Neurodynamics

Week 2 Computational Lab

Problem 1
Part (a)

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

For an ODE of the form:

$$\frac{dy}{dx} + P(x)y = Q(x)$$

Integrating factor! I(x)

The integrating factor is:

$$I(x) = e^{\int P(x)dx}$$

And the resulting solution to the ODE is:

$$y(x) = \frac{1}{I(x)} \int I(x)Q(x)dx$$

To get numerical values, use:

$$\mathrm{V}_m(t) = \begin{cases} 0mV & 0 \leq t < 10sec \\ 30mV & 10 \leq t \end{cases}$$

Problem 1 (b,c)

Part (b):

Definition of tau? (week 2 slides)

Part (c):

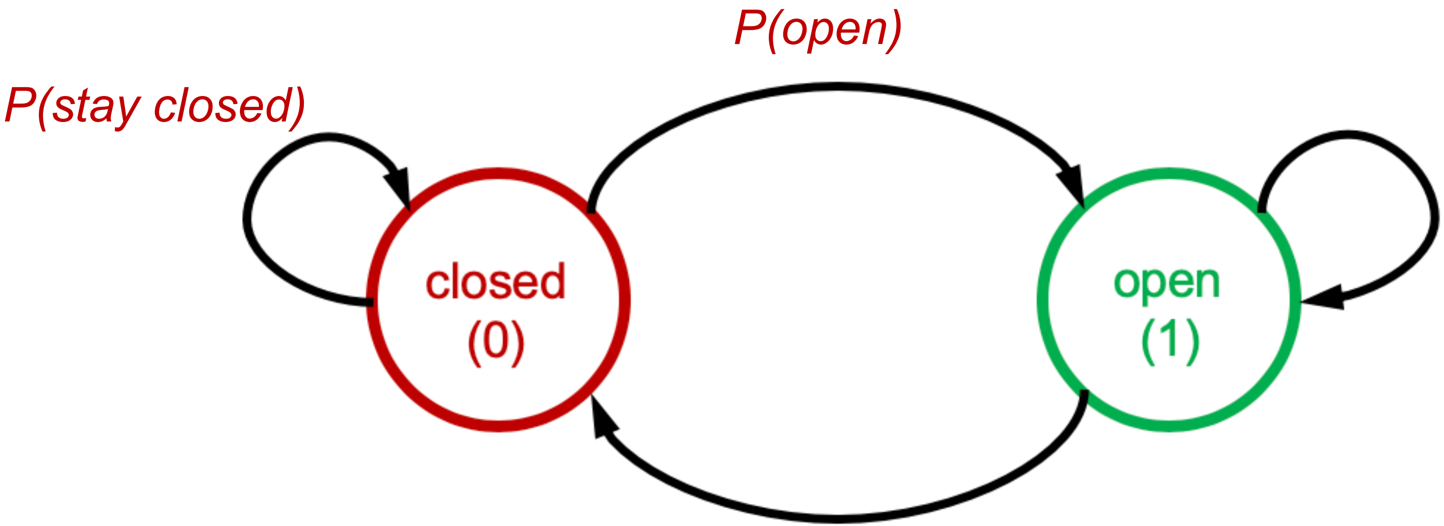
ODE function in language of choce

Problem 1 (d)

