Theoretical Concept

Let us examine the three-state continuous-time Markov jump model of the cyclic enzymatic reaction illustrated in Figure 1. In this model, $\boldsymbol{E}$ represents the state "free enzyme," $\boldsymbol{ES}$ denotes "enzyme bound to substrate," and $\boldsymbol{EP}$ indicates "enzyme bound to product." The parameters k_+ and k_- correspond to the clockwise and anticlockwise transition rates, respectively, within the state space.

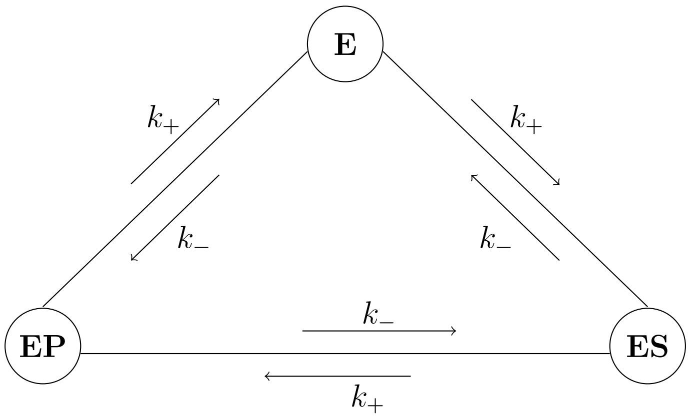

Figure 1: Sketch of the enzymatic reaction

We can derive a differential equation for $P_i(t)$ by relating the probabilities at two closed times t and $t+dt$, and implicitly assume the transition rate from one state to the other during the time interval $(t, t+dt)$ does not depend on the previous history (memoryless property of Markov jumps).

The probability $P_i(t+dt)$ that the particle is in state i at time $t+dt$ is based on two factors:

1. The probability of the particle being in i at time t and not having jumped to any other state $k \neq i$ during the time interval $(t, t+dt)$.
2. The probability of the particle being in a different state $k \neq i$ at time t and making a jump from k to i in the interval $(t, t+dt)$.

This is given as;

$$P_i(t+dt) = P_i(t) \cdot P(\text{staying in } i) + \sum_{k \neq i} P_k(t) \cdot P(\text{jumping from } k \text{ to } i) \tag{1}$$

Thus

$$P_E(t+dt) = P_E(t) \cdot [1 - k_- - k_+] + P_{ES}(t) \cdot k_- + P_{EP}(t) \cdot k_+$$

$$P_{ES}(t+dt) = P_E(t) \cdot k_+ + P_{ES}(t) \cdot [1 - k_+ - k_-] + P_{EP}(t) \cdot k_-$$

$$P_{EP}(t+dt) = P_E(t) \cdot k_- + P_{ES}(t) \cdot k_+ + P_{EP}(t) \cdot [1 - k_+ - k_-]$$

Hence the master equation associated with the dynamics is given as

$$\frac{dP_E(t)}{dt} = -(k_+ + k_-)P_E(t) + k_-P_{ES}(t) + k_+P_{EP}(t)$$
$$\frac{dP_{ES}(t)}{dt} = k_+P_E(t) - (k_+ + k_-)P_{ES}(t) + k_-P_{EP}(t) \tag{16}$$
$$\frac{dP_{EP}(t)}{dt} = k_-P_E(t) + k_+P_{ES}(t) - (k_+ + k_-)P_{EP}(t)$$

This can be expressed in matrix form as;

$$\frac{d}{dt}\begin{bmatrix} P_E \\ P_{ES} \\ P_{EP} \end{bmatrix} = \begin{bmatrix} -(k_+ + k_-) & k_- & k_+ \\ k_+ & -(k_+ + k_-) & k_- \\ k_- & k_+ & -(k_+ + k_-) \end{bmatrix} \begin{bmatrix} P_E \\ P_{ES} \\ P_{EP} \end{bmatrix}$$

The transition matrix $\boldsymbol{W}$ is given as;

$$\boldsymbol{W} = \begin{bmatrix} -(k_+ + k_-) & k_- & k_+ \\ k_+ & -(k_+ + k_-) & k_- \\ k_- & k_+ & -(k_+ + k_-) \end{bmatrix}$$

The eigenvalues (λ) of the transition matrix $\boldsymbol{W}$ is given by solving:

$$\det(\boldsymbol{W} - \lambda \mathbb{1}) = 0$$

Thus

$$\det(\boldsymbol{W} - \lambda \mathbb{1}) = \det\left(\begin{bmatrix} -[(k_+ + k_-) + \lambda] & k_- & k_+ \\ k_+ & -[(k_+ + k_-) + \lambda] & k_- \\ k_- & k_+ & -[(k_+ + k_-) + \lambda] \end{bmatrix}\right) = 0$$

$$\implies -[(k_+ + k_-) + \lambda]\Big[[(k_+ + k_-) + \lambda]^2 - k_+k_-\Big] - k_-\Big[[-k_+(k_+ + k_-) + \lambda] - k_-^2\Big] + k_-k_+[(k_+ + k_-) + \lambda] = 0$$

$$\implies -\Big[(k_+ + k_-)^3 + 3(k_+ + k_-)^2\lambda + 3(k_+ + k_-)\lambda^2 + \lambda^3\Big] + k_+k_-(k_+ + k_-) + k_+k_-\lambda + k_+k_-(k_+ + k_-) + k_+k_-\lambda + k_-^3 + k_+^3 + k_+k_-(k_+ + k_-) + k_+k_-\lambda = 0$$

