting the probability definitions:

$$\boxed{\frac{dP_A}{dt} = k_2 P_B - k_1 P_A} \tag{11}$$

Conclusion on Linearity

Because the reaction is linear (1st order), the math governing billions of molecules (concentration) is the exact same math governing one molecule's probability . We can simply model one molecule hopping between states.

2.3 Mathematical Formalism: The Q-Matrix

We generalize the system to n states using Linear Algebra.

The Probability Vector

Let $p(t)$ be a column vector representing the probability distribution at time t:

$$p(t) = \begin{bmatrix} P_1(t) \\ P_2(t) \\ \vdots \\ P_n(t) \end{bmatrix} \tag{12}$$

where $\sum_{i=1}^{n} P_i(t) = 1$.

The Transition Rate Matrix (Q)

The evolution of probability is given by the differential equation:

$$\frac{dp}{dt} = Qp \tag{13}$$

The matrix Q is constructed as follows:

- Off-diagonal terms (q_{ij} where $i \neq j$): The rate of transitioning **FROM** state j **TO** state i.
- Diagonal terms (q_{ii}): The negative sum of all outgoing rates from state i.

Mathematical Detail: Conservation of Probability

Since probability cannot be created or destroyed, the columns of Q must sum to zero.

$$\sum_{i=1}^{n} q_{ij} = 0 \tag{14}$$

This implies the diagonal term is:

$$q_{jj} = -\sum_{i \neq j} q_{ij} \tag{15}$$

Physical meaning: The rate of leaving state j (q_{jj}) is exactly the sum of rates going to all other states.

2.4 Steady States and Ergodicity

Steady State Distribution (π)

At steady state, the probability distribution does not change with time.

$$\frac{dp}{dt} = 0 \implies Q\pi = 0 \tag{16}$$

Here, π is the Steady State Distribution . Mathematically, it is the Right Eigenvector of matrix Q corresponding to the eigenvalue $\lambda = 0$.

Ergodicity

Definition: Strong Connectivity

A graph is strongly connected if there is a path from every state to every other state.

If the state transition graph is strongly connected (Irreducible):

1. The steady state π is Unique .
2. The system is Ergodic .

Implication for Simulation: Due to ergodicity, the Time Average equals the Ensemble Average .

$$\text{Time Average of 1 trajectory} = \text{Average over infinite populations}$$

You only need to simulate **one** single molecule trajectory for a long time to calculate the distribution of the entire population.

2.5 Detailed Balance (Equilibrium vs. Steady State)

A system can be in a steady state (constant concentrations) without being in thermodynamic equilibrium (e.g., a battery powering a circuit).

Detailed Balance Condition

For a system to be in true Equilibrium , the flux between any two specific states must balance out to zero.

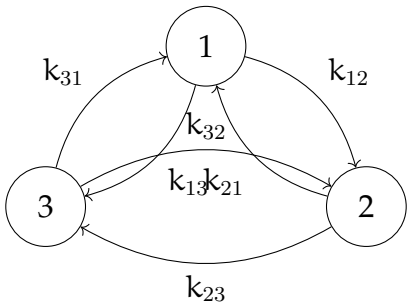
$$\text{Flux}_{j\to i} = \text{Flux}_{i\to j} \tag{17}$$

$$\pi_j q_{ij} = \pi_i q_{ji} \tag{18}$$

Cycle Condition (Kolmogorov Criterion)

An easier way to check for equilibrium is to look at loops in the graph. For any closed cycle (e.g., $1 \to 2 \to 3 \to 1$):

$$\text{Product of Clockwise Rates} = \text{Product of Counter-Clockwise Rates} \tag{19}$$



Equilibrium Requirement:

$$k_{12} \cdot k_{23} \cdot k_{31} = k_{13} \cdot k_{32} \cdot k_{21} \tag{20}$$

If this equation does not hold, there is a net flux, and the system is "Non-Equilibrium".

2.6 The Philosophy of "State": Markovian vs. Non-Markovian

There is a debate in biology that biological systems depend on history (memory) and are therefore "Non-Markovian."

The Lecturer's Counter-Argument

The lecturer argues that all dynamical systems are Markovian if you define the "State" correctly.

Definition: State

A State is a collection of variables that summarizes **all** necessary information from the past to predict the future evolution of the system.

Example: Hidden Variables

If a system appears to depend on history (x_t depends on x_{t-1} and x_{t-2}), it is only because we defined the state too narrowly.

