Biomeng 261: Lab 2 (2024)

Michael Hoffman
michael.hoffman@auckland.ac.nz

Due: **Friday 9 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 three parts worth 10 marks each, for a total of 30 marks. These marks will be converted to **2.5% of your grade for the course**.

Part I: Simulating ODEs continued

(a) Consider the system of reactions representing an enzyme with two binding sites:

$$S + E \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C_1 \xrightarrow{k_2} P + E$$

$$S + C_1 \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} C_2 \xrightarrow{k_4} P + C_1$$

The equations governing this system are

$$\begin{aligned}
\frac{d[S]}{dt} &= -J_1 + J_{-1} - J_3 + J_{-3} \\
\frac{d[E]}{dt} &= -J_1 + J_{-1} + J_2 \\
\frac{d[C_1]}{dt} &= J_1 - J_{-1} - J_2 - J_3 + J_{-3} + J_4 \\
\frac{d[C_2]}{dt} &= J_3 - J_{-3} - J_4 \\
\frac{d[P]}{dt} &= J_2 + J_4
\end{aligned}$$

where

$$\begin{aligned}
J_1 &= k_1[S][E] \\
J_{-1} &= k_{-1}[C_1] \\
J_2 &= k_2[C_1] \\
J_3 &= k_3[S][C_1] \\
J_{-3} &= k_{-3}[C_2] \\
J_4 &= k_4[C_2]
\end{aligned}$$

- Implement this model in Python or Matlab (you can modify the code defining the ODE in Lab 1 or write it from scratch).
- Next construct a script to solve the ODE above using the following settings:
  - Initial conditions: [S] = 15 μM, [E] = 0.5 μM, all other species set to 0 μM.
  - $k_1 = 2\ \mu M^{-1}s^{-1}$, $k_{-1} = 11\ s^{-1}$, $k_2 = 5\ s^{-1}$, $k_3 = 9\ \mu M^{-1}s^{-1}$, $k_{-3} = 21\ s^{-1}$, and $k_4 = 11\ s^{-1}$.

- Once constructed, run your script in order to simulate the system for a period of time such that it reaches steady state.
- Plot all concentrations, either on the same figure or on separate figures, over this time period such that the shapes of the time courses, from start to steady state, are clearly shown. Make sure to include appropriate legends/titles and label your axes.
- Put your figure(s) into your answer document.

The above question is worth **10 marks** and will be based on your figure(s). You should **include your code as an appendix**, however, so that you may be awarded partial marks if needed. And to show that you did it yourself...

Part II: Estimating parameters

This question involves the **same ODE system as the previous question**.

Suppose we had a vector of experimental data $[P]_{\text{data}}$ of product concentrations at the times (in s)

$$\text{times (s)} = [0, 1, 2, ..., 10].$$

The $[P]_{\text{data}}$ is given by

$$[P]_{\text{data}}\ (\mu M) = [0.26, 4.92, 9.53, 11.56, 14.04, 14.04, 14.18, 14.61, 14.62, 15.24, 15.62].$$

There is a small amount of measurement noise in this data.

We happen to know that the true system for which the data was collected can be described by the above model, with all the parameters the same as in the previous question, **except that k_3 is different**. We know that the k_3 for this system is somewhere between 10 and 50 $\mu M^{-1}s^{-1}$, but unfortunately we don't know the exact value and need to estimate it based on $[P]_{\text{data}}$. We can do this as follows.

Firstly, notice that for any k_3 we can simulate our system to get $[P]_{\text{sim}}$ and compare the result to the given data $[P]_{\text{data}}$. We can hence define an 'objective function' to measure how well a given k_3 can 'fit' the given $[P]_{\text{data}}$ via its implied $[P]_{\text{sim}}$ values. Here is a Python version of a possible objective function (it calculates the sum of squared differences between observed and simulated data at the given time points):

```
import numpy as np
def objective(k3):
    P_data = np.array([ 0.26,  4.92,  9.53, 11.56, 14.04, 14.04, \
        14.18, 14.61, 14.62, 15.24, 15.62])
    t_data = np.linspace(0,10,11)
    prm[3] = k3
    dydt = lambda t, y: dydt_p(t,y,prm)
    sol = solve_ivp(fun=dydt,t_span=[0,10],t_eval=t_data,y0=y0)
    P_sim = sol.y[4,:]
    return np.linalg.norm(P_sim-P_data,2)**2
```

In Matlab this would be something like:

```
function obj = objective(k3)
    P_data = [0.26,  4.92,  9.53, 11.56, 14.04, 14.04, 14.18, 14.61, 14.62, 15.24, 15.62];
    t_data = linspace(0,10,11);
    prm(4) = k3
    sol=ode15s(@lab2ode,[0 10],ics,options,prm);
    X = deval(sol,t_data)
    P_sim=X(5,:)
```

```
    obj = norm(P_sim-P_data)^2;
end
```

Note that I've left out some variables like initial conditions etc that you'd need to define within, or provide externally to, the objective function to simulate the ODE.