$$\implies -[k_+^3 + 3k_+^2k_- + 3k_+k_-^2 + k_-^3 + 3k_+^2\lambda + 6k_+k_-\lambda + 3k_-^2\lambda + 3k_+\lambda^2 + 3k_-\lambda^2 + \lambda^3] + 3k_+^2k_- + 3k_+k_-^2 + 3k_+k_-\lambda + k_-^3 + k_+^3 = 0$$

$$\implies -\lambda^3 - 3k_+^2\lambda - 3k_-^2\lambda - 3k_+\lambda^2 - 3k_-\lambda^2 - 3k_+k_-\lambda = 0$$

$$\implies -\lambda^3 - 3(k_+ + k_-)\lambda^2 - 3(k_+^2 + k_-^2 + k_+k_-)\lambda = 0$$

$$\implies -\lambda\Big[\lambda^2 + 3(k_+ + k_-)\lambda + 3(k_+^2 + k_-^2 + k_+k_-)\Big] = 0$$

$$\implies \lambda_1 = 0, \qquad \lambda^2 + 3(k_+ + k_-)\lambda + 3(k_+^2 + k_-^2 + k_+k_-) = 0$$

$$\lambda_{2,3} = \frac{-3(k_+ + k_-) \pm \sqrt{9(k_+ + k_-)^2 - 4(1)\Big[3(k_+^2 + k_-^2 + k_+k_-)\Big]}}{2(1)}$$

$$= \frac{-3(k_+ + k_-) \pm \sqrt{3(-k_+ - k_- - 2k_+k_-)}}{2}$$

$$= \frac{-3(k_+ + k_-) \pm \sqrt{3}\,\sqrt{-(k_+ - k_-)^2}}{2}$$

Thus the eigenvalues are:

$$\lambda_1 = 0$$

$$\lambda_2 = \frac{\sqrt{3}\,\sqrt{-(k_+ - k_-)^2} - 3(k_+ + k_-)}{2} = -\frac{3}{2}(k_+ + k_-) + \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

$$\lambda_3 = \frac{-\sqrt{3}\,\sqrt{-(k_+ - k_-)^2} - 3(k_+ + k_-)}{2} = -\frac{3}{2}(k_+ + k_-) - \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

And the eigenvectors are

$$\vec{V}_{\lambda_1} = \boldsymbol{W} - (0)\mathbb{1} = \vec{0}$$

$$= \left[\begin{array}{ccc|c} -(k_+ + k_-) & k_- & k_+ & 0 \\ k_+ & -(k_+ + k_-) & k_- & 0 \\ k_- & k_+ & -(k_+ + k_-) & 0 \end{array}\right]$$

Thus we have:

$$\begin{cases} -(k_+ + k_-)P_E \; + \; k_-P_{ES} \; + \; k_+P_{EP} = 0 \\ k_+P_E \; - \; (k_+ + k_-)P_{ES} \; + \; k_-P_{EP} = 0 \\ k_-P_E \; + \; k_+P_{ES} \; - \; (k_+ + k_-)P_{EP} = 0 \\ P_E + P_{ES} + P_{EP} = 1 \quad \text{(normalization)} \end{cases} \tag{17}$$

$$\Longrightarrow P_E = 1 - P_{ES} - P_{EP} \tag{18}$$

$$\begin{cases} -(k_+ + k_-) \;+\; (k_+ + k_- + k_-)P_{ES} \;+\; (k_+ + k_- + k_+)P_{EP} = 0 \\ k_+ \;-\; (k_+ + k_+ + k_-)P_{ES} \;+\; (-k_+ + k_-)P_{EP} = 0 \\ k_- \;+\; (-k_- + k_+)P_{ES} \;-\; (k_- + k_+ + k_-)P_{EP} = 0 \end{cases}$$

$$\begin{cases} (k_+ + 2k_-)P_{ES} \;+\; (2k_+ + k_-)P_{EP} = k_+ + k_- \\ -(2k_+ + k_-)P_{ES} \;+\; (-k_+ + k_-)P_{EP} = -k_+ \\ (-k_- + k_+)P_{ES} \;-\; (k_+ + 2k_-)P_{EP} = -k_- \end{cases}$$

Taking

$$\begin{cases} -(2k_+ + k_-)P_{ES} \;+\; (-k_+ + k_-)P_{EP} = -k_+ \\ (-k_- + k_+)P_{ES} \;-\; (k_+ + 2k_-)P_{EP} = -k_- \end{cases}$$

solving simultaneously,

$$\begin{cases} (2k_+ + k_-)(k_+ + 2k_-)P_{ES} \;-\; (k_+ + 2k_-)(-k_+ + k_-)P_{EP} = k_+(k_+ + 2k_-) \\ -(-k_- + k_+)(-k_+ + k_-)P_{ES} \;+\; (-k_+ + k_-)(k_+ + 2k_-)P_{EP} = k_-(-k_+ + k_-) \end{cases}$$

$$\Longrightarrow \Big[(2k_+ + k_-)(k_+ + 2k_-) - (-k_- + k_+)(-k_+ + k_-)\Big]P_{ES} = k_+(k_+ + 2k_-) + k_-(-k_+ + k_-)$$

$$\Longrightarrow \Big[(2k_+^2 + 5k_+k_- + 2k_-^2) - (-k_+^2 + 2k_+k_- - 2k_-^2)\Big]P_{ES} = k_+^2 + k_+k_- + k_-^2$$

$$\Longrightarrow 3\Big[k_+^2 + k_+k_- + k_-^2\Big]P_{ES} = k_+^2 + k_+k_- + k_-^2$$

$$\Longrightarrow \boxed{P_{ES} = \frac{1}{3}}$$

putting it into $(k_+ + 2k_-)P_{ES} \;+\; (2k_+ + k_-)P_{EP} = k_+ + k_-$

$$\Longrightarrow (k_+ + 2k_-)\cdot\frac{1}{3} \;+\; (2k_+ + k_-)P_{EP} = k_+ + k_-$$