- Narrow View (Non-Markovian): State = $[x_t]$. The future depends on the past history.
- Expanded View (Markovian): State = $\begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix}$. Now, the current vector contains all the information needed for the next step.

Analogy: In Newtonian mechanics, position x alone is not a state (need velocity). But (x, v) is a state.

Conclusion: When modeling cell biology, assuming a Markov chain is not a restrictive assumption about "lack of memory." It is simply an assumption that the variables we chose (e.g., gene expression levels) are sufficient to define the system's future.

2.7 First-Passage Time to an Active State

Consider a CTMC with discrete states. One state is designated as **active** (state A). We want to compute: **How long does it take to reach A starting from some initial state i?**

Notation:

- $X(t)$: state of the system at time t
- $Q = (q_{ij})$: generator matrix
  - $q_{ij} \geqslant 0$ for $i \neq j$ (transition rate $i \to j$)
  - $q_{ii} = -\sum_{j \neq i} q_{ij}$ (total exit rate from i)
- First-passage time: $T_i = \inf\{t \geqslant 0 : X(t) = A \mid X(0) = i\}$
- Mean first-passage time: $m_i = \mathbb{E}[T_i]$
- CDF: $F_i(t) = P(T_i \leqslant t)$

Mean First-Passage Time Calculation

Intuitive Derivation: Starting from state $i \neq A$:

1. Wait in state i for exponentially distributed time with rate $-q_{ii}$. Mean waiting time = $1/(-q_{ii})$.
2. Jump to state $j \neq i$ with probability $\frac{q_{ij}}{-q_{ii}}$.
3. After arriving at j:
   - If $j = A$: done (extra time = 0)
   - If $j \neq A$: need additional mean time m_j

Thus:

$$m_i = \frac{1}{-q_{ii}} + \sum_{j \neq i} \frac{q_{ij}}{-q_{ii}} m_j \tag{21}$$

Multiply by $-q_{ii}$:

$$-q_{ii} m_i = 1 + \sum_{j \neq i} q_{ij} m_j \tag{22}$$

Since $m_A = 0$, we separate terms:

$$\boxed{-q_{ii} m_i = 1 + \sum_{\substack{j \neq i \\ j \neq A}} q_{ij} m_j} \tag{23}$$

This is a system of linear equations for m_i ($i \neq A$).

Matrix Formulation: Make state A absorbing: set $q_{Aj} = 0$ for all j.

Partition states: **transient states** T and **absorbing state** A. Reorder so transient states come first:

$$Q = \begin{pmatrix} \tilde{Q} & \mathbf{r} \\ \mathbf{0} & 0 \end{pmatrix}$$

- $\tilde{Q}$: $n_T \times n_T$ matrix for transient states
- $\mathbf{r}$: exit rates from transient to absorbing state
- $\mathbf{0}$: row for A

Let $\mathbf{m}$ = vector of m_i for $i \in T$, and $\mathbf{1}$ = column vector of ones.

Then:

$$\boxed{\mathbf{m} = (-\tilde{Q})^{-1}\mathbf{1}} \tag{24}$$

Proof: From $-q_{ii}m_i = 1 + \sum_{j \in T, j \neq i} q_{ij} m_j$, rewrite as:

$$(-q_{ii})m_i - \sum_{j \in T, j \neq i} q_{ij} m_j = 1$$

This is the i-th row of $(-\tilde{Q})\mathbf{m} = \mathbf{1}$.

2.8 Distribution and Example

Distribution of First-Passage Time

Let $F_i(t) = P(T_i \leqslant t)$.

Backward Equation: Condition on first jump:

$$\begin{aligned} F_i(t) &= \int_0^t (-q_{ii}) e^{q_{ii}s} \left[\sum_j \frac{q_{ij}}{-q_{ii}} F_j(t-s) \right] ds \\ &= \int_0^t e^{q_{ii}s} \left[\sum_{j \neq i} q_{ij} F_j(t-s) \right] ds \end{aligned}$$

Differentiate with respect to t:

$$\boxed{\frac{dF_i}{dt} = q_{ii} F_i(t) + \sum_{j \neq i} q_{ij} F_j(t)} \tag{25}$$

Boundary Conditions:

- $F_A(t) = 1$ for $t \geqslant 0$
- $F_i(0) = 0$ for $i \neq A$

Matrix Form: Let $\mathbf{F}(t)$ = vector of $F_i(t)$ for $i \in T$. Then:

$$\frac{d\mathbf{F}}{dt} = \tilde{Q}\mathbf{F}(t) + \mathbf{r} \tag{26}$$

with $\mathbf{F}(0) = \mathbf{0}$.

