Neurodynamics

Week 2 Computational Lab
Problem 1
Part (a)

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

Integrating factor! I(x)

To get numerical values, use:

\[ V_m(t) = \begin{cases} 
0 \text{mV} & 0 \leq t < 10 \text{sec} \\
30 \text{mV} & 10 \leq t 
\end{cases} \]

For an ODE of the form:

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

The integrating factor is:

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

And the resulting solution to the ODE is:

\[ y(x) = \frac{1}{I(x)} \int I(x)Q(x)dx \]
Problem 1 (b, c)

Part (b):

Definition of $\tau$? (week 2 slides)

Part (c):

ODE function in language of choice
Problem 1 (d)

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

\[
\begin{align*}
1 - n & \quad \Rightarrow \quad n \\
\alpha_n(V_m) & \\
\beta_n(V_m) &
\end{align*}
\]
Markov chains

- Closed (0)
- Open (1)

\[ P(\text{open}) \]

\[ P(\text{stay closed}) \]
Markov chains

- $P(\text{open})$
- $P(\text{close})$
- $P(\text{stay closed})$
- $P(\text{stay open})$
Markov chains

\[ 1 - 0.8 = 0.2 \]
Markov chains

If state is closed:

\[ P(\text{open}) = 0.8 \]
\[ P(\text{stay closed}) = 0.2 \]
Markov chains

If state is closed:

\[ P(\text{open}) = 0.8 \]
\[ P(\text{stay closed}) = 0.2 \]
Markov chains

If state is closed:

\[ P(\text{open}) = 0.8 \]
\[ P(\text{stay closed}) = 0.2 \]
Markov chains

If state is closed:

\[ P(\text{open}) = 0.8 \]
\[ P(\text{stay closed}) = 0.2 \]
Problem 1 (d) – What are the probabilities?

- Opening rate: $\alpha_n$
- Closing rate: $\beta_n$
- $P(\text{open}): \alpha_n \Delta t$
- $P(\text{close}): \beta_n \Delta t$
Problem 1(d)

- Assume there are $N$ gates;

- At a short time window $\Delta t$, 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!
Example codes should be helpful.
Problem 3(a)

\[ p = \text{np.polyfit}(n, h, 1); \]

\[ h_{\text{reg}} = \lambda - \mu n. \]
Problem 3(a,b)

linear regression

\[ p = \text{np.polyfit}(n,h,1); \]
\[ h_{\text{reg}} = ? \]
Problem 3(a,b)

How strong is the relationship?

calculate the correlation coefficient:

corrcoef(n,h);
Good Luck!