$$\Longrightarrow P_{EP} = \frac{k_+ + k_- - \frac{1}{3}\cdot(k_+ + 2k_-)}{2k_+ + k_-} = \frac{3k_+ + 3k_- - k_+ + 2k_-}{3(2k_+ + k_-)} = \frac{2k_+ + k_-}{3(2k_+ + k_-)} = \frac{1}{3}$$

$$\Longrightarrow \boxed{P_{EP} = \frac{1}{3}}$$

$$\text{and from } P_E = 1 - P_{ES} - P_{EP}$$
$$= 1 - \frac{1}{3} - \frac{1}{3} = \frac{1}{3}$$
$$\Longrightarrow \boxed{P_E = \frac{1}{3}}$$

Thus the eigenvector for λ_1 is:

$$\vec{V}_{\lambda_1} = \frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix}$$

Now for the other eigenvectors $\vec{V}_{\lambda_{2,3}}$

$$\vec{V}_{\lambda_{2,3}} = \boldsymbol{W} - \left(-\frac{3}{2}(k_+ + k_-) \pm \frac{\sqrt{3}}{2}(k_+ - k_-)i\right)\mathbb{1} = \vec{0}$$

Thus for $\lambda_{2,3} = -\frac{3}{2}(k_+ + k_-) \pm \frac{\sqrt{3}}{2}(k_+ - k_-)i$,

$$\vec{V}_{\lambda_{2,3}} = \begin{bmatrix} -(k_+ + k_-) - \lambda_{2,3} & k_- & k_+ & | & 0 \\ k_+ & -(k_+ + k_-) - \lambda_{2,3} & k_- & | & 0 \\ k_- & k_+ & -(k_+ + k_-) - \lambda_{2,3} & | & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i & k_- & k_+ & | & 0 \\ k_+ & \frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i & k_- & | & 0 \\ k_- & k_+ & \frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i & | & 0 \end{bmatrix}$$

Thus we have:

$$\begin{cases} \left[\frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i\right] P_E \ + \ k_- P_{ES} \ + \ k_+ P_{EP} = 0 \\ k_+ P_E \ + \ \left[\frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i\right] P_{ES} \ + \ k_- P_{EP} = 0 \\ k_- P_E \ + \ k_+ P_{ES} \ + \ \left[\frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i\right] P_{EP} = 0 \\ P_E + P_{ES} + P_{EP} = 1 \quad \text{(normalization)} \end{cases} \tag{19}$$

$$\Longrightarrow P_E = 1 - P_{ES} - P_{EP} \tag{20}$$

Let $A = \frac{1}{2}(k_+ + k_-) \mp \frac{\sqrt{3}}{2}(k_+ - k_-)i$. Then Equation 19 becomes:

$$\begin{cases} AP_E \ + \ k_-P_{ES} \ + \ k_+P_{EP} = 0 \\ k_+P_E \ + \ AP_{ES} \ + \ k_-P_{EP} = 0 \\ k_-P_E \ + \ k_+P_{ES} \ + \ AP_{EP} = 0 \\ P_E + P_{ES} + P_{EP} = 1 \quad \text{(normalization)} \end{cases}$$

Subtituting *Equation* 20 $\implies \begin{cases} A \ + \ (-A + k_-)P_{ES} \ + \ (-A + k_+)P_{EP} = 0 \\ k_+ \ + \ (-k_+ + A)P_{ES} \ + \ (-k_+ + k_-)P_{EP} = 0 \\ k_- \ + \ (-k_- + k_+)P_{ES} \ + \ (-k_- + A)P_{EP} = 0 \end{cases}$

$$\implies \begin{cases} (k_- - A)P_{ES} \ + \ (k_+ - A)P_{EP} = -A \\ (A - k_+)P_{ES} \ + \ (k_- - k_+)P_{EP} = -k_+ \\ (k_+ - k_-)P_{ES} \ + \ (A - k_-)P_{EP} = -k_- \end{cases}$$

Taking

$$\begin{cases} (A - k_+)P_{ES} \ + \ (k_- - k_+)P_{EP} = -k_+ \\ (k_+ - k_-)P_{ES} \ + \ (A - k_-)P_{EP} = -k_- \end{cases}$$

solving simultaneously,

$$\begin{cases} (k_+ - k_-)(A - k_+)P_{ES} \ + \ (k_+ - k_-)(k_- - k_+)P_{EP} = -(k_+ - k_-)k_+ \\ -(A - k_+)(k_+ - k_-)P_{ES} \ - \ (A - k_+)(A - k_-)P_{EP} = (A - k_+)k_- \end{cases}$$

$$\implies \Big[(k_+ - k_-)(k_- - k_+) - (A - k_+)(A - k_-)\Big]P_{EP} = -(k_+ - k_-)k_+ + (A - k_+)k_-$$

$$\implies \Big[- (k_+ - k_-)^2 - (A - k_+)(A - k_-)\Big]P_{EP} = Ak_- - k_+^2$$

$$\implies \Big[- k_+^2 + k_+k_- - k_-^2 - A^2 + Ak_+ + Ak_-\Big]P_{EP} = Ak_- - k_+^2$$

$$\implies P_{EP} = \frac{Ak_- - k_+^2}{-k_+^2 + k_+k_- - k_-^2 - A^2 + Ak_+ + Ak_-} \tag{21}$$

