Math 486/522 - Homework 9 - Epidemics

Fall 2024 **Len Washington III**

1. An SIR model with vital statistics was described in class, i.e. with births and deaths. The population size is fixed to be N and the relevant system of equations is:

$$\begin{aligned} \frac{dS}{dt} &= B - \beta\frac{I}{N}S - dS, \\ \frac{dI}{dt} &= \beta\frac{I}{N}S - \mu I - dI, \\ \frac{dR}{dt} &= \mu I - dR. \end{aligned} \tag{1}$$

(a) Since the population is to be fixed, define $B = dN$ and introduce fractions - you can re-use the upper case letters if you choose but remember they are now fractions.

$$\frac{dS}{dt} = dN - \beta\frac{I}{N}S - dS \qquad \frac{dI}{dt} = \beta\frac{I}{N}S - \mu I - dI \qquad \frac{dR}{dt} = \mu I - dR$$
$$\frac{ds}{dt} = b - \beta is - s \qquad \frac{di}{dt} = \beta is - \mu i - i \qquad \frac{dR}{dt} = \mu i - r$$

(b) Find all the critical points, denoted by S^* and I^*, using the scaled $\frac{dS}{dt}$ and $\frac{dI}{dt}$ equations. Problem 1b answer here.

(c) Study the stability of the Disease-Free Equilibrium (DFE) and identify the reproduction number R_0 from its eigenvalues. Problem 1c answer here.

(d) Write all the critical points in terms of R_0. Problem 1d answer here.

(e) Give an explicit condition to require the endemic critical point I^* be in the domain. Problem 1e answer here.

(f) Test the stability of the endemic critical point. Problem 1f answer here.

2. We discussed an SIS model in class with a fixed population size. If we assume births and deaths can occur and the population is not constant, then the model becomes

$$\begin{aligned} \frac{dS}{dt} &= bN - \beta\frac{I}{N}S - dS, \\ \frac{dI}{dt} &= \beta\frac{I}{N}S - \mu I - dI, \end{aligned} \tag{2}$$

This differs from Problem 1 since the birth rate b is not equal to the death rate d.

(a) Define $N(t) = S(t) + I(t)$ and derive a formula for $N(t)$. State conditions for population growth and decline.

$$\begin{aligned} \frac{dN}{dt} &= bN - dS - dI \\ &= bN - d(S + I) \end{aligned}$$

(b) Introduce fractions $x(t) = \frac{S(t)}{N(t)}$ and $y(t) = \frac{I(t)}{N(t)}$ into (2) and derive a logistic equation for $y(t)$. Note: $x(t) + y(t) = 1$. Problem 2b answer here.

3. The concentration $y(t)$ in the second compartment (plasma) of a two compartment model has a functional form as the sum of two exponentials

$$y(t) = b(1)e^{b(2)t} + b(3)e^{b(4)t}, \quad t \geq 0 \tag{3}$$

where $b(j)$, $j = 1, 2, 3, 4$ are unknown parameters, which you are to estimate. The experimental data is given in the data set (column 1=time and column 2 = y) in Table 1.

(a) Use exponential peeling to approximate $b(j)$, $j = 1, 2, 3, 4$.

Assume $b(4) \gg b(2)$

$$\begin{aligned} y_2(t) &\approx b(1)e^{b(2)t} \\ \ln(y_2(t)) &\approx \ln\left(b(1)e^{b(2)t}\right) \\ \ln(y_2(t)) &\approx b(2)t + \ln(b(1)) \\ \ln(y_2(t)) &\approx b(2)t + c \end{aligned}$$

$$A = \begin{bmatrix} 0.00 & 1 \\ 0.05 & 1 \\ 0.10 & 1 \\ 0.15 & 1 \\ 0.20 & 1 \\ 0.25 & 1 \\ \vdots & \vdots \\ 4.85 & 1 \\ 4.90 & 1 \\ 4.95 & 1 \\ 5.00 & 1 \end{bmatrix}_{101\times 2} \qquad \vec{y} = \begin{bmatrix} b(2) \\ c \end{bmatrix}_{2\times 1} \qquad \vec{b} = \begin{bmatrix} \ln(10.18066949) \\ \ln(9.68532650) \\ \ln(9.57756420) \\ \ln(8.97698692) \\ \ln(9.26775331) \\ \ln(8.57867356) \\ \vdots \\ \ln(3.15725052) \\ \ln(3.09239149) \\ \ln(3.02114805) \\ \ln(2.98016748) \end{bmatrix}_{101\times 1}$$