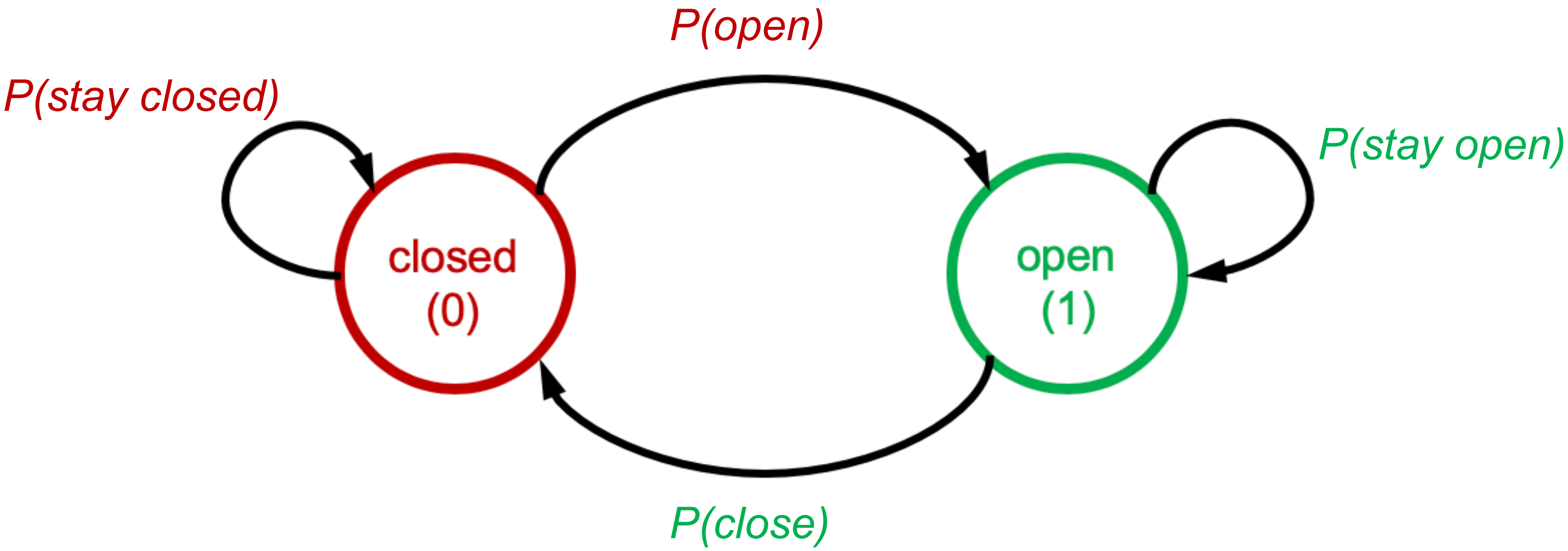
Simulate this Markov process stochastically to find the fraction of gates open, n(t).

$$\mathbf{1-n} \quad \underset{\beta_n(V_m)}{\overset{\alpha_n(V_m)}{\rightleftharpoons}} \quad \mathbf{n}$$