Hence taking $(k_- - A)P_{ES} \ + \ (k_+ - A)P_{EP} = -A$,

$$\implies P_{ES} = \frac{-A-(k_+-A)P_{EP}}{(k_- - A)} = \frac{-A - \left[\dfrac{(k_+-A)\cdot(Ak_- - k_+^2)}{-k_+^2+k_+k_- - k_-^2 - A^2 + Ak_+ + Ak_-}\right]}{(k_- - A)}$$

$$\implies P_{ES} = \frac{\left[\dfrac{-2Ak_+k_- + Ak_-^2 + A^3 - A^2k_+ + k_+^3}{-k_+^2+k_+k_- - k_-^2 - A^2 + Ak_+ + Ak_-}\right]}{(k_- - A)}$$

$$= -\frac{2Ak_+k_- - Ak_-^2 - A^3 + A^2k_+ - k_+^3}{-k_+^2+k_+k_- - k_-^2 - A^2 + Ak_+ + Ak_-} \cdot \frac{1}{(k_- - A)}$$

$$\implies P_{ES} = -\frac{2Ak_+k_- - Ak_-^2 - A^3 + A^2k_+ - k_+^3}{-k_+^2k_- + k_+^2A + k_+k_-^2 - k_-^3 + 2k_-^2A - 2k_-A^2 + A^3 - k_+A^2} \tag{22}$$

Substituting Equation 21 and Equation 22 into Equation 20,

$$P_E = 1 - P_{ES} - P_{EP}$$

$$= 1 - \frac{Ak_- - k_+^2}{-k_+^2+k_+k_- - k_-^2 - A^2 + Ak_+ + Ak_-} + \frac{2Ak_+k_- - Ak_-^2 - A^3 + A^2k_+ - k_+^3}{-k_+^2k_- + k_+^2A + k_+k_-^2 - k_-^3 + 2k_-^2A - 2k_-A^2 + A^3 - k_+A^2}$$

$$= \frac{k_+k_- - k_-^2 - A^2 + k_+A}{-k_+^2+k_+k_- - k_-^2 - A^2 + k_+A + k_-A} + \frac{2Ak_+k_- - Ak_-^2 - A^3 + A^2k_+ - k_+^3}{-k_+^2k_- + k_+^2A + k_+k_-^2 - k_-^3 + 2k_-^2A - 2k_-A^2 + A^3 - k_+A^2}$$

$$\implies P_E = \frac{k_+k_- - k_-^2 - A^2 + k_+A}{-k_+^2+k_+k_- - k_-^2 - A^2 + k_+A + k_-A} + \frac{2Ak_+k_- - Ak_-^2 - A^3 + A^2k_+ - k_+^3}{-k_+^2k_- + k_+^2A + k_+k_-^2 - k_-^3 + 2k_-^2A - 2k_-A^2 + A^3 - k_+A^2} \tag{23}$$

We can now compute the eigenvectors $\vec{V}_{\lambda_2}$ and $\vec{V}_{\lambda_3}$ from Equation 23, Equation 22 and Equation 21. For $\vec{V}_{\lambda_2}$,

$$A_{\lambda_2} = \frac{1}{2}(k_+ + k_-) - \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

and for $\vec{V}_{\lambda_3}$,

$$A_{\lambda_3} = \frac{1}{2}(k_+ + k_-) + \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

Hence the eigenvalues are:

$$\lambda_1 = 0$$

$$\lambda_2 = -\frac{3}{2}(k_+ + k_-) + \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

$$\lambda_3 = -\frac{3}{2}(k_+ + k_-) - \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

And their respective eigenvectors are:

$$\vec{V}_{\lambda_1} = \frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$\vec{V}_{\lambda_2} = \frac{1}{B_{\lambda_2}}\begin{bmatrix} k_+k_- - k_-^2 - A_{\lambda_2}^2 + k_+A_{\lambda_2} + \dfrac{2A_{\lambda_2}k_+k_- - A_{\lambda_2}k_-^2 - A_{\lambda_2}^3 + A_{\lambda_2}^2k_+ - k_+^3}{K_- - A_{\lambda_2}} \\ -\dfrac{2A_{\lambda_2}k_+k_- - A_{\lambda_2}k_-^2 - A_{\lambda_2}^3 + A_{\lambda_2}^2k_+ - k_+^3}{k_- - A_{\lambda_2}} \\ A_{\lambda_2}k_- - k_+^2 \end{bmatrix}$$

where:

$$A_{\lambda_2} = \frac{1}{2}(k_+ + k_-) - \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

$$B_{\lambda_2} = -k_+^2 + k_+k_- - k_-^2 - A_{\lambda_2}^2 + A_{\lambda_2}k_+ + A_{\lambda_2}k_-$$

$$\vec{V}_{\lambda_3} = \frac{1}{B_{\lambda_3}}\begin{bmatrix} k_+k_- - k_-^2 - A_{\lambda_3}^2 + k_+A_{\lambda_3} + \dfrac{2A_{\lambda_3}k_+k_- - A_{\lambda_3}k_-^2 - A_{\lambda_3}^3 + A_{\lambda_3}^2k_+ - k_+^3}{K_- - A_{\lambda_3}} \\ -\dfrac{2A_{\lambda_3}k_+k_- - A_{\lambda_3}k_-^2 - A_{\lambda_3}^3 + A_{\lambda_3}^2k_+ - k_+^3}{k_- - A_{\lambda_3}} \\ A_{\lambda_3}k_- - k_+^2 \end{bmatrix}$$

where:

$$A_{\lambda_3} = \frac{1}{2}(k_+ + k_-) - \frac{\sqrt{3}}{2}(k_+ - k_-)i$$

$$B_{\lambda_3} = -k_+^2 + k_+k_- - k_-^2 - A_{\lambda_3}^2 + A_{\lambda_3}k_+ + A_{\lambda_3}k_-$$

The stationary distribution of a continuous-time Markov jump is the eigenvector of the transition matrix $\boldsymbol{W}$ with eigenvalue 0.

