Biomeng 261: Lab 1 (2022)

Oliver Maclaren
oliver.maclaren@auckland.ac.nz

Due: **Monday 8 August 11.59pm** (*online only!*)

Create an **answer document** (e.g. in Word) and write your answers in that as you go. Please put your **name** and **ID** at the top of this document.

This lab is assessed individually, with **2 Parts worth 10 marks each** for a total of **20 marks**. These marks will be converted to **2.5% of your grade for the course**.

Part I: Simulating ODEs [10 marks total]

In lectures we discussed the enzyme-catalysed reaction

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \xrightarrow{k_2} P + E$$

where E is the enzyme and C is the enzyme-substrate complex formed when the substrate S binds to the enzyme's active site, forming product P. Using conservation of mass and the law of mass action we derived

$$\begin{aligned}
\frac{d[S]}{dt} &= -J_1 + J_{-1} \\
\frac{d[E]}{dt} &= -J_1 + J_{-1} + J_2 \\
\frac{d[C]}{dt} &= J_1 - J_{-1} - J_2 \\
\frac{d[P]}{dt} &= J_2
\end{aligned}$$

where

$$\begin{aligned}
J_1 &= k_1[S][E] \\
J_{-1} &= k_{-1}[C] \\
J_2 &= k_2[C]
\end{aligned}$$

In laboratory enzyme kinetics experiments, the rate of reaction is measured for different initial amounts of substrate and enzyme. Initial conditions appropriate for modeling these experiments are

$$[S](0) = S_0, \quad [E](0) = E_0, \quad [C](0) = 0, \quad [P](0) = 0.$$

Using the quasi-steady state *approximation*, we derived the Michaelis-Menten rate equation

$$v(t) = \frac{d[P]}{dt} = \frac{V_{\max}[S]}{K_M + [S]}$$

where

$$V_{\max} = k_2 E_0,$$

and K_M, the Michaelis-Menten constant, is given by

$$K_M = \frac{k_{-1} + k_2}{k_1}.$$

We are going to simulate this system with the following parameter set and initial conditions:

- $k_1 = 10\ \mu M^{-1}s^{-1}$, $k_{-1} = 19\ s^{-1}$, $k_2 = 11\ s^{-1}$.
- Initial conditions: $[S](0) = 16\ \mu M$, $[E](0) = 1\ \mu M$, all other species set to 0 μM.

The MATLAB script **Biomeng261_L1_Part1.m** (download from Canvas) integrates the ODEs to provide solutions $[S], [E], [C]$ and $[P]$ as functions of time. It also produces plots of $[S]$ and $[E]$. There is also a Python version if you want to use that.

In this lab you will use and modify the MATLAB/Python script to explore the concepts of enzyme kinetic models covered in the lectures.

Ia. Getting familiar with the script

- Check that script runs and provides a graph of $[S]$ and $[E]$. Check that workspace has variables for each species. What are the end values of $[S], [C], [E], [P]$?

[1 mark]

- Modify the script to also plot $[C]$ and $[P]$. Make sure to include the correct labels. **Paste these plots into your answer document**.

[1 marks]

- Have the results reached steady state? If not modify the script to change the integration time and provide steady state values. State your integration time, the steady state values and **paste a figure into your answer document that justifies your answer**.

[1 marks]

- Does solution show that $[E] + [C] = E_0$? **Create a plot to support your answer**.

[1 mark]

Ib. Exploring assumptions

- Do results indicate that $[C]$ reaches quasi-steady state during the simulations (with your new time range from the previous question). If so, how quickly? **Justify with a plot**. (Hints: plot the RHS of two of the ODEs on the same graph and compare. You will need to use MATLAB's '.' operator to multiply vectors element-wise. Remember to label your plots!)

[2 marks]

- Do results agree with Michaelis-Menten kinetics? For all times? **Justify with a plot**. (Hint: you should compare an expression involving $[C]$ with an expression involving $[S]$. Plot both on the same graph and remember to use '.' whenever multiplying or dividing vectors by vectors!)

[2 marks]

- Repeat the above two questions with $E_0 = 10\ \mu M$ (instead of $E_0 = 1\ \mu M$). **Include the two modified plots and comment on the difference**. What important dimensionless parameter has changed?

[2 marks]

Part II [10 marks total]. Hill curves

Recall from lectures that the Hill curve relates how many apparent binding sites n an enzyme has to the reaction velocity v

$$v = \frac{V_{\max}[S]^n}{K_d + [S]^n}$$

where here we denote $K_d = K_M^n$, i.e. for this term we have lumped the power into the definition of K_d.

The following Table shows some v and S data for an enzyme known to have **more than one cooperative binding site**. The $V_{\max}$ is known to be 30 $\mu M/s$ and the K_d is known to be 60 μM^n.

$[S]$ (μM)	v $(\mu M/s)$
0.5	0.07
1.0	0.47
2.0	2.76
3.0	6.74
4.0	11.4
6.0	19.1
10	26.2
20	29.1
30	30.0

What we don't know is how many binding sites n it has.

Create a new script to answer the following questions.

IIa. Plotting

- Plot the data points above (put reaction velocity v on the y-axis and $[S]$ on the x-axis, and use scatter) with appropriate axis ranges etc. Note the shape of the implied curve. **Paste the resulting figure into your answer document**.

[2 marks]

Hint: modifying the following (MATLAB) commands might be helpful…

```
plot(x,y,'ko');
xlabel('x');
ylabel('y');
title('my cool plot')
```

Python has similar plotting methods e.g.

```
import matplotlib.pyplot as plt
plt.plot(x,y,'ko');
plt.xlabel('x');
plt.ylabel('y');
plt.title('my cool plot')
```

IIb. Curve fitting

We are going to try fitting a curve to these data, where the curve is defined by the Hill function above.

First, let's define a function that takes a vector of $[S]$ values, a single n and returns a predicted velocity $v_{\text{predicted}}$. (We will use the $V_{\max}$ and K_d given above.). We use MATLAB's '.' operator to do vector operations. If you want to use Python then you should use numpy.

To do this, copy the following function definition into your new script (note: for MATLAB you probably need to put it at the very end of your script, after any other commands you add):

```
function v_hill = hill(S,n)
    Kd=60;
    Vmax=30;
    v_hill = Vmax*S.^n./(Kd+S.^n);
end
```

- Evaluate this function for $n = 2$ and the given $[S]$ data above. Paste the resulting set of values into your answer document.

[2 marks]

- Next write a simple for loop to repeat the above for $n = 1, 2, ..., 5$ and plot all corresponding **predicted** v values on the same graph. Then add the **measured** v values to the graph too. Use lines for predicted values and markers only for the measured values. **Paste this figure into your answer document** and state which integer n value you think appears to fit best. Do you think a non-integer n would fit better? Guess a reasonable value.

[4 marks]

We can try to **quantify** our judgement of distance or discrepancy between predicted and observed data as follows. A simple distance can be defined using the (Euclidean) norm of the difference of two vectors, e.g.

```
X = [-2 3 -1];
Y = [1 4 5];
distance = norm(X-Y)
```

Here the default norm MATLAB uses is the so-called Euclidean norm (which equals the square-root of the sum of squared differences between the elements of the vectors).

- For each integer n in the above loop, include a calculation of the **distance** (or error) between the predicted and the measured v vectors. **Paste the differences between predicted and measured** v vectors for $n = 1, ..., 5$ into your answer document. Do these agree with your intuition and visual inspection?

[2 marks]