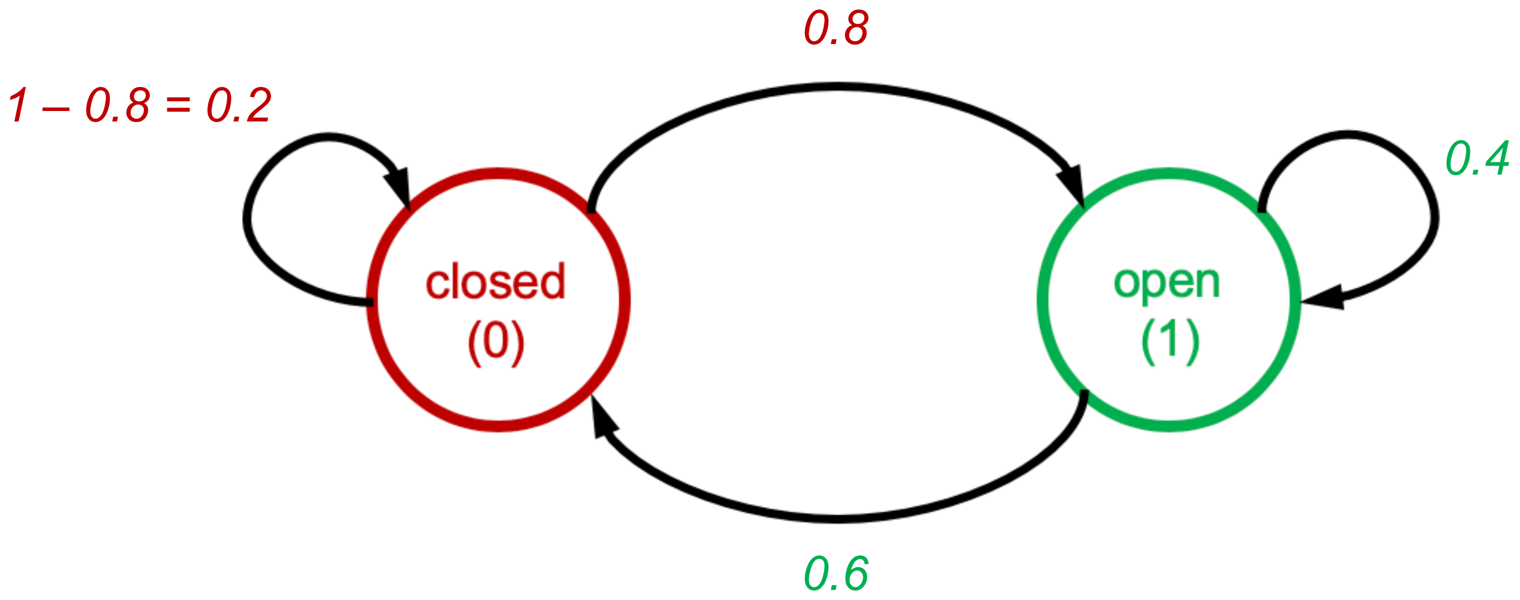
Markov chains



Markov chains



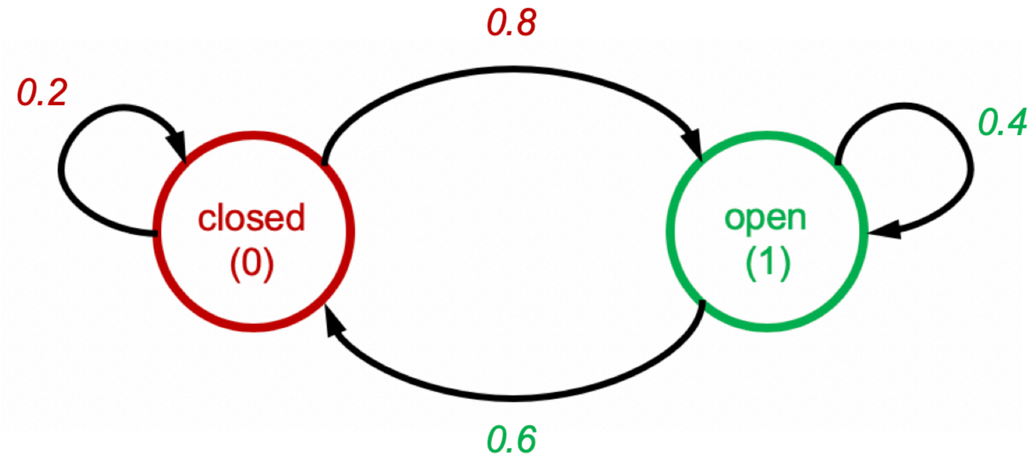
Markov chains

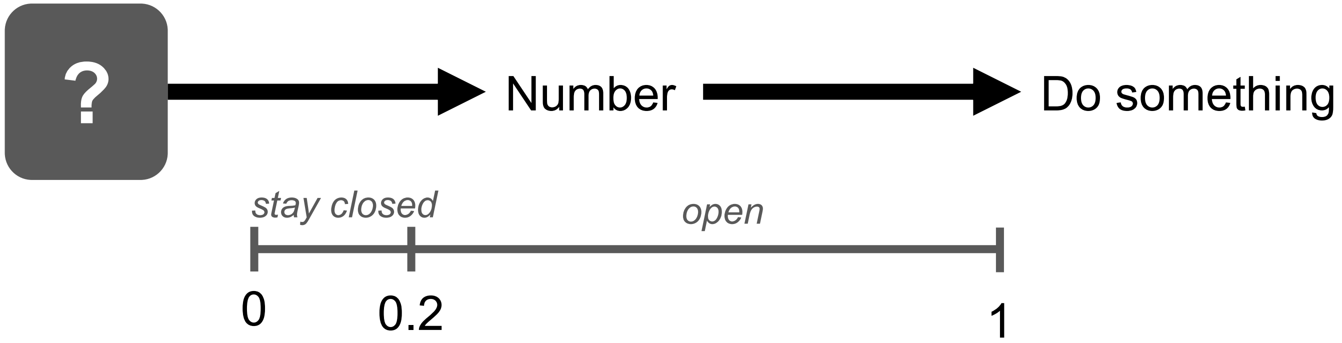


Markov chains

If *state* is closed:

P(open) = 0.8
P(stay closed) = 0.2

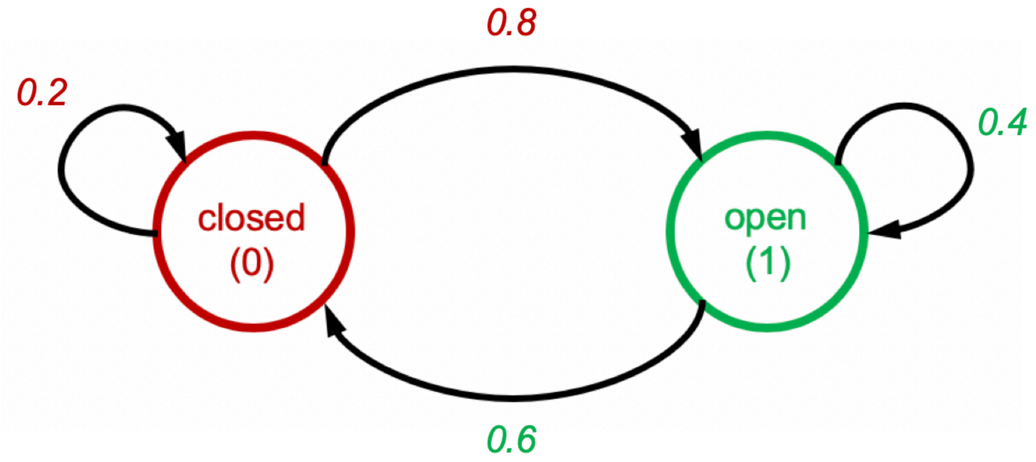


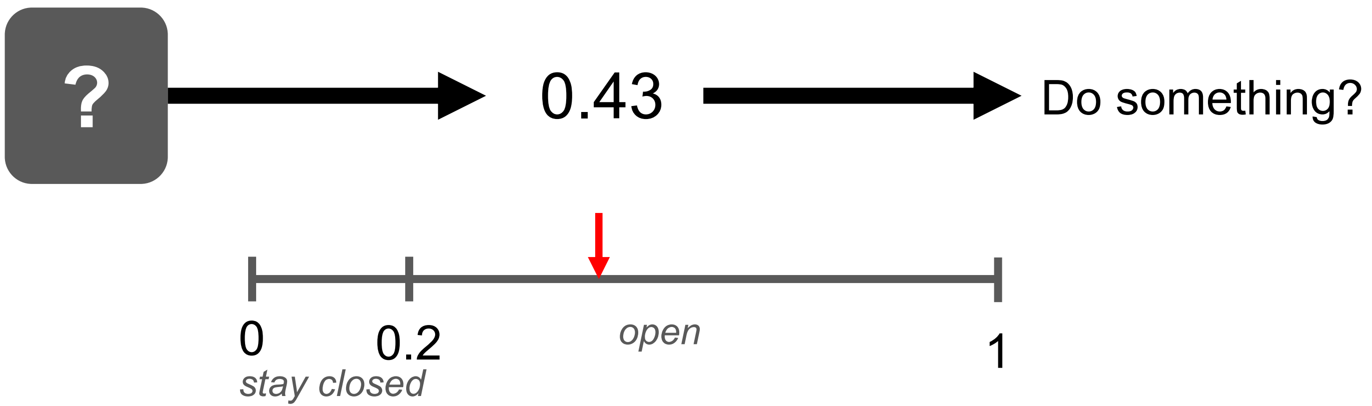


Markov chains

If *state* is closed:

P(open) = 0.8
P(stay closed) = 0.2

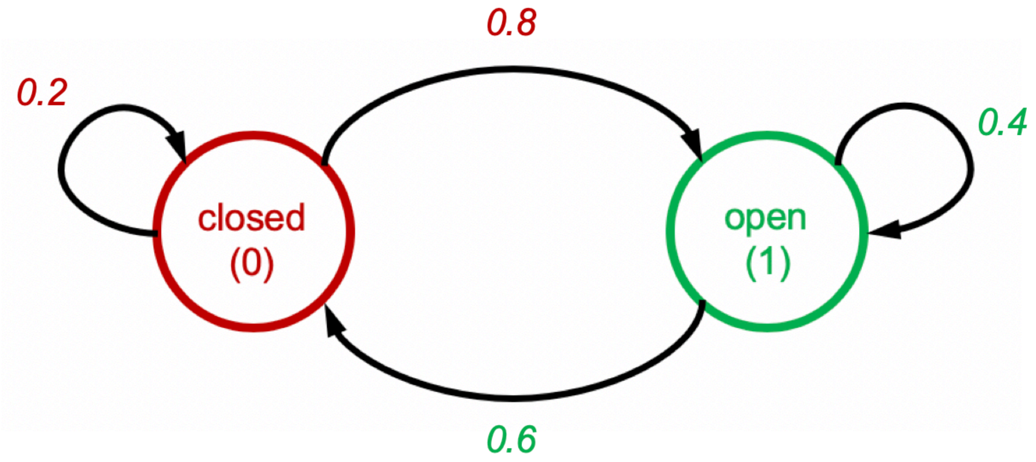


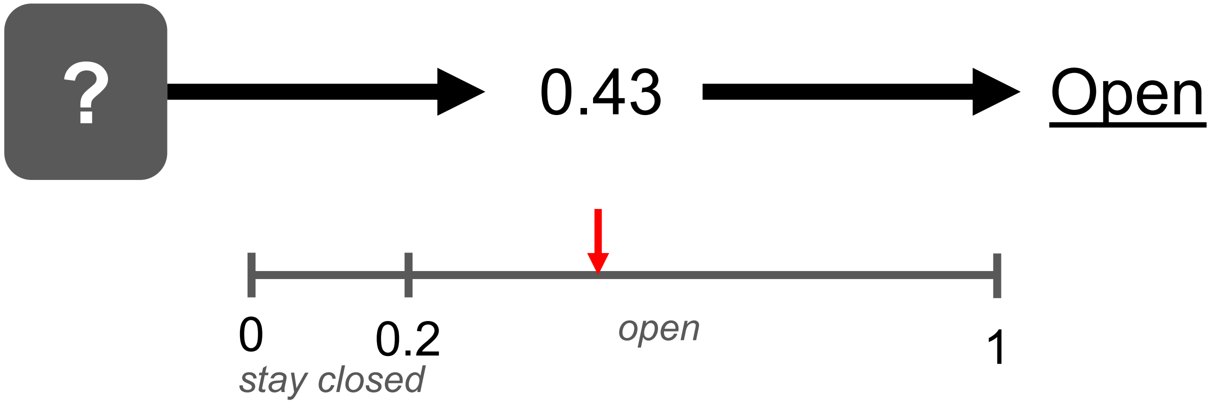


Markov chains

If *state* is closed:

P(open) = 0.8
P(stay closed) = 0.2

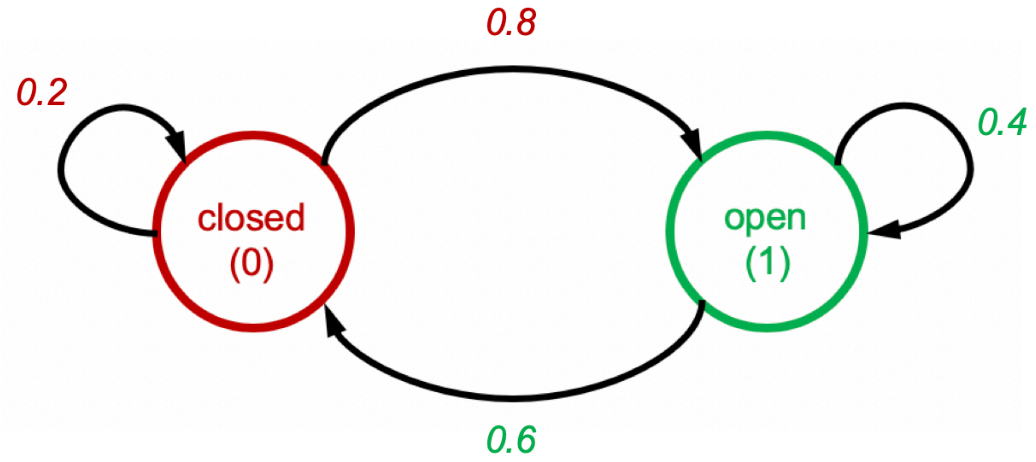


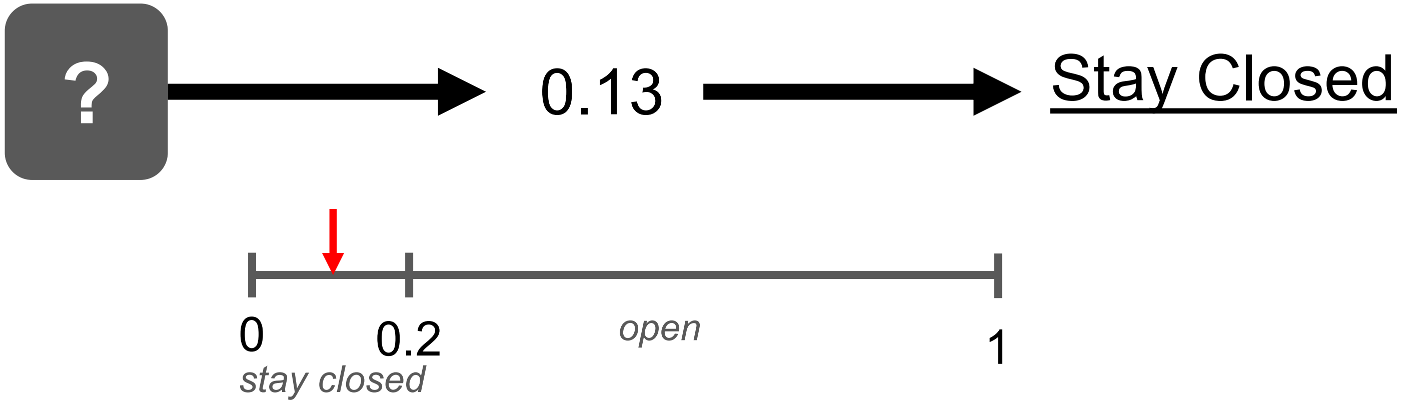


Markov chains

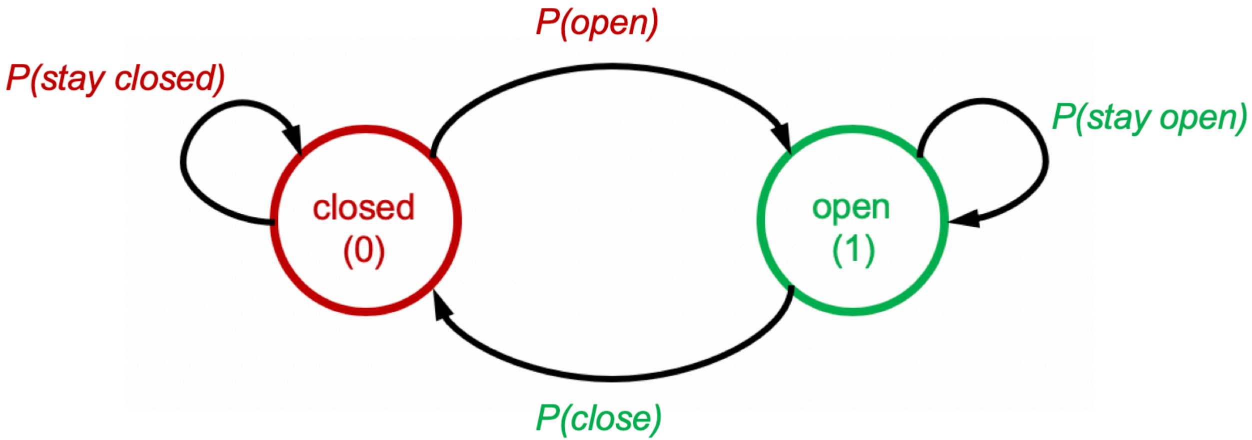
If *state* is closed:

P(open) = 0.8
P(stay closed) = 0.2





Problem 1 (d) – What are the probabilities?



- Opening *rate*: α_n
- Closing *rate*: β_n

•P(open): $\alpha_n \Delta t$

•P(close): $\beta_n \Delta t$

Problem 1(d)

- Assume there are N gates;
- At a short time window Δt, every gate will update its state (from close to open or from open to close or keep its state)
- Calculate the fraction of open gates after time T.

Problem 1 (d)

```
N = 1000                # number of gates

    gate_states = np.zeros(N)          # all gates start closed
    output = []
    for timepoint in t:                # do this for all time points in simulation
        for gate in range(N):          # "throw a dart" for each gate
            r = np.random.rand()       # psuedo-random number generator
            if gate_states[gate] == 0:
                # Probability of transition to open if the gate is closed
                gate_states[gate] = int(r < (t_step * alpha_n(tp)))
            else:
                # Probability gate will stay open if open
                gate_states[gate] = int(r < (1 - t_step * beta_n(tp)))
        output.append(sum(gate_states) * 1.0 / N)
    return output
```

This is the partial codes of this problem !

Problem 2

Example codes should be helpful.

Problem 3(a)

```
p = np.polyfit(n,h,1);
```

h_reg $= \lambda - \mu n.$

Problem 3(a,b)

linear regression

```
p = np.polyfit(n,h,1);
h_reg = ?
```

Problem 3(a,b)

How strong is the relationship?

calculate the correlation coefficient:

```
corrcoef(n,h);
```

Good Luck!