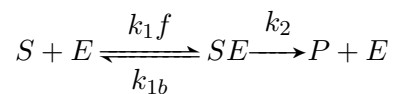


CS341 SYSTEMS BIOLOGY I

LECTURE 5

1. ENZYME KINETICS CONTINUED

Recall the system we are studying. Enzymes catalyze reactions. We are studying $S \rightarrow P$ in which enzyme E either dramatically speeds up that reaction or is necessary for it to occur. We want to know what the relationship is between the rate of accumulation of P and the amount of substrate S . If there were no enzymes present, then it would be a straight line (i.e. it would follow mass action kinetics).



We can model this with 4 equations, but the more equations we have, the more computation is required. As models grow, we want to keep the number of equations small without sacrificing accuracy. We have simulated the kinetics and see that the concentrations of E and SE are nearly at steady-state. That suggests we can remove them.

Using what is called a quasi-steady-state-assumption (that SE and E are both at constant concentrations), it is possible to derive a mathematical expression for $\frac{dP}{dt}$ that is a function of S , but not E or SE (they aren't changing so why model them?). It is

$$v_{max} \frac{S}{K_m + S}$$

where $v_{max} = E(0) \cdot k_2$ and $K_m = \frac{k_{-1} + k_2}{k_1}$.

(See <https://depts.washington.edu/wmatkins/kinetics/michaelis-menten.html> for details on how to derive the expression.)

The kinetics captured by this expression are referred to as Michaelis-Menton kinetics.

Plot the Michaelis-Menton with the `velocity_of_P` curve we computed from the simulation on Tuesday:

```
Km = (k1b + k2)/k1f
vmax = E0*k2
analytic_dPdt = vmax * np.divide( S, (S+Km) )
```

```
plt.plot(S, analytic_dPdt)
plt.plot( S, velocity_of_P)
plt.xlabel( "S")
plt.ylabel("dP/dt")
plt.legend( ["M-M expression", "Simulation"] );
```

We can use the Michaelis-Menton equation to rewrite the model with two ODEs.

$$\begin{aligned}\frac{dS(t)}{dt} &= -v_{max} \frac{S}{K_m + S} \\ \frac{dP(t)}{dt} &= v_{max} \frac{S}{K_m + S}\end{aligned}$$

```
def MM_model( t, y, params ):
    vmax = params[0]
    Km = params[1]
    S = y[0]
    P = y[1]
    dydt = np.zeros( (2,) )
    dydt[0] = -vmax * S/(S+Km)
    dydt[1] = vmax * S/(S+Km)
    return dydt
```

Run the new model, using the parameters and initial conditions from above

```
vmax = E0*k2
Km = (k1b+k2)/k1f

em2 = lambda t,y: MM_model(t,y,(vmax,Km))
sol2 = scipy.integrate.solve_ivp( fun=em2, t_span=(0,100), y0=(S0,P0),
                                  t_eval=np.arange(0,100,1) )
```

Plot the simulation with M-M kinetics and compare it to the output from the original simulation.

```
P_full = sol.y[2,: ]
P_MM = sol2.y[1,: ]
plt.plot( sol.t, P_full )
plt.plot( sol.t, P_MM )
plt.xlabel( "Time")
plt.ylabel( "Concentration" )
plt.legend( ["full model", "M-M model"] )
plt.title( "Product of Enzyme-catalyzed reaction");
```

Now let's spend a couple of minutes getting to understand the effect of KM on dPdt.

[Play the tutorial section 5. They say K_M is constant that describes the dependence of dP/dt on S – its value determines the steepness of the curve.)]

K_M indicates the concentration of S that causes production of P to be at half its maximal rate. Its value thus determines how steep the curve is. Large K_M make it gentler; small K_M make it steeper.

Let's look at K_M :

$$K_M = \frac{k_{1b} + k_2}{k_{1f}}$$

K_M is small when the rate of dimerization k_{1f} is high.

These kinetics cover most of the processes we are interested in

- zeroeth order (mass action) – basal rate of transcription.
- first order (mass action) – transport, non-specific degradation, undimerization
- second order (mass action) – dimerization
- Michaelis-Menten – enzyme-mediated processes with few enzymes, which may include phosphorylation and targeted degradation

But there is one thing missing – regulation of transcription. Genes are often turned “on” or “off”. So, we need a sigmoidal curve. We will talk about that on Tuesday.