Thus

$$\lambda_1 = 0; \quad \vec{V}_{\lambda_1} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \Longrightarrow \vec{P}^{st} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \tag{24}$$

Hence,

$$P_E^{st} = \frac{1}{3}; \qquad P_{ES}^{st} = \frac{1}{3}; \qquad P_{EP}^{st} = \frac{1}{3}$$

Now according to Kirchhoff Law, $J^{st}_{E\to ES} = J^{st}_{ES\to EP} = J^{st}_{EP\to E}$ which implies

$$J^{st}_{E\to ES} = W_{ES,E}\cdot P_E^{st} - W_{E,ES}\cdot P_{ES}^{st} = k_+ P_E^{st} - k_- \cdot P_{ES}^{st} = \frac{1}{3}(k_+ - k_-)$$

$$J^{st}_{ES\to EP} = W_{EP,ES}\cdot P_{ES}^{st} - W_{ES,EP}\cdot P_{EP}^{st} = k_+ P_{ES}^{st} - k_- \cdot P_{EP}^{st} = \frac{1}{3}(k_+ - k_-)$$

$$J^{st}_{EP\to E} = W_{E,EP}\cdot P_{EP}^{st} - W_{EP,E}\cdot P_{E}^{st} = k_+ P_{EP}^{st} - k_- \cdot P_{E}^{st} = \frac{1}{3}(k_+ - k_-)$$

Hence the stationary current J^{st} is:

$$J^{st} = J^{st}_{E\to ES} = J^{st}_{ES\to EP} = J^{st}_{EP\to E} = \frac{1}{3}(k_+ - k_-)$$

The characteristic relaxation time of a system describes how quickly the system returns to equilibrium (steady state) after initially out of steady state. It is inversely proportional to the least absolute eigenvalue ($|\lambda_k| > 0$).

Since the eigenvalue $-\frac{3}{2}(k_+ + k_-) + \frac{\sqrt{3}}{2}(k_+ - k_-)i$ is complex, the system will relax towards equilibrium, decaying exponentially with rate $\left|-\frac{3}{2}(k_+ + k_-)\right|$ while the imaginary part determines the frequency of oscillation $\left(\frac{\sqrt{3}}{2}(k_+ - k_-)\right)$.

Thus the system relax towards the steady state with a characteristic relaxation time of

$$\tau = \frac{1}{\left|-\frac{3}{2}(k_+ + k_-)\right|} = \frac{2}{3}(k_+ - k_-)^{-1}s$$

oscillating with a frequency of

$$\frac{\sqrt{3}}{2}(k_+ - k_-)\ rad/s$$

We analyzed the dynamics of the three-state Markov process defined by the transitions between states E, ES, and EP. Using the master equaions as given in Equation 16, we solved for the time-dependent probabilities $\vec{P}(t)$ of being in each state by:

i. Numerically integrating the system of ODEs (Listing 1):

The initial condition assumed was $\vec{P}(0) = \begin{bmatrix} P_E(0) \\ P_{ES}(0) \\ P_{EP}(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$.

The probability values at $t = 30s$, $\vec{P}(30) = \begin{bmatrix} 0.33334151 \\ 0.33339544 \\ 0.33326306 \end{bmatrix}$ which is approximately $\vec{P}^{st}$ (Listing 2).

The numerical integration of the master equation provided smooth curves for the probabilities P_E, P_{ES} and P_{EP} as shown in Figure 2. These curves showed how the system evolves over time, with the probabilities approaching a steady state $\vec{P}^{st}$ (Equation 19) as time t becomes large.