Solution using matrix exponential:

$$\mathbf{F}(t) = \int_0^t e^{\tilde{Q}s} \mathbf{r} \, ds \tag{27}$$

If $\tilde{Q}$ is invertible:

$$\mathbf{F}(t) = \left(e^{\tilde{Q}t} - I \right) \tilde{Q}^{-1} \mathbf{r} \tag{28}$$

Example: Three-State System

System Description: States: 1 (initial), 2, A (active).

Transition rates:

$$1 \xrightarrow{k_1} 2$$
$$2 \xrightarrow{k_2} A$$
$$2 \xrightarrow{k_b} 1$$
$$1 \xrightarrow{k_f} A$$

Generator Matrix:

$$Q = \begin{pmatrix} -(k_1 + k_f) & k_1 & k_f \\ k_b & -(k_b + k_2) & k_2 \\ 0 & 0 & 0 \end{pmatrix}$$

Mean First-Passage Times: Transient states: $\{1, 2\}$.

$$\tilde{Q} = \begin{pmatrix} -(k_1 + k_f) & k_1 \\ k_b & -(k_b + k_2) \end{pmatrix}$$

$$-\tilde{Q} = \begin{pmatrix} k_1 + k_f & -k_1 \\ -k_b & k_b + k_2 \end{pmatrix}$$

$$\mathbf{m} = (-\tilde{Q})^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

The explicit solution:

$$\begin{aligned} \det\bigl(-\tilde{Q}\bigr) &= (k_1 + k_f)(k_b + k_2) - k_1 k_b \\ &= k_1 k_2 + k_f k_b + k_f k_2 \end{aligned}$$

$$\mathbf{m} = \frac{1}{\det\bigl(-\tilde{Q}\bigr)} \begin{pmatrix} k_b + k_2 + k_1 \\ k_b + k_1 + k_f \end{pmatrix}$$

So:

$$m_1 = \frac{k_b + k_2 + k_1}{k_1 k_2 + k_f k_b + k_f k_2}$$
$$m_2 = \frac{k_b + k_1 + k_f}{k_1 k_2 + k_f k_b + k_f k_2}$$

3 Computation Biomachine

Again like in adaptation biomachine, when we consider biological systems, we often consider they have distinct functions, like computation. To achieve such functions, we have built engineered machines, eg, computer. Since biological systems also need to achieve the same tasks, the structures and design principles of engineered machines could be used to better understand why biological systems are built in that way.

3.1 Today's Computation biomachine

Biological systems need to process information to respond to environments.

- Example:
  - activate a gene expression
  - to take up new nutrient
  - start a stringent response when harsh conditions hit
  - become a new cell type when growth factors say "differentiate"

So cells for sure need to do computation, but how is it done? How complex a computation does the cell achieve in that way? In ENgineered machines, we built computational units, and then linked them up for the larger and larger computation.

And now let us search for the existing biological computational unit.

logic gates

We built computers consisting of binary logic gates to perform computation. Maybe cells do the same?

- Example:
  - AND Gate
  - OR gate
  - NOT gate

Cells have chemical reactions inside, maybe implement gates via chemical reactions (catalysis)? Input and output turn out to be the concentration of species.

- '0' corresponds to low conc relative to the total conc
- '1' corresponds to high conc relative to the total conc

- Example:
  - The variable Y is represented in two forms, Y0 and Y1. (e.g., like phosphorylation and dephosphoorylation)

    $$Y0 \Leftrightarrow Y1$$

  - Then 'transition' of a signal can be done by catalysis of production and degradation.

How to build an universal turing machine out of logic gate? Namely, we can do whatever computation we want. The answer is NAND gate.

- NAND gate

$$X_0 + Y_0 + Z_0 \to X_0 + Y_0 + Z_1$$
$$X_0 + Y_1 + Z_0 \to X_0 + Y_1 + Z_1$$
$$X_1 + Y_0 + Z_0 \to X_1 + Y_0 + Z_1$$
$$X_1 + Y_1 + Z_1 \to X_1 + Y_1 + Z_0$$

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

So Chemical reaction (catalysis) networks are Turing universal!

