

CS341 Systems Biology I

Lecture 4

Enzyme Kinetics

Questions from HW3

- Q: What is xinit for? A: it's a typo

```
# initial concentrations (M, R, B)
xinit = np.array([0, 1.5, 0.1]);

# parameters
k = 0.05;
params = np.array([k]);

# Simulate it at specific time steps
yinit = np.array([0,1.5,0.1])
sol = scipy.integrate.solve_ivp(fun=dimer_model, t_span=(0,100), y0=yinit, method='RK45',
                                t_eval=np.arange(0,100,0.1), args=[params] )
```

Q: When do you need to use solve_ivp?

A: Any time you need to run a simulation. (e.g. any time you change a parameter value or an initial condition)

- Q: Why do we always need to include `t_span` and `t_eval`?

A: `solve_ivp` need to be told when the simulation should begin and end (`t_span`), but it is most efficient when it can make its own decisions about how large the time steps should be. I want to have the simulation always at a certain set of time steps so that I can compare simulations to each other, so I mandate which timesteps it should give me output for.

- Q: What is the loop for? What are the linespecs?

```
plt.figure(figsize=(8,4));  
linespecs = ['m','r','b'];  
for i in range(sol.y.shape[0]):  
    plt.plot( sol.t, sol.y[i,:], linespecs[i] );
```

A: I wanted to plot each line using the colors of the animation (magenta dimers, red proteins, blue proteins). “m”, “r”, and “b” are the colors to use for each line. I loop through the state variables and plot each one with the appropriate color for the line.

- Q: Can you talk more about this line of code?

```
dydt = np.zeros((3,))
```

A: `np.zeros` makes a `ndarray` of zeros. In general, if we know how big an `ndarray` is going to be, it is more efficient to allocate all the space for it at the beginning, then set elements. That is what we do here.

`(3,)` is a 1-dimensional tuple with the first entry of 3. We use this notation because `numpy` is expecting a tuple (not a single) number as the dimensionality of the `ndarray`.

- Q: What is RK45?

```
sol = scipy.integrate.solve_ivp(fun=dimer_model, t_span=(0,100), y0=yinit, method='RK45',  
                                t_eval=np.arange(0,100,0.1), args=[params])
```

A: It is the algorithm used to solve the ordinary differential equations. It is from a family of methods calls Runge-Kutta methods. It uses a 4th order (high precision) and a 5th order (higher precision) together, using one to find the solution and the other to estimate the error (if error is too big, then next step will be smaller, if error is tiny, then the next step can be bigger)

- Q: Is it possible to have three proteins “dimerize”?

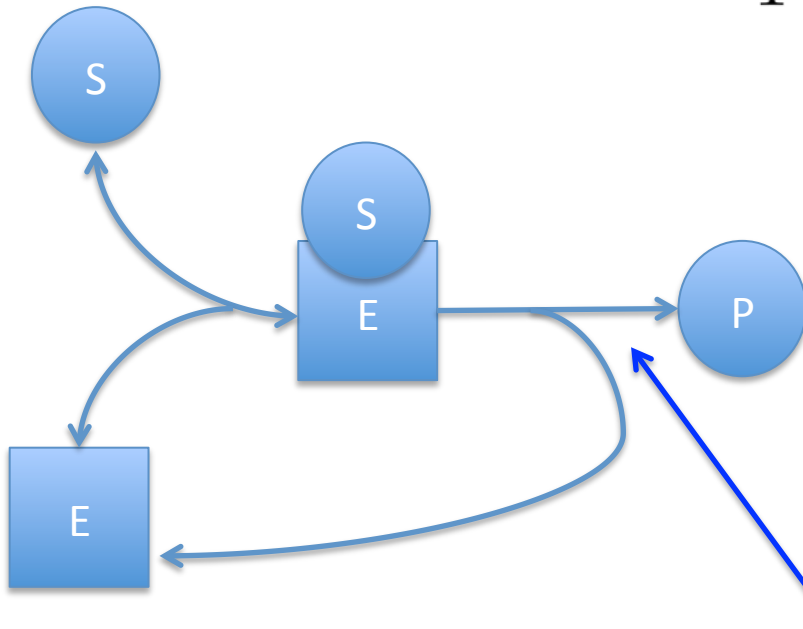
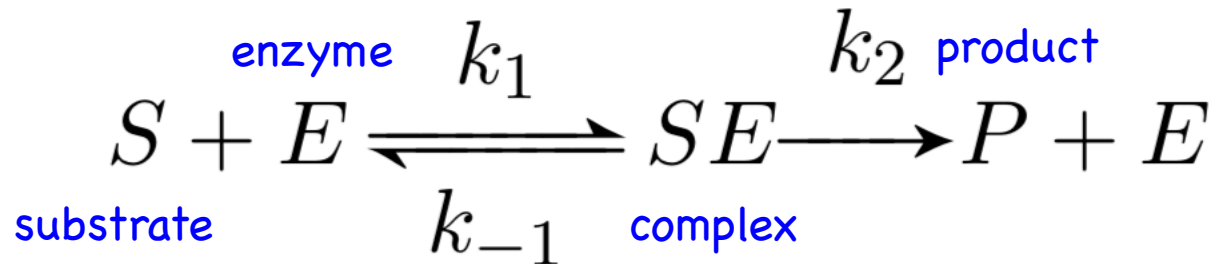
A: yes, but we would call it a trimer. The word for any number of proteins bound together is “complex”