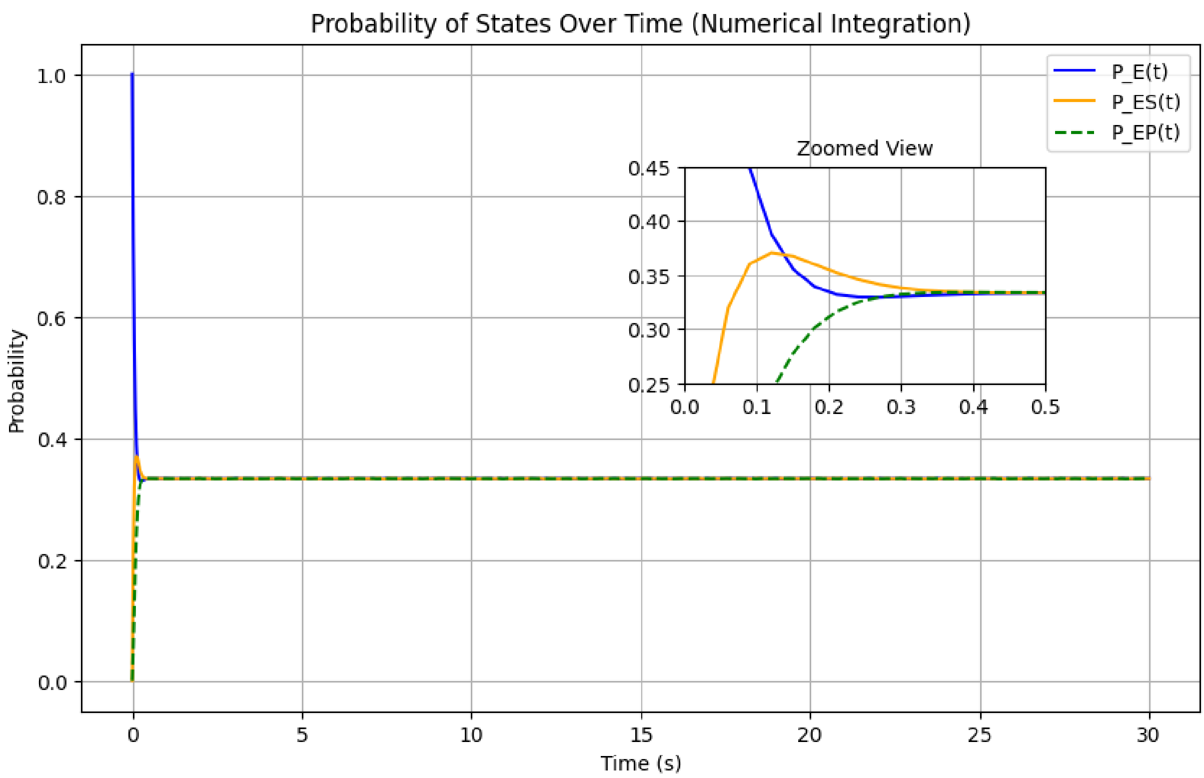

Figure 2: Results for the ODE given by the Master equation. (Listing 3,)

ii. Stochastic simulation with Gillespie algorithm:

We implemented the Gillespie algorithm (Listing 4) on the three-state continuous-time Markov jump model and we ran stochastic 20,000 trajectories to compute the average probabilities of each state over time as shown in Figure 3. I was observed that similarly to the results of the numerical integration of the ODE given by the master equation, the probabilities approaches the steady state $\vec{P}^{st}$ as time t increases.

The probability values at $t = 30s$, $\vec{P}(30) = \begin{bmatrix} 0.3334 \\ 0.3317 \\ 0.3349 \end{bmatrix}$ which is approximately $\vec{P}^{st}$ (Listing 5).

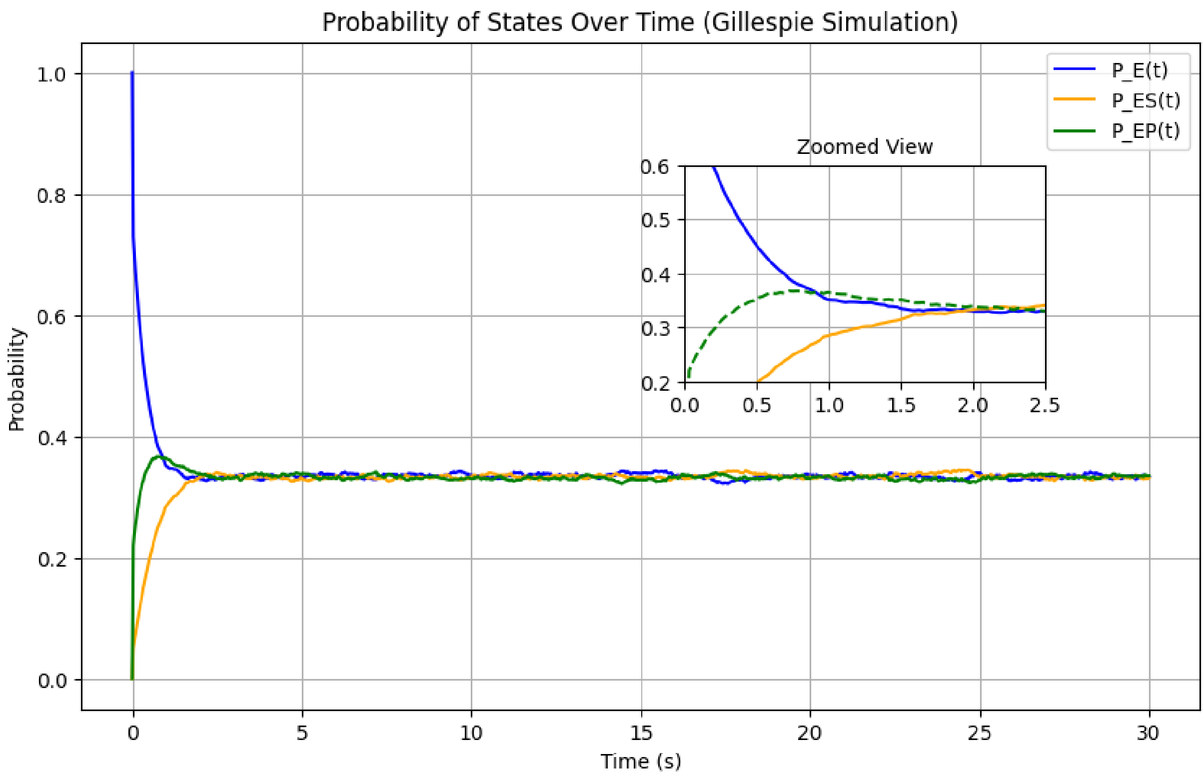

Figure 3: Results for average probabilities from stochastic trajectories obtained using the Gillespie algorithm. (Listing 6,)

Another worth noting point from the comparison was that even though in both methods the probabilities $\vec{P}(t)$ approaches $\vec{P}^{st}$, the zoomed views allowed us to understand that the numerical integration of the ODE given by the Master equation method showed a faster convergence to the steady-state compared to the average probabilities from the stochastic trajectories obtained by using the Gillespie algorithm.

We plotted four stochastic trajectories to highlight the random nature of transitions between states from the Gillespie algorithm method as shown in Figure 4.