Cells can perform logic gate computations via catalytic chemical reactions. In fact, they can perform arbitrarily complex computations!

- Points to worry:
  - Signal may get corrupted in a cascade (repeat / restore).
  - Such reactions may be hard to implement (e.g. not trimolecular; need to make them first- or second-order).
  - Reactions are stochastic / discrete / noisy (in fact that's even better . . .).
  - Maybe reactions are more powerful:
    - Use mass–action catalytic reactions themselves as computational units (similar, maybe a bit better).
- WAIT! But how complex can a cell be in this way?
  - Number of catalytic reactions $\approx$ number of enzyme types.
  - This is about 10^3 (e.g. $\sim 5 \times 10^3$ reactions in databases for all known enzymes, $\sim 10^3$ in a human cell).
  - If we need about 2–8 reactions per gate, that gives $\sim 3 \times 10^2$ logic gates.
- Compare:
  - 1972: first pocket calculators, $\sim 3 \times 10^3$ transistors, with about 2–8 transistors per gate (e.g. 4 for a NAND gate) $\Rightarrow$ roughly 4-bit data bandwidth (one decimal digit at a time).
  - Moon landing computers (1970s): $\sim 5 \times 10^3$ gates ($\sim 20\times$ more than a cell in this estimate).
  - Nowadays: microcontroller chips have $\sim 10^6$ gates.
  - Intel laptop chips typically have $\sim 10^9$ gates ($\sim 10^3\times$ more than a cell).

Maybe cells do computations in smarter ways? More efficient computing than logic gates?

Neural networks

In recent years (2010 on wards), artificial neural networks (ANNs) have taken over in the complexity of certain computational tasks, e.g. computer vision (image recognition),

language models and translation, robotics, etc.

ANNs are made more powerful by their *depth*. Instead of just linking computational units in simple cascades, we put them into *layers*, so that the network can achieve an exponential-in-layers growth in representational complexity. The idea is to use layers to build a very complex input–output map that represents the solutions of a computational task, and then let the ANN learn/represent this map.

- The computational units of ANNs are *linear threshold units* (LTUs, perceptrons). For an input vector $x = (x_1, \dots, x_n)$ and a weight vector $w = (w_1, \dots, w_n)$, the output y is

$$y = \begin{cases} 1, & w^\top x = \sum_i w_i x_i > \theta, \\ 0, & w^\top x = \sum_i w_i x_i \leqslant \theta, \end{cases}$$

  where θ is the threshold.

- Example: an LTU network can implement the XOR function $z = \mathrm{XOR}(x, y)$ with two binary inputs $x, y \in \{0, 1\}$. Its truth table is

x	y	z
0	0	0
1	0	1
0	1	1
1	1	0

- How to implement an LTU in cells? We need many inputs to be combined and weighted. A natural candidate mechanism is *gene regulation*: multiple activators and repressors bind to DNA and jointly control the expression level of a gene.

Gene regulatory networks (GRNs) as an implementation:

- **Transcription**

$$G_i \xrightarrow{k_{tx,i}} G_i + T_i,$$

  where G_i is gene i and T_i is its transcript.

- **Translation**

$$T_i \xrightarrow{k_{tl,i}} T_i + R_i,$$

  where R_i is the regulatory species (e.g. a transcription factor).

- **Degradation**

$$T_i \xrightarrow{k_{rd,i}} \varnothing, \qquad R_i \xrightarrow{k_{pd,i}} \varnothing.$$

- **Typical rates**

$$k_{tx,i},\ k_{tl,i} \sim 10^{-3}\ \mathrm{s}^{-1} \quad \text{(per molecule)},$$
$$k_{rd,i},\ k_{pd,i} \sim 10^{-3}\ \mathrm{s}^{-1} \quad \text{(e.g. } \sim 20 \text{ min timescale)}.$$

- **Regulation of transcription** by regulator R_j:

$$R_j + G_i^0 \underset{k_{ji}^-}{\overset{k_{ji}^+}{\rightleftharpoons}} G_{ij},$$

where G_i^0 is the unbound promoter state and G_{ij} the promoter bound by regulator j. Typically

$$k_{ji}^+ \sim \frac{1}{\mathrm{nM} \cdot \mathrm{s}}, \qquad k_{ji}^- = K_M \, k_{ji}^+,$$

with binding constant (dissociation constant) K_M in the range 0.1–$10^5\,\mathrm{nM}$.