Enzyme Kinetics

- We will watch sections 1-5 of a tutorial:
http://www.wiley.com/college/pratt/0471393878/student/animations/enzyme_kinetics/index.html
- If the reaction $S \rightarrow P$ (substrate is transformed into product) is aided by an enzyme, then
$$dP/dt = k \cdot S$$

is not the appropriate model

The complete model includes the enzyme as a reactant (a variable that is part of a reaction)



$$\begin{aligned}
 \frac{dS}{dt} &= -k_1 \cdot S \cdot E + k_{-1} \cdot SE \\
 \frac{dE}{dt} &= -k_1 S \cdot E + k_{-1} SE + k_2 \cdot SE \\
 \frac{dP}{dt} &= k_2 \cdot SE \\
 \frac{dSE}{dt} &= k_1 \cdot S \cdot E - k_{-1} \cdot SE - k_2 \cdot SE
 \end{aligned}$$

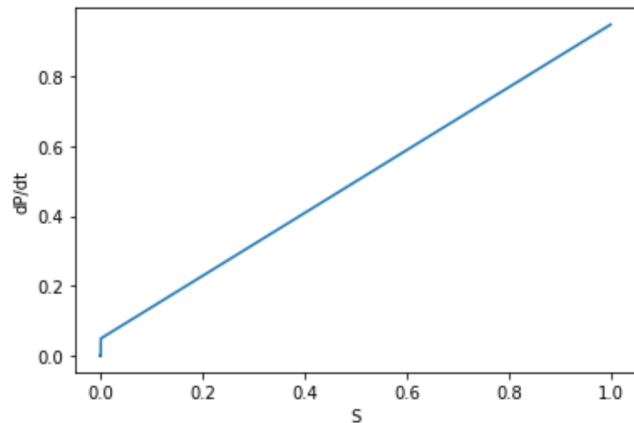
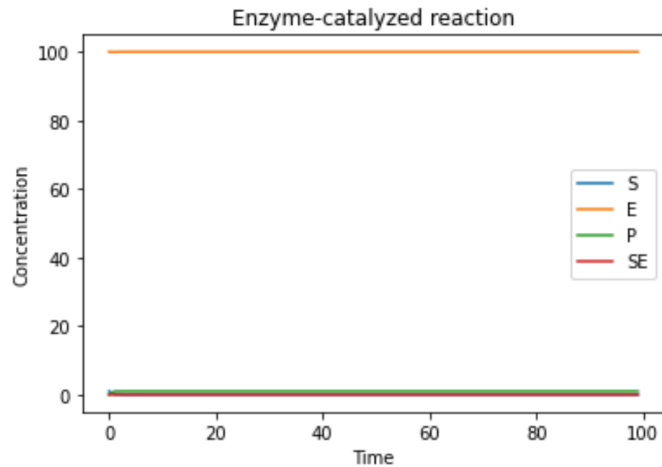
the process we are most interested in simulating

Goal: understand dynamics of full system

But then make smaller model that does not need ODEs for E or SE

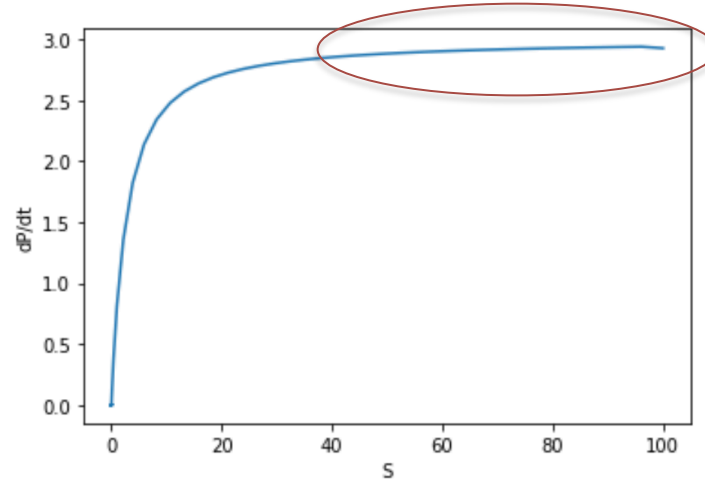
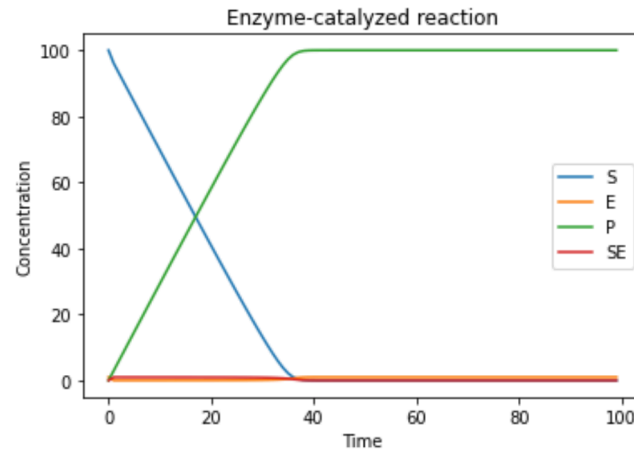
i.e. find expression for dP/dt that depends on S instead of on SE.

Much less substrate than enzymes
(rarely seen in biology)



$dP/dt = kS$
except when $S \sim E$

Not very many enzymes
(common in biology)



$dP/dt = c$
would be
appropriate

To capture entire curve, we either need full model
or Michaelis-Menton kinetics
 $dP/dt = v_{max} * S/(S+K)$