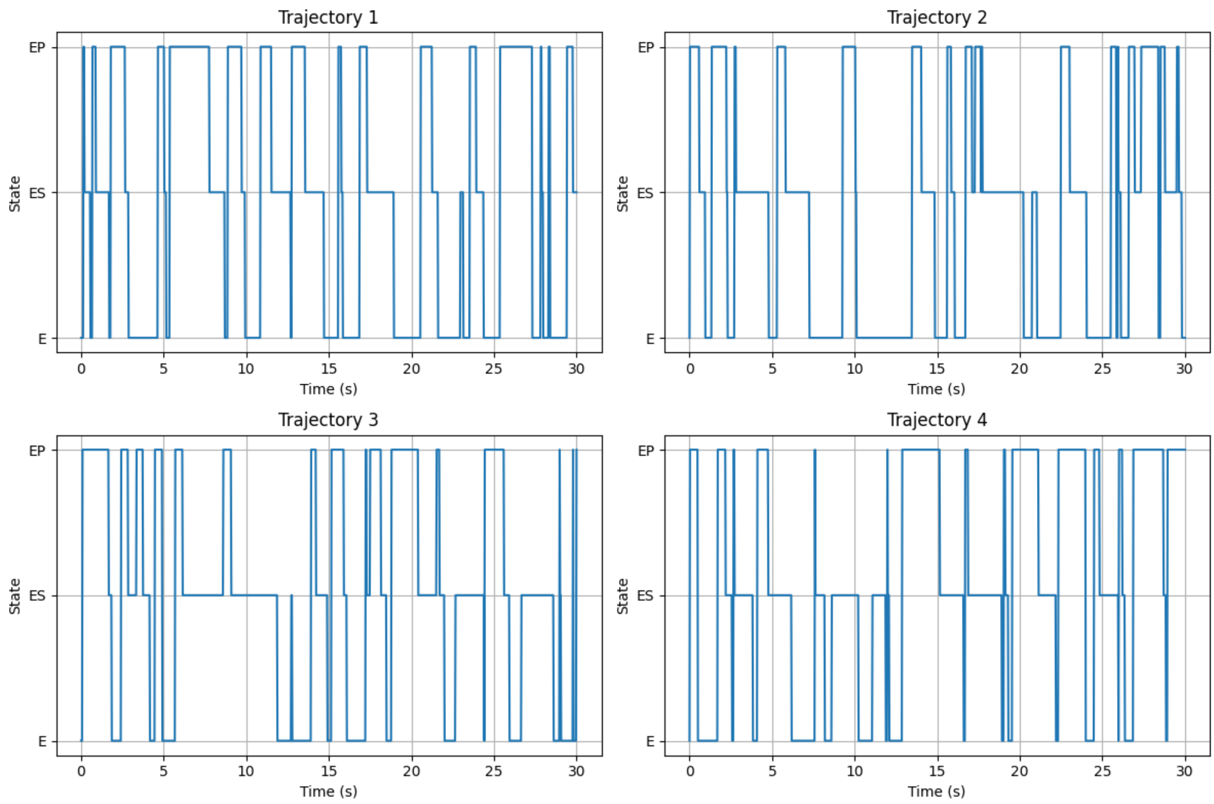

Figure 4: Sample trajectories from the Gillespie algorithm method. (Listing 7 ,)

Appendices

```
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Parameters
k_plus = 10  #forward rate (Hz)
k_minus = 1  #backward rate (Hz)
t_max = 30  #total integration time (s)
t_eval = np.linspace(0, t_max, 1000)  #time points for evaluation

#Master equation as a system of ODEs
def master_eq(t, P):
P_E, P_ES, P_EP = P
dP_E_dt = -(k_plus + k_minus) * P_E + k_minus * P_ES + k_plus * P_EP
dP_ES_dt = k_plus * P_E - (k_plus + k_minus) * P_ES + k_minus * P_EP
dP_EP_dt = k_minus * P_E + k_plus * P_ES - (k_plus + k_minus) * P_EP
return [dP_E_dt, dP_ES_dt, dP_EP_dt]
```

Listing 1: Parameters and Master Equation

```python
#Initial conditions
P0 = [1, 0, 0] #initially all in state E

#Solve the system numerically
solution = solve_ivp(master_eq, [0, t_max], P0, t_eval=t_eval)

#Extract probabilities
P_E, P_ES, P_EP = solution.y

#Print probability values as t = 30s
last_values_NI = solution.y[:, -1]
print(last_values_NI)
```

Listing 2: Solving the initial value problem for system of ODEs given by the Master equation

```python
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import inset_axes, mark_inset

#Plot the results from numerical integration
plt.figure(figsize=(10, 6))
plt.plot(solution.t, P_E, label='P_E(t)', color='blue')
plt.plot(solution.t, P_ES, label='P_ES(t)', color='orange')
plt.plot(solution.t, P_EP, label='P_EP(t)', color='green', linestyle='dashed')
plt.title('Probability of States Over Time (Numerical Integration)')
plt.xlabel('Time (s)')
plt.ylabel('Probability')
plt.legend()
plt.grid()

#Add an inset axis for zoomed-in view
ax_inset = inset_axes(plt.gca(), width=2.5, height=1.5, bbox_to_anchor=(0.8, 0.75), bbox_transform=plt.gcf().transFigure)
ax_inset.plot(solution.t, P_E, color='blue')
ax_inset.plot(solution.t, P_ES, color='orange')
ax_inset.plot(solution.t, P_EP, color='green', linestyle='dashed')
ax_inset.set_xlim(0, 0.5)  # Define x-axis limits for zoom
ax_inset.set_ylim(0.25, 0.45)  # Define y-axis limits for zoom
ax_inset.set_title('Zoomed View', fontsize=10)
ax_inset.grid()

plt.show()
```