$$= \begin{bmatrix} 2.3204907741858722 \\ 2.2706120082163834 \\ 2.2594233007808504 \\ 2.1946642937165666 \\ 2.2265409888226997 \\ 2.1492793048119670 \\ \vdots \\ 1.1497015602444218 \\ 1.1289447365524927 \\ 1.1056369081550740 \\ 1.0919795002803587 \end{bmatrix}_{101\times 1}$$

$$A^T A\vec{y} = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.15 & 0.20 & \dots & 4.85 & 4.90 & 4.95 & 5.00 \\ 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.00 & 1 \\ 0.05 & 1 \\ 0.10 & 1 \\ 0.15 & 1 \\ 0.20 & 1 \\ 0.25 & 1 \\ \vdots & \vdots \\ 4.85 & 1 \\ 4.90 & 1 \\ 4.95 & 1 \\ 5.00 & 1 \end{bmatrix} \begin{bmatrix} b(2) \\ c \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=0}^{100} \left(\frac{i}{20}\right)^2 & \sum_{i=1}^{100} \frac{i}{20} \\ \sum_{i=0}^{100} \frac{i}{20} & \sum_{i=0}^{100} 1 \end{bmatrix} \begin{bmatrix} b(2) \\ c \end{bmatrix}$$

$$= \begin{bmatrix} \frac{338350}{400} & \frac{5050}{20} \\ \frac{5050}{20} & 101 \end{bmatrix} \begin{bmatrix} b(2) \\ c \end{bmatrix}$$

$$A^T\vec{b} = \begin{bmatrix} 0.00 & 0.05 & 0.10 & 0.15 & 0.20 & \dots & 4.85 & 4.90 & 4.95 & 5.00 \\ 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2.3204907741858722 \\ 2.2706120082163834 \\ 2.2594233007808504 \\ 2.1946642937165666 \\ 2.2265409888226997 \\ 2.1492793048119670 \\ \vdots \\ 1.1497015602444218 \\ 1.1289447365524927 \\ 1.1056369081550740 \\ 1.0919795002803587 \end{bmatrix}$$

$$= \begin{bmatrix} 341.77314505 \\ 155.25674767 \end{bmatrix}$$

$$A^T A\vec{y} = A^T\vec{b}$$

$$\begin{bmatrix} \frac{338350}{400} & \frac{5050}{20} \\ \frac{5050}{20} & 101 \end{bmatrix} \begin{bmatrix} b(2) \\ c \end{bmatrix} = \begin{bmatrix} 341.77314505 \\ 155.25674767 \end{bmatrix}$$

$$\begin{bmatrix} b(2) \\ c \end{bmatrix} = \begin{bmatrix} \frac{33835}{40} & \frac{505}{2} \\ \frac{505}{2} & 101 \end{bmatrix}^{-1} \begin{bmatrix} 341.77314505 \\ 155.25674767 \end{bmatrix}$$

$$= \begin{bmatrix} 0.00465929 & -0.01164822 \\ -0.01164822 & 0.03902155 \end{bmatrix} \begin{bmatrix} 341.77314505 \\ 155.25674767 \end{bmatrix}$$

$$= \begin{bmatrix} -0.21604531 \\ 2.07730879 \end{bmatrix}$$

$$\begin{aligned}
\ln(b(1)) &= c \\
\ln(b(1)) &= 2.07730879 \\
b(1) &= e^{2.07730879} \\
&= 7.98295617
\end{aligned}$$

$$\begin{aligned}
\hat{y}(t) &= y(t) - b(1)e^{b(2)t} \\
\hat{y}(t) &\approx b(3)e^{b(4)t} \\
\ln(\hat{y}(t)) &\approx b(4)t + \ln(b(3)) \\
&\approx b(4)t + d
\end{aligned}$$