- Regulation among transcription factors:

$$R_i + L_j \underset{k_{ij}^-}{\overset{k_{ij}^+}{\rightleftharpoons}} D_{ij},$$

$$D_{ij} \xrightarrow{k_{ij}^{pd}} \varnothing,$$

$$D_{ij} + G_k \underset{k_{ijk}^-}{\overset{k_{ijk}^+}{\rightleftharpoons}} G_{ijk}.$$

These reactions allow transcription factors to form dimers D_{ij} and regulate different genes G_k, effectively changing their transcription rates.

- LTU in titration GRN:

Represent inputs by normalized variables

$$x_j = \frac{I_j}{\bar{I}_j},$$

where I_j is the concentration of input j and $\bar{I}_j$ is a reference value.

Assumptions:

  - I_j act as repressors, so they only affect transcription rates through binding (no extra production term, $k_{tx}^j = 0$).
  - Tight binding between transcription factors R_i and $R_{i'}$, i.e. $K_M^{ii'} \to 0$.
  - Dimers $D_{ii'}$ do not bind promoters G_i or $G_{i'}$, so promoter states with dimers (e.g. $G_{ii'}$, $G_{i'i}$) can be ignored.

Idea: competitive binding among transcription factors R_i, $R_{i'}$ forms a decision unit that behaves like a linear threshold unit.

- Binding equilibrium with inputs:

For repressor I_j binding to promoter G_i, let

$$G_{ji}^c = \text{promoter i bound by input j}.$$

At equilibrium,

$$G_{ji}^{c} = \frac{I_j G_i}{K_M^{ji}}.$$

The total amount of promoter for gene i is

$$\begin{aligned} G_i^{tot} &= G_i^{c} + \sum_j G_{ji}^{c} = G_i + \sum_j \frac{I_j G_i}{K_M^{ji}} \\ &= G_i \left(1 + \sum_j \frac{I_j}{K_M^{ji}} \right). \end{aligned}$$

Hence

$$G_i = \frac{G_i^{tot}}{1 + \sum_j I_j / K_M^{ji}}.$$

- Gene expression of regulator R_i:

  Let

  $$R_{i,tot} = R_i + D_{ii'}$$

  be the total amount of R_i (free plus dimer). Its production–degradation dynamics are

  $$\dot{R}_{i,tot} = k_{tl} T_i - k_{rd} R_{i,tot}.$$

  At steady state,

  $$R_{i,tot} = \frac{k_{tl}}{k_{rd}} T_i.$$

  For the transcript T_i,

  $$\dot{T}_i = g_i k_{tx} G_i - k_{pd} T_i \quad \Rightarrow \quad T_i = \frac{k_{tx}}{k_{pd}} g_i G_i$$

  at steady state. Combining the two expressions gives

  $$R_{i,tot} = \frac{k_{tx} k_{tl}}{k_{pd} k_{rd}} g_i G_i \equiv g_i G_i,$$

  where the prefactor is absorbed into an effective gain g_i (typically of order 1 per copy of G_i^{tot}). Using the expression for G_i,

  $$R_{i,tot} = g_i \frac{G_i^{tot}}{1 + \sum_j I_j / K_M^{ji}}.$$

- Competitive binding / titration between R_i and $R_{i'}$:

  For very tight binding between R_i and $R_{i'}$, the dimer concentration and the remaining free R_i can be approximated by

  $$D_{ii'} = \min\{R_{i,tot}, R_{i',tot}\},$$

and

$$R_i = \begin{cases} 0, & R_{i,\mathrm{tot}} < R_{i',\mathrm{tot}}, \\ R_{i,\mathrm{tot}} - R_{i',\mathrm{tot}}, & \text{otherwise.} \end{cases}$$

This titration mechanism implements a nonlinear, threshold-like dependence of the free regulator R_i on the inputs, which is what allows the GRN to behave like an LTU.

- Decision boundary.

  There are regulators R'_i that can bind to the output gene G_0. If $R'_{i,\mathrm{tot}} > R_{i,\mathrm{tot}}$ (and binding is tight), R'_i represses G_0 and the output is 0. In this case we require

$$\frac{g'_i}{1 + \sum_j I_j / K_M^{ji'}} > \frac{g_i}{1 + \sum_j I_j / K_M^{ji}}$$

  as the condition for output $= 0$.