Listing 3: Plot for numerical intergration results

```python
#Gillespie algorithm for the three-state Markov process
def gillespie_algorithm(k_plus, k_minus, t_max, num_trajectories):
    states = ['E', 'ES', 'EP']  #E, ES, EP
    num_states = len(states)

    #Rates for each state
    rates = np.array([
        [-k_plus - k_minus, k_minus, k_plus],
        [k_plus, -k_plus - k_minus, k_minus],
        [k_minus, k_plus, -k_plus - k_minus],
    ])

    #Time points and probabilities for averaging
    times = np.linspace(0, t_max, 1000)
    dt = t_max / len(times)
    prob_trajectories = np.zeros((len(times), num_states))

    #Simulate multiple trajectories
    for _ in range(num_trajectories):
        t = 0
        state = 0  #start in state E
        traj_probs = np.zeros((len(times), num_states))
        traj_probs[0, state] = 1  # Initialize at t=0

        for i, t_point in enumerate(times[1:], start=1):
            while t < t_point:
                rates_out = -rates[state, state]
                if rates_out == 0:
                    break
                tau = np.random.exponential(1 / rates_out) #exponential waiting times
                t += tau

                if t >= t_point:
                    break

                #Choose next state based on transition probabilities
                transitions = rates[state, :].copy()
                transitions[state] = 0
                probabilities = transitions / transitions.sum()
                state = np.random.choice(range(num_states), p=probabilities)

            traj_probs[i, state] = 1

        prob_trajectories += traj_probs

    #Average over all trajectories
    avg_probs = prob_trajectories / num_trajectories

    return avg_probs, times, rates
```

Listing 4: Gillespie algorithm

```python
#Parameters for Gillespie algorithm
num_trajectories = 20000

#Run Gillespie simulation
avg_probs, gillespie_times, rates = gillespie_algorithm(
k_plus, k_minus, t_max, num_trajectories)

#Print probability values as t = 30s
last_values_SGA = avg_probs[-1,:]
print(last_values_SGA)
```

Listing 5: Run 20000 simulations on Gillespie Algorithm

```python
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import inset_axes, mark_inset

#Plot average probabilities from Gillespie simulation
plt.figure(figsize=(10, 6))
plt.plot(gillespie_times, avg_probs[:, 0], label='P_E(t)', color='blue')
plt.plot(gillespie_times, avg_probs[:, 1], label='P_ES(t)', color='orange')
plt.plot(gillespie_times, avg_probs[:, 2], label='P_EP(t)', color='green')
plt.title('Probability of States Over Time (Gillespie Simulation)')
plt.xlabel('Time (s)')
plt.ylabel('Probability')
plt.legend()
plt.grid()

#Add an inset axis for zoomed-in view
ax_inset = inset_axes(plt.gca(), width=2.5, height=1.5, bbox_to_anchor=(0.8, 0.75), bbox_transform=plt.gcf().transFigure)
ax_inset.plot(gillespie_times, avg_probs[:, 0], color='blue')
ax_inset.plot(gillespie_times, avg_probs[:, 1], color='orange')
ax_inset.plot(gillespie_times, avg_probs[:, 2], color='green', linestyle='dashed')
ax_inset.set_xlim(0, 2.5)  # Define x-axis limits for zoom
ax_inset.set_ylim(0.2, 0.6)  # Define y-axis limits for zoom
ax_inset.set_title('Zoomed View', fontsize=10)
ax_inset.grid()

plt.show()
```

Listing 6: Plot average probabilities from Gillespie simulation

```python
import matplotlib.pyplot as plt
import numpy as np

#2x2 subplot
fig, axes = plt.subplots(2, 2, figsize=(12, 8))

#Loop through the first 4 trajectories and plot on the respective subplots
for i, ax in enumerate(axes.flat):
    if i < num_trajectories:
        t = 0
        state = 0  #Start in state E
        state_trajectory = []  #store the state at each time step
        state_trajectory.append(state)

        for j, t_point in enumerate(gillespie_times[1:], start=1):
            while t < t_point:
                rates_out = -rates[state, state]
                if rates_out == 0:
                    break
                tau = np.random.exponential(1 / rates_out)
                t += tau

                if t >= t_point:
                    break

                #Choose next state based on transition probabilities
                transitions = rates[state, :].copy()
                transitions[state] = 0
                probabilities = transitions / transitions.sum()
                state = np.random.choice(range(3), p=probabilities)

            #Append the current state at this time point
            state_trajectory.append(state)

        #Plot the state trajectory for the i-th subplot
        ax.plot(gillespie_times, state_trajectory, label=f'Trajectory {i+1}')
        ax.set_title(f'Trajectory {i+1}')
        ax.set_xlabel('Time (s)')
        ax.set_ylabel('State')
        ax.set_yticks([0, 1, 2])  #label for states E, ES, EP
        ax.set_yticklabels(['E', 'ES', 'EP'])
        ax.grid(True)

plt.tight_layout()
plt.show()
```

Listing 7: Plot of sample trajectories

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters

ratio_values = np.linspace(1.1, 10, 100)  # k_+ / k_- ratio, must be > 1

# Compute Q for the ratio values
ln_ratio = np.log(ratio_values)
coth_term = 1 / np.tanh(0.5 * ln_ratio)  # coth(x) = 1 / tanh(x)
Q =  ln_ratio * coth_term

# Mask where Q <= 2
Q_masked = np.where(Q >= 2, Q, np.nan)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(ratio_values, Q, label='Q', color='blue')
plt.axhline(y=2, color='red', linestyle='--', label='Q = 2 (Threshold)')

# Labels and title
plt.xlabel('$k_+ / k_-$')
plt.ylabel('$Q$')
plt.title('Thermodynamic Uncertainty Relationship ($Q$ vs $k_+ / k_-$)')
plt.grid(True)
plt.legend()
plt.show()
```
Listing 8: Thermodynamic uncertainty relation validity check