$$A = \begin{bmatrix} 0.00 & 1 \\ 0.05 & 1 \\ 0.10 & 1 \\ 0.15 & 1 \\ 0.20 & 1 \\ 0.25 & 1 \\ \vdots & \vdots \\ 4.85 & 1 \\ 4.90 & 1 \\ 4.95 & 1 \\ 5.00 & 1 \end{bmatrix}_{101\times 2} \qquad \vec{y} = \begin{bmatrix} b(4) \\ d \end{bmatrix}_{2\times 1} \qquad \vec{b} = \begin{bmatrix} \ln(-5.662465396) \\ \ln(-5.626574240) \\ \ln(-5.552914548) \\ \ln(-5.533736779) \\ \ln(-5.418825134) \\ \ln(-5.413944008) \\ \vdots \\ \ln(-1.6499334321) \\ \ln(-1.6406106123) \\ \ln(-1.6341619768) \\ \ln(-1.6183826281) \end{bmatrix}_{101\times 1}$$

$$= \begin{bmatrix} \\ \\ \vdots \\ \\ \\ \end{bmatrix}_{101\times 1}$$

(b) Graph the data points and (3) with your values $b(j)$, $j = 1, 2, 3, 4$ on the same axes. Is the fit reasonable. Compute the SSE using (4).

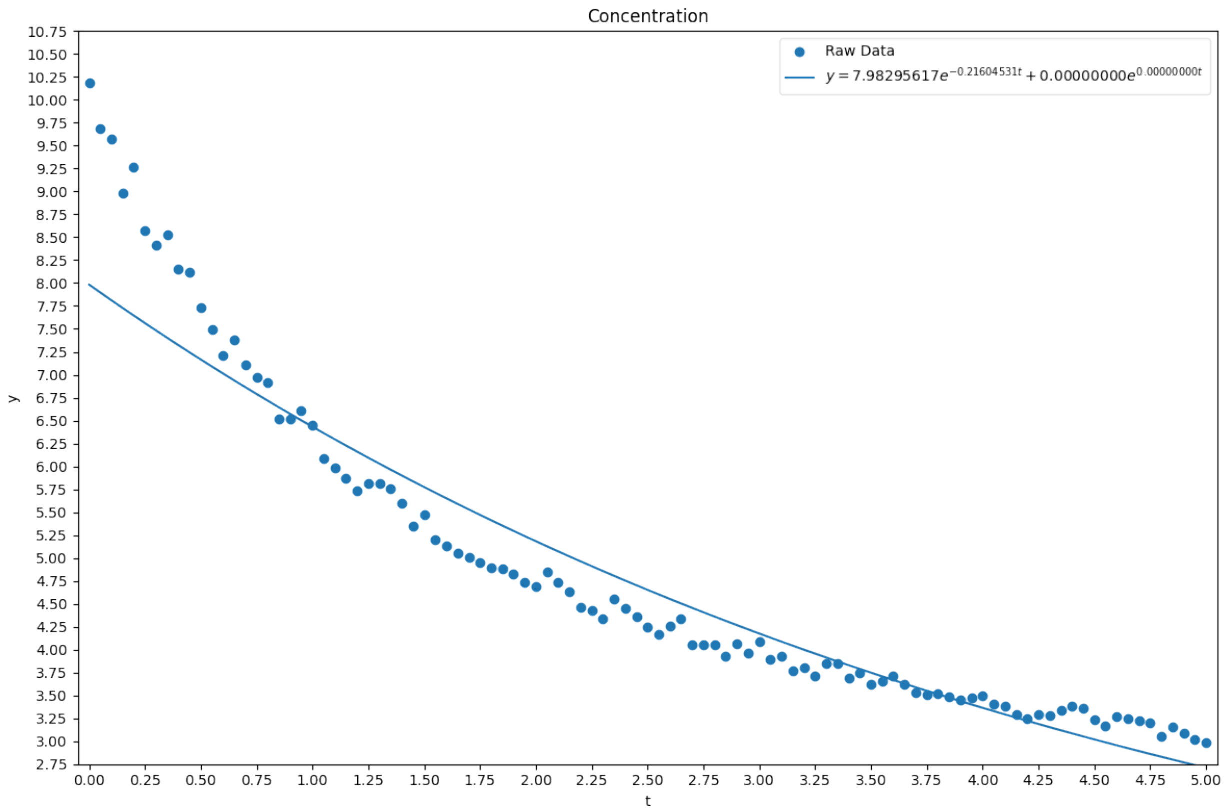

Figure 1: Data points and continuous function for Problem **(b)**.