  Rearranging gives

$$g'_i\Big(1 + \sum_j \frac{I_j}{K_M^{ji'}}\Big) > g_i\Big(1 + \sum_j \frac{I_j}{K_M^{ji}}\Big)$$

$$(g'_i - g_i) > \sum_j I_j \Big(\frac{g'_i}{K_M^{ji'}} - \frac{g_i}{K_M^{ji}}\Big).$$

  Introducing normalized inputs

$$x_j = \frac{I_j}{m},$$

  we obtain

$$\underbrace{\frac{g'_i - g_i}{m}}_{\theta_i} > \sum_j x_j \underbrace{\Big(\frac{g'_i}{K_M^{ji'}} - \frac{g_i}{K_M^{ji}}\Big)}_{w_j}.$$

  Thus, for output 0,

$$\sum_j w_j x_j < \theta_i,$$

  which is exactly the decision rule of a linear threshold unit (LTU).

- That's great: we can build an LTU out of just three genes. Cells can in principle achieve much more efficient computation using neural networks built from such genetic units.

- But wait — the complexity of an ANN scales with the number of *weights*.

  For a gene regulatory network (GRN), the effective number of weights is roughly

$$\#\text{connections in GRN} \approx \#\text{TFs per gene} \times \#\text{genes}.$$

  A rough comparison:

	E. coli	**Yeast**	**Human**
# TFs	~ 300 (7%)	~ 200 (3%)	~ 1800
# TFs per gene	1–2 (0–10)	1–2 (0–20)	10–12 (0–100)
Estimated #weights	$\sim 10^2$	$\sim 10^3$	$\sim 10^4$

For ANNs, the number of weights has grown dramatically:

	1990s	2010s	2020s
Example models	LeNet, MLP on MNIST	AlexNet, VGG-16, ResNet on ImageNet	Transformers (GPT-1–4)
#weights	$\sim 10^3$	$\sim 10^4$–10^6	$\sim 10^7$–10^{10}

So even human cells, with only $\mathcal{O}(10^4)$ effective weights, would correspond to just a small fraction of a classic CNN such as LeNet doing handwritten digit recognition.

Something seems off: cells must be processing information much more efficiently than a naive weight-count comparison with ANNs would suggest.

Side note: binding is much more efficient Binding can implement a rectified–linear (ReLU)–type operation much more efficiently than catalysis or gene expression.

Define

$$\mathrm{ReLU}(x) = \begin{cases} x, & x > 0, \\ 0, & \text{otherwise}. \end{cases}$$

Consider tight binding between two species X and Y:

$$X + Y \rightleftharpoons D.$$

Let X_{tot} and Y_{tot} be their total amounts. Under strong binding, the free X concentration satisfies

$$X_{\mathrm{free}} = \begin{cases} X_{\mathrm{tot}} - Y_{\mathrm{tot}}, & X_{\mathrm{tot}} > Y_{\mathrm{tot}}, \\ 0, & \text{otherwise}, \end{cases}$$

so

$$X_{\mathrm{free}} = f(X_{\mathrm{tot}}, Y_{\mathrm{tot}}) = \mathrm{ReLU}(X_{\mathrm{tot}} - Y_{\mathrm{tot}}).$$

Composition of such functions f can achieve arbitrarily complex behaviour, e.g.

- any piecewise–linear function,
- any logic gate,
- any LTU, etc.

The number of binding reactions in cells is huge. Very roughly:

	E. coli	Yeast	Mammalian
protein–protein	10^4	10^5–10^6	10^6–10^7
enzyme–substrate / cofactor	10^5		
protein–DNA		10^4–10^5	
protein–RNA			
protein–lipid / ion ...			
Total	10^5–10^6	10^6–10^7	10^7–10^8

Computational tasks in Bio vs Engineered

- Logic gates in computers.
  - Task: logical deduction / symbolic computation.
  - Question: do cells need to do logical deduction?
- LTUs in ANNs.
  - Task: represent / fit / learn complex input–output maps.
  - Question: do cells need to do this kind of generic function approximation?
- One task we clearly see cells doing:
  - In response to changing environments, cells perform different actions, often via changes in gene expression.
  - So at least, a cell needs to encode *one* complex input–output map (though not an arbitrary one) that maps environmental conditions to appropriate actions.