Given all of the required information, our goal is to minimise the difference between the model predictions and the given data, by varying k_3 and minimising the above objective.

- Evaluate the above objective function over a grid of 100 equally-spaced values of k_3 between 10 and 50, and plot the result. By looking at the graph, state what you think would be a reasonable guess for the k_3 that generated the data.

(5 marks)

We can also use Matlab/Python's optimisation tools to minimise the above function to get a more precise estimate. Using Python we can do:

```
from scipy.optimize import minimize
k3_initial_guess = 1
res = minimize(objective, x0=k3_initial_guess,tol=1e-3)
```

In Matlab we can use fminsearch and do:

```
k3_initial_guess = 1;
k3_sol = fminsearch(@objective,k3_initial_guess)
```

- Running the above will determine a k_3 that 'best' fits the given data, according to our objective. Run it and then extract (if using Python) the estimate for k_3 that it returns and report this. Does this estimate (roughly) agree with your previous estimate based on visual inspection of the objective function? Would you expect to get exactly the same answer from a new experiment on the same system? Why/why not?

(5 marks)

Part III: Flux Balance Analysis

Consider a metabolic system with the following stoichiometric matrix:

$$\mathbf{S} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 1 & -3 & 0 & 1 & 0 & -2 \end{bmatrix}$$

Suppose we want to know the vector of net fluxes $\mathbf{J}$ such that a particular flux J_2 (an element of the flux vector) is **maximised**. We also want to enforce **lower bounds of zero and upper bounds of 15 for all fluxes**. Here the fluxes are measured in units $\mu M/s$.

Recall that to find $\mathbf{J}$ we can use an optimisation method known as Linear Programming (LP). In Python we can again use the scipy.optimize library in the form

```
from scipy.optimize import linprog
res = linprog(c,A_ub,b_ub,A_eq,b_eq,bounds)
```

Matlab has essentially the same function (though it may require additional licenses...) and takes input in the form

```
J = linprog(c,A_ub,b_ub,A_eq,b_eq,lb,ub)
```

Only the format for passing upper and lower bounds on individual fluxes is slightly different (look up the documentation if you are unsure exactly how to pass the arguments - the documentation for both scipy and Matlab is fairly good). Both of these solve the general linear programming problem:

$$\begin{aligned} \min_{\mathbf{J}} \ & \mathbf{c}^T\mathbf{J} \\ \text{subject to } & A_{ub}\mathbf{J} \leq \mathbf{b}_{ub} \\ & A_{eq}\mathbf{J} = \mathbf{b}_{eq} \\ & \mathbf{l} \leq \mathbf{J} \leq \mathbf{u} \end{aligned}$$

where $\mathbf{J}$ is a vector of flux variables, $\mathbf{c}$, $\mathbf{b}_{ub}$, $\mathbf{b}_{eq}$, $\mathbf{l}$ and $\mathbf{u}$ are vectors defining the costs and constraints, and $\mathbf{A}_{ub}$ and $\mathbf{A}_{eq}$ are matrices defining inequality and equality constraints respectively. Note also that the above format requires the problem to be input as a **minimisation** problem.

We do not need all of the above arguments to solve flux balance problems - if not needed, some of these can be set to None (Python) or [] (Matlab).

- What should the arguments to linprog (c, A_ub, b_ub, A_eq, b_eq, bounds/lb,ub) be set to in order to define our problem?

(5 marks)

- Solve the resulting problem using Python or Matlab. Or, if you want, use the simple online solver at http://comnuan.com/cmnn03/cmnn03004/. Briefly describe what this solution represents in a way somebody unfamiliar with flux balance analysis would understand, and the conditions for which it is valid. State which flux(es) appear to constrain the solution the most (and why).

(5 marks)