Listing 1 Source Code for Problem 3b.

```
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
import pandas as pd


def solve_initial(raw_t, raw_b) -> tuple[float, float]:
        b = np.log(raw_b)

        A = np.array(list(map(lambda t_: (t_, 1), raw_t)))

        AT_A = A.transpose() @ A
        AT_b = A.transpose() @ b
        AT_A_inv = np.linalg.inv(AT_A)
        y = AT_A_inv @ AT_b
        b2, c = y
        b1 = np.exp(c)
        return b1, b2


def solve_second_half(t, b, b1, b2) -> tuple[float, float]:
        # x_hat = b - (b1 * np.exp(b2 * t))
        # FIXME: Some of the values are negative, figure out what to do with them
        # x_hat = np.log(x_hat)
        return 0, 0

def y_basic(t:float, b1:float=np.float64(-0.21604530753326834),
                         b2:float=np.float64(7.982956170854898)) -> float:
        return b1 * np.exp(b2*t)


def y_full(t:float, b1:float=np.float64(-0.21604530753326834),
                   b2:float=np.float64(7.982956170854898),
           b3:float=np.float64(0), b4:float=np.float64(1)) -> float:
        return (b1 * np.exp(b2*t)) + (b3 * np.exp(b4*t))


def problem3b(image_dir:Path, **kwargs):
        df = pd.read_csv(image_dir / "data.csv", header=None)
        t, y = df[0], df[1]

        b1, b2 = solve_initial(t, y)
        b3, b4 = solve_second_half(t, y, b1, b2)
        if kwargs.get("verbose", False):
                print(f"The coefficients for b(1)-b(2) are {b1:,.6f}, {b2:,.6f}, {b3:,.6
        t_values = np.arange(0, 5, 1e-3)
        y_values = tuple(y_full(_t, b1, b2, b3, b4) for _t in t_values)

        plt.figure(figsize=kwargs.get("figsize", (12, 8)))
```

(c) Use an approximation package in Matlab, Mathematica, etc. to find the unknown parameters by minimizing the sum of the squares of the error (SSE) for the model in **(a)**, i.e. minimize

$$SSE = \sum_{i=1}^{N} (y(t_i) - datapoint_i)^2 \tag{4}$$

Again graph the data and your approximate (3). How does this compare with the exponential peeling result. Problem 3c answer here.

Table 1: Data for Problem 3.

0.00	10.18066949
0.05	9.68532650
0.10	9.57756420
0.15	8.97698692
0.20	9.26775331
0.25	8.57867356
0.30	8.41627338
0.35	8.52557991
0.40	8.14847057
0.45	8.11814474
0.50	7.73210163
0.55	7.49610525
0.60	7.21145539
0.65	7.38406612
0.70	7.10966780
0.75	6.97690056
0.80	6.91499709
0.85	6.52208220
0.90	6.51737682
0.95	6.60428743
1.00	6.45108555
1.05	6.08735980
1.10	5.98447390
1.15	5.86766072
1.20	5.73590517
1.25	5.81354866
1.30	5.81578848
1.35	5.75317737
1.40	5.59196586
1.45	5.34741359
1.50	5.46917538
1.55	5.20361472
1.60	5.13529086
1.65	5.05361424
1.70	5.00315004
1.75	4.94850650
1.80	4.89647212
1.85	4.87856994
1.90	4.81989700
1.95	4.73013991
2.00	4.68719730
2.05	4.84664770
2.10	4.73156881
2.15	4.63313463
2.20	4.46573500
2.25	4.42847230
2.30	4.34159682
2.